CRANFIELD UNIVERSITY

Paul Simon Lewis

The Global Vulnerability Discovery and Disclosure System: A Thematic System Dynamics Approach

Cranfield Defence and Security

PhD
Academic Year: 2016 - 2017

Supervisor: Mr J.C. Hilton
June 2017

CRANFIELD UNIVERSITY

Cranfield Defence and Security

PhD

Academic Year 2016 - 2017

Paul Simon Lewis

The Global Vulnerability Discovery and Disclosure System: A Thematic System Dynamics Approach

Supervisor: Mr J.C. Hilton
June 2017

This thesis is submitted in partial fulfilment of the requirements for the degree of PhD

ABSTRACT

Vulnerabilities within software are the fundamental issue that provide both the means, and opportunity for malicious threat actors to compromise critical IT systems (Younis et al., 2016). Consequentially, the reduction of vulnerabilities within software should be of paramount importance, however, it is argued that software development practitioners have historically failed in reducing the risks associated with software vulnerabilities. This failure is illustrated in, and by the growth of software vulnerabilities over the past 20 years. This increase which is both unprecedented and unwelcome has led to an acknowledgement that novel and radical approaches to both understand the vulnerability discovery and disclosure system (VDDS) and to mitigate the risks associate with software vulnerability centred risk is needed (Bradbury, 2015; Marconato et al., 2012).

The findings from this research show that whilst technological mitigations are vital, the social and economic features of the VDDS are of critical importance. For example, hitherto unknown systemic themes identified by this research are of key and include; Perception of Punishment; Vendor Interactions; Disclosure Stance; Ethical Considerations; Economic factors for Discovery and Disclosure and Emergence of New Vulnerability Markets. Each theme uniquely impacts the system, and ultimately the scale of vulnerability based risks. Within the research each theme within the VDDS is represented by several key variables which interact and shape the system. Specifically: Vender Sentiment; Vulnerability Removal Rate; Time to fix; Market Share; Participants within VDDS, Full and Coordinated Disclosure Ratio and Participant Activity. Each variable is quantified and explored, defining both the parameter space and progression over time. These variables are utilised within a system dynamic model to simulate differing policy strategies and assess the impact of these policies upon the VDDS. Three simulated vulnerability disclosure futures are hypothesised and are presented, characterised as *depletion*, *steady* and *exponential* with each scenario dependent upon the parameter space within the key variables.

Keywords:

Vulnerability, Zero Day, Thematic Analysis, Cyber Security, System Dynamics

PEER REVIEWED LITERATURE ARISING FROM THIS RESEARCH

The following papers have been published in peer reviewed journals and symposia in support of this research:

Lewis. P (2016) Software Vulnerability Discovery and Disclosure System: A Systems Dynamics Approach, Defence and Security Doctoral Symposium, (DSDS'16) 15-16th November 2016, Shrivenham, Wiltshire, UK.

Lewis. P & Hilton.J.C., (2015), A Statistical Analysis of Vulnerability Discovery: Microsoft Operating Systems, IET Engineering and Technical Reference, Vol 1, Issue 1, pp 7.

Lewis. P (2014), Empirical Analysis of the Vulnerability Discovery System, External DSTL Symposium, Porton Down, Wiltshire, UK [14th Nov 2014].

ACKNOWLEDGEMENTS

Reflecting in late May 2017 whilst writing these acknowledgements I am struck by the both the immense undertaking a part time doctoral research programme is, the naïve assumptions I was under at the beginning of this research and the time that it has taken to produce. I started my doctoral journey in September 2008, at different academic institution, finally joining Cranfield in 2010 – in total a period of 9 years. During that time, I have become a husband, moved to a new house, become a father - twice - essentially the top 3 stressful things one can do! Therefore I would like to extend my gratitude and absolute thanks to the following people.

Firstly, I would like to thank my PhD supervisor Jeremy Hilton, who has picked me up and dusted me off on several occasions when the odds were stacked against me. Alongside Jeremy, both Dr Ruth Massie and Dr Adam Zagorecki have help immeasurably as part of my supervisory team, without which I would have not been able to complete this research. I would also like to thank my extended support network for reading draft after draft and providing comments especially Mr Nigel Jones. Furthermore, I would also like to extend my thanks to Dr Oliver Buckley and Dr Duncan Hodges whose podcasts kept me sane though the lunacy of writing up and who's daily witticisms provided welcome relief. It also would be remiss of me not to mention the series of funders who have assisted in making this research a reality over the years: InnovateUK, JISC, Cranfield University and Crossword Cybersecurity plc.

Finally, I would like to dedicate this thesis to my family, my children Sam and Sophie, who have shown great patience, and especially to my wife Laura whose constant support has provided the time and space to complete this research. I would like to also dedicate this thesis to my late grandparents Samuel and Annie Wolstenholme whose insatiable appetite for science and technology must have rubbed off on me - they would be very excited. Thank you, x.

Daddy can come and play trains now….

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

VDDS	Vulnerability Discovery and Disclosure System
EDA	Exploratory Data Analysis
CPS	Crown Prosecution Service
ACPO	Association of Chief Police Officers
VDS	Vulnerability Discovery System
AML	Alhazmi-Malayia Logistic Model
ZDI	Zero Day Initiative
COTS	Commercial of the Shelf Software
SD	System Dynamics
SFD	Stock and Flow Diagram
CLD	Causal Loop Diagram
CWE	Common Weakness Enumeration
RAM	Random Access Memory
CVE	Common Vulnerability Enumeration
MVM	Multi Version Model
BIND	Berkeley Internet Name Domain
CGI	Common Gateway Interface
AT	Anderson Thermodynamic
RQ	Rescorla Quadratic
VBM	Vulnerability Black Markets
CCD	Code, Churn and Developer
KWIC	Key Word in Context
CAQDAS	Computer Aided Qualitative Data Analysis Software
RE	Rescorla Exponential
NVD	National Vulnerability Database
SLOC	Source Lines of Code
CK	Chidamber-Kemerer
0day	Zero Day Vulnerability
DEP	Data Execution Prevention
CCC	Complexity, Cohesion and Coupling
MTTF	Mean Time To Failure
ASLR	Address Space Layout Randomisation

“The solutions to small problems yield small rewards”

Jay W. Forrester, Industrial Dynamics, 1961, p, 449

1 Introduction

The growth of software vulnerabilities, which are defined as a weaknesses that may result in harm (ISO/IEC, 2005, p.3), has been unprecedented within the past 15-20 years. This has led to an acknowledgement that existing approaches which aim to moderate this growth, have on the most part failed (Bradbury, 2015; Marconato et al., 2012). It is argued that fundamentally new and radical approaches are needed which attempt to both reframe the complex issue of software vulnerability and offer new insight into the system and processes causing this increase. (Schneier, 2007; Schwartz and Knake, 2016) A number of historically promising techniques that combine technological, mathematical and economic approaches have been offered, however, all have been in isolation, and therefore the utility has not thoroughly been investigated, not least from a systems perspective. (Alhazmi and Malaiya, 2005; Anderson, 2001a; Finifter et al., 2013; Frei, 2009; Ozment, 2007). Despite the best efforts of academics and practitioners the relentless surge of discovered vulnerabilities continues unabated, and for all intents and purposes in an uncontrolled manner, bringing both the risk and harm this unchecked growth brings. (Alhazmi et al., 2005; Ozment, 2007; Woo et al., 2011).

This research investigates the fundamental question at the heart of the vulnerability debate; what are the factors driving the growth of software vulnerabilities within commercial-off-the-shelf (COTS) software? More specifically, what are the structures, interactions and entities within the system that is driving the increased rate of recorded vulnerabilities. Moreover, an oft overlooked aspect of this growth is the way in which disclosure of vulnerabilities are treated, as a significant amount of research focuses upon the technical approaches to discovery (Joh and Malaiya, 2010; Mcqueen et al., 2009; Rahimi and Zargham, 2013). Consequently, the aim of this research is to advance the underlying theory of software vulnerability, ground it in real empirical data and provide a systemic model to increase the understanding of policy choices upon the vulnerability system. Additionally, it is the authors' assertion and to the best of his knowledge that no one has systematically collected and scrutinised a

corpus of evidence to examine from within the vulnerability discovery and disclosure system (VDDS).

1.1 High Level Investigation Outline

This research is split into two distinct analytical sections, with the third consisting of modelling and simulation activities which builds upon the previous activities. The initial analysis consists of a qualitative investigation into the VDDS and provides a grounded analytical foundation based upon empirically observed data. This initial analysis investigates a number of factors that are hypothesised to take part and influence VDDS processes, information flows and structural interactions. This foundational analysis uses a technique widely used within psychology known as thematic analysis (Braun and Clarke, 2006). Thematic analysis is a technique that allows exploration and classification of human centred systems, and provides techniques to investigate the VDDS in a systematic manner. Following on from the thematic analysis an exploration of the numerical aspects of the VDDS is provided. This numeric examination consists of exploratory statistics, observations and remarks upon behaviours identified from the previous thematic analysis. The numerical exploration uses techniques garnered from the movement known as exploratory data analysis or EDA (Tukey, 1980). Statistics gathered and interpreted via EDA techniques provide insight as to how entities behave over time, quantification of potential variables of interest and any behavioural systemic effects. The final section brings both analytical pieces together within a system dynamics framework, presenting both simulated policy recommendations and quantitative impacts of those polices.

1.2 Why Investigate the Vulnerability Discovery and Disclosure System?

Simply put we have failed (Anderson, 2001b; Anderson et al., 2013; Schneier, 2007). This failure is centred on the inability to implement secure software development practices and standards and influence organisational incentives and to produce vulnerability free software, thus helping eliminate a large percentage vulnerability centred risk. Consequently, the potential for malicious

exploitation and subsequent compromise of information systems has increased, and, as we have now become critically dependent upon these systems the safety and wellbeing of society maybe in jeopardy (Lewis, 2006, p.11; Trim and Lee, 2014, p.45). Furthermore, it is argued by the UK government that if the software systems and services that are used to transact and transmit our information are considered 'unsafe' or 'insecure' then confidence placed in conducting business within cyberspace may diminish, along with the benefits that come with it (HMG, 2016, p.25; UK Cabinet-Office, 2011, p.40; UK Cabinet Office, 2009, p.7). Nevertheless, it is unrealistic and unfortunate to assume that society and policy will undergo an epiphany to solve the aforementioned failures.

It is a widely accepted assumption that software vulnerabilities exist due to mistakes (human or otherwise) within source code that is compiled to create software (Klein, 2011, p.3; Ring, 2015). These mistakes are traditionally classed by software engineers as software faults, or bugs, and are considered to be benign in nature (Chowdhury and Zulkernine, 2011; Hassan, 2009). Conversely, software vulnerabilities whilst closely related to software faults (and in some cases may indeed be identical), differ in the malicious way that the fault or vulnerability is activated, or to use the correct terminology exploited (Sowa et al., 2009). This exploitation leads to the circumvention of a security control within a software system, which in turn may provide privileged access to an operating system or subvert a critical security function (Holt and Kilger, 2012; Wei et al., 2010).

The act of vulnerability discovery is defined by Klein (2011) as:

> ".. the process of finding security critical software bugs" (Klein, 2011, p.2).

This definition is added to by Gallagher et al (2006) defining the process of discovery as

> "…testing attempts to find vulnerabilities in the software and to verify whether it's possible for an attacker to misuse the software program…".
>
> (Gallagher et al., 2006, p.2)

Moreover, Maurushat (2013) extends these definitions by providing specific techniques for uncovering vulnerabilities within software such as reverse engineering and traffic observation. These three definitions identify several aspects of the vulnerability growth problem, suggesting three or four areas that are of importance in the context of a vulnerability: the discovery processes, people who uncover issues and what actions are undertaken with the discovered vulnerability.

The increase of vulnerabilities is particularly acute when it is considered in terms of the criticality of the software systems that underpin the wellbeing of the of todays globally connected economic and social systems. However, given the vast and ill-defined potential scope of the problem, limited research has been performed in characterising systemic, technological, cultural, economic and social influences which are considered to have caused the growth to occur (Goldsmith et al., 2013). Furthermore, the structures that surround the discovery and disclosure of vulnerabilities have been characterised as a 'wicked problem' (Singer and Friedman, 2014, p.9). A wicked problem is defined as an ill-formulated, confused and has many entities and actors within it (Churchman, 1967). This class of problem is generally considered difficult as the solution to the problem is not immediately obvious, or is potentially counterintuitive. However, whilst difficult, the VDDS is not intractable, and with significant effort investigations can yield positive results. So, given the critical nature of software the malfunctioning of this software, either intentionally or otherwise, could if left unchecked may have significant global ramifications (Brodsky and Radvanovsky, 2011, p.189). Consequentially, if we assume that the software vulnerability issue, is by definition a global one, inclusive of the dynamics of human behaviour it is clear that approaching this in a pure software engineering or mathematical manner will be inadequate.

1.3 High Level Research Question

Given the breadth of the VDDS, the high-level research questions of this research are defined and bounded as:

What are the constituent parts, dynamics and structures of the COTS legitimate vulnerability discovery and disclosure system?

And;

What are is the potential impacts of strategies employed to reduce vulnerability centred risk?

As stated current academic research on software vulnerabilities is, and continues to be centred on new models for predicting or discovering software vulnerabilities. This is coupled with philosophical discussions around the economics of security or creating theoretical models to control risks generated by software vulnerabilities (Anderson, 2001a; Cohen, 1997; Miller, 2011; Rahimi et al., 2013). Consequentially a thorough understanding of the VDDS dynamics, structures and modes of behaviour are not well understood.

Whilst undoubtedly useful, reductive mathematical models or philosophical initiatives are akin to treating the symptom of an illness, rather than the underlying cause (Forrester, 1975, p.239). Therefore given the complex nature of vulnerability discovery it is arguably impossible to get a clear understanding of the total system by considering it via reductionist approaches (Pye and Warren, 2011, p.204). Consequently, proposed within this research is a new approach that builds upon existing theories and approaches which are then incorporated into a novel supervenient framework based upon empirically observed and analysed data.

1.3.1 Investigatory Strategy

This investigation was carried out within a mixed methods framework, bringing together the desirable features of qualitative and quantitative data gathering and analysis (Baxter et al., 2008). Mixed methods approaches have produced good results when investigating a previously unknown phenomenon. Thus, a combination of a mixed method approach and a phenomenologically based stance is advocated within this research to provide clarity on the creation of a model to describe the VDDS.

Phenomenological approaches, and resultant models, aim to capture the functional aspects of empirical observation without trying to explain the underlying processes in reductionist detail (Brodie et al., 2014; Moulin et al., 2014) The phenomenological approach when applied to complex problems is extremely powerful in subjects as diverse as Particle Physics and Leisure and Tourism. Examples of differing usage of the phenomenological approach can be found in studies by Gastmans and Shim et al (Gastmans et al., 2011; Shim and Santos, 2014). It is therefore argued that the phenomenological approach is well suited to the study of vulnerability discovery and disclosure and used to construct models of behaviour.

As a basis for investigation, the following assumptions can be made. A) that entities within the VDDS interact and undertake as yet unknown discovery and disclosure activities, which in turn cause actions to occur within the B) the driving forces behind the observed behaviours from these interactions are definable and can be modelled and C) the factors stated in assumptions A and B can be changed to provide differing observed behaviours. However, the positive nature of these interventions is at present unknown.

1.4 Research Aims

It is argued that to understand any system or process it is necessary to characterise the key aspects of that system. Specifically, the processes within that system, the entities that exist within it, the environment it sits within and external forces that act upon it (Forrester, 1975, p.2). Therefore, this research contributes to the understanding of these factors and characterises the VDDS in these terms with subsequent policy recommendations.

1.4.1 Novelty and Research Contributions

Central to this research is the creation of a novel systems approach. This approach looks at the software vulnerability discovery and disclosure system as a whole, and draws together qualitative, quantitative and system dynamic modelling techniques under a mixed methods framework. The approach outlined within this research draws upon the previous vulnerability discovery

work and extends it in several ways. Firstly, by combining concepts from previous work, the best, or most appropriate can be considered and included within the model as appropriate. Secondly, nothing is excluded from the model based on academic purity or discipline scope. Thirdly the resultant model has been constructed from data derived from the VDDS, and closely matches how the reality of the VDDS behaves. Finally, a systems approach and characterisation of the VDDS has not been undertaken previously, and is again considered unique.

The novel systems model unifies several existing, key yet unconnected theories, alongside providing empirically observed data from both a quantitative and qualitative perspective. Additionally, this research provides new knowledge and contribution in several areas, specifically:

- Systematic analysis of the vulnerability discovery and disclosure system, and constituent components and structures.
- Inductively constructed Thematic model, showing the structure of the vulnerability discovery and disclosure system;
- Construction of a System Dynamics model to simulate differing policy impacts;
- Analysis of the impacts, with suggesting of new policy strategies;
- Collection of a wide-ranging data corpus, to use in further research;
- A taxonomy of the entities involved within the vulnerability discovery system is provided.

Crucially, this research also argues that current reductionist strategies for vulnerability mitigation do not offer credible solutions for contemporary information systems. In contrast to this, this research offers novel and actionable strategies to assist in reducing software vulnerabilities.

1.5 Scope

Due to the global nature of vulnerability discovery and disclosure, and the potentially vast amount of data available it is necessary to scope the problem tightly. As such the research, will be focused upon three key elements of the

system. Firstly, the processes and discourse that constitute the vulnerability discovery systems itself. The initial discovery aspect of the system is largely uncoordinated, decentralised and globally distributed. This is due to ubiquitous nature of deployed software and the potential for millions of hackers, crackers and researchers (both internal to software vendors and externally) looking to uncover software vulnerabilities. Therefore, a representative number of empirical data sources will be used to draw out the essence of the VDDS, outlined in chapter 3.

Secondly as it is impractical to investigate all software products and all software vulnerabilities within them, at any depth the research will use a software applications to derive behaviours of vulnerability discovery and disclosure. Consequentially the Microsoft suite of operating systems have been chosen to test hypotheses set out by this research. Microsoft represents a significant proportion of all vulnerabilities found within COTS, and therefore, provides a set of behaviours upon which to test hypotheses.

Fourthly, the scope of the investigation is limited to the legitimate or 'white' VDDS as data is readily available, and relatively easy to obtain. The illegitimate VDDS, whilst important is not within the ethical approval of this research. Explicitly out of scope are technical methods for discovery of vulnerabilities, complex economic models, risk assessment or analysis techniques and technical countermeasures.

Finally, as this research is not political in nature, the ramifications of the recent WannaCryptor malware is out of scope. However, this particular malware strain is unique insofar as the vulnerability that was used to overcome the security controls was allegedly uncovered by the US government (Goodin, 2017b, 2017a). The vulnerability that was released was a zero-day vulnerability, which when used in conjunction with the malware as it infected hundreds of thousands of vulnerable legacy Windows machines. It is alleged that the US government has stockpiled this vulnerability since 2013, and had had not disclosed the details (Smith, 2017). This example serves to reinforce the need for coordinated

disclosure, whatever the rationale for discovery or indeed the stockpiling of vulnerabilities by nation states (Ablon and Bogart, 2017)

1.6 Personal Motivation

The initial motivation that inspired this research is a seminal study from researchers located within Colorado University, USA. Yashwant Malaiya and Omar Alhazmi published in 2005 a study titled 'Quantitative Vulnerability Assessment of Software Systems', (Alhazmi et al., 2005). Alhazmi and Malaiya applied a software engineering approach to characterise the growth of software vulnerabilities over time known as a software reliability growth model (SRGM). The study proposed a number of key factors that shown the growth of the number of vulnerabilities that exist within a selected group of software applications (Alhazmi et al., 2005). These factors were limited to market acceptance and total number of vulnerabilities within the software. Each of these factors were measured, counted and proposed as being proportional to the number of discovered vulnerabilities. Alongside this, Alhazmi and Malaiya (2005) postulated that three distinct phases of vulnerability discovery existed; learning, linear and plateau. Each phase is centred around specific periods within the discovery lifecycle, with discoverers learning how the software works, a period of productivity and discovery and finally saturation or exhaustion of the supply of vulnerabilities within the software (Alhazmi et al., 2005). This study marked the first attempt at quantifying vulnerability growth, and importantly speculated on the reason for observed growth.

A secondary personal motivation stems from the keen interest in cyber security globally. This interest is despite the fact that software vulnerability as a topic of academic discourse has existed for a number of years (Aslam, 1995; Bellovin, 1989; Gupta and Gligor, 1991). However, recent societal changes, for example the social media revolution, our dependence upon software for increases in both economic efficiency and global trade have made software hypercritical to the continued functioning of the global economy (Dunn, 2005; Westrin, 2001).

1.7 Thesis Outline

This thesis is organised as follows. Chapter one introduces the thesis. Chapter two provides critical analysis and synthesises existing literature and research around software vulnerability and cyber security. This chapter explores the problem and critically appraises previous studies, highlighting any limitations within the findings and conclusions. An exposition of the search methodology and key philosophical concepts within vulnerability discovery is also presented. Chapter three outlines the experimental approach and data collection methods that support the construction of the analytical and modelling phases. Chapter three also provides a rationale why the modelling technique known as system dynamics (SD) has been used and why a mix methods approach is appropriate. Chapter four provides in-depth analysis of the collected data from a qualitative stance and outlines the thematic analysis technique. It also provides the foundation for model construction. Chapter five contains an exploration of collected data and several historical reference behaviours (or modes) derived from collected data showing behaviours over time. Chapter six provides detailed causal models and sub-models to assist the understanding the structure, entities and dynamics of discovery and disclosure. Furthermore, several causal loop diagrams and stock and flow models are offered. Chapter seven contains simulation and sensitivity analysis of policy choices, and the impacts thereof. Chapter seven concludes with a range of simulation runs, based upon the observed dynamics of the system. Chapter eight provides a discussion of the results, hypotheses and relationship to work in the area. Chapter eight concludes the research and offers recommendations for future research.

1.8 Chapter Summary

This chapter provided an overview of the vulnerability discovery and disclosure problem space, research methods and research scope. By integrating three distinct yet complementary approaches EDA, Thematic analysis and S**ystem Dynamics within a mixed methods framework this research** provides a unique perspective on the VDDS. This is crucial given this

deficiency and inability for society and global organisations to tackle this complex problem of significant proportions. In the seminal book*, the Limits to Growth*, which outlines the potential collapse of the global economy and society due to pollution, food production and resource depletion we see there are alternatives to seemingly intractable problems (Meadows, 1972). First published in 1972, the Limits to Growth uses systems dynamics in a consequence and interaction model known as World3 to simulate the world's human and natural systems. Using the approach outlined in this research and utilising system dynamics we can see that integration of differing methods is suited to dealing with messy behavioural aspects of systems i.e. the way entities interact and influence to generate an overall system behaviour. As such a novel way is needed to investigate and define what VDDS is, how it operates and what entities impact upon it. Secondary to this characterisation, but no less important is the idea of positive interdiction, whereby policy decisions are made to influence how the system operates. It is these ideas of consequence and interaction that are central to this research, and subsequent policy recommendations.

2 Review of Existing Literature

This chapter presents a critical review of literature that was carried out to explore the topic of software vulnerability discovery and disclosure. This is to identify key academic studies and texts, and understand current theory and how it applied to the VDDS. One of the initial choices that must be made when embarking upon a critical review of the surrounding literature is the review approach and to weighing up both the pros and cons of each approach.

In the case of this research a traditional narrative review was chosen over more systematic or chronological approaches. A narrative review summarises key texts, presents critical analyses and finally draws conclusions from the common body of knowledge, differing from chronologic and systematic in how the review progresses – thematically. A narrative review was chosen as it provides transparent, structured and rigorous framework to draw conclusions from the literature. Narrative reviews are also well suited to some exploratory mixed methods approaches as the review provides a balanced platform to base both qualitative and quantitative theory upon and ultimately draw conclusions. The result of this process is a clear and explicit search strategy providing confidence of the final conclusions, and is less biased than a more traditional systematic or chorological approach. This is due to the transparent selection criteria that has been used to select studies. Systematic reviews were initially considered however discounted due to the exploratory nature of this research. Used in the healthcare sector, systematic reviews provide an evidence base upon which to base sound clinical judgements upon (Cochrane, 1972; Oakley et al, 2005).

2.1 Thematic Review Question

At this initial stage of the review it is key to restate the research questions, outlined in the initial chapter of this study, this is to draw out salient aspects of the literature review:

What are the constituent parts, dynamics and structures of the COTS legitimate vulnerability discovery and disclosure system?

And;

What are is the potential impacts of strategies employed to reduce vulnerability centred risk?

Within this statement there are several areas that need to be expanded upon. Firstly, an investigation of the 'constituent parts' that make up the vulnerability discovery and disclosure system is needed. How many parts exist, and what are the interaction aspects of the system and entities. Secondly what is meant by change, and what the factors or influences that are more critical than others, or are all factors equal? Thirdly what are the interrelationships between 'parts' that make up the discovery system? How strong are the relationships and how transient are they? Finally, how do the observable phenomena, such as rate of change or quantity of vulnerabilities relate to the discovery system?

To structure the investigation a series of research sub-questions (RSQ) were created to allow a systematic approach to take place:

- **RSQ1** Which structures, behaviours and interconnectivity exist within the software vulnerability discovery and disclosure system?
- **RSQ2** What are the actors/entities of the components stated in RSQ1
- **RSQ3** Are all components stated in RSQ1 equally important to the software vulnerability discovery and disclosure system?
- **RSQ4** How do identified component behave over time?

Figure 1 below shows the relationship between the sub research questions.

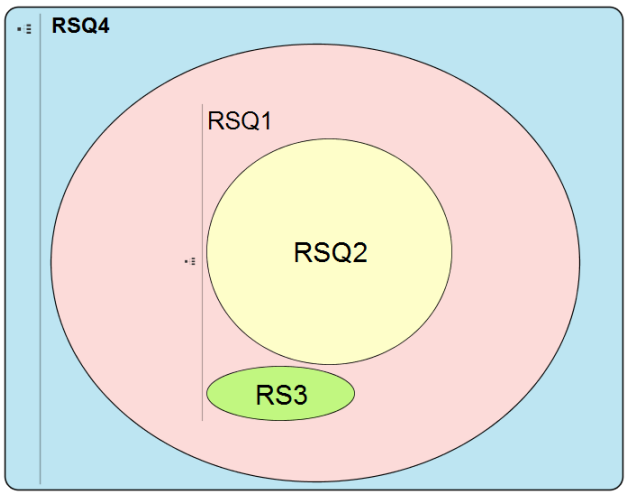


Figure 1 - Relationships of Sub Research Questions

2.1.1 Search Process and Review Concepts

The search process used to conduct this search is structured around the sub research questions, and therefore provides certain key concepts that allows us to search existing literature. This structure allows us to search wider than just specific terms related to software vulnerability. As expected most academic studies and literature that exists does so in electronic databases, such as Scopus, Google scholar, Sciencedirect, IEEE Xplore and the ACM digital Library. These resources were utilised to search for key texts and studies. Alongside these electronic databases, traditional library resources have also been utilised allowing access to non-digital literature articles. Searching electronic database provides a wide range of sources, however key texts can be obfuscated due to the sheer volume of the results that are returned. As such appropriate keyword and phraseology was used to aid in searches coupled with both synonyms and acronyms (e.g. VDM, or SLOC). Moreover, specific language and terminology were also included. Finally, combinations of Boolean operators such as AND, OR and NOT where use to expand and narrow the search as appropriate.

The use of online databases provides a critical tool in providing a good overview and set of pointers to the subsequent stages of the research as they provide indicators on existing research. As with most literature searches there is always

a level of a-priori knowledge that influences the initial stages of the search. As such initial search terms based upon the authors knowledge and experience were used to start the search. These terms, or concepts, are set out in Table 1 below.

Concept	Synonym
Software Vulnerability	{Software} Fault, Weakness, Flaw, Susceptibility, Error
Vulnerability Components	{Vulnerability} Mechanisms, Section, Module, Part
Vulnerability Discovery	{Discovery} Detection, Finding, Unearth, Recognition, Exposure
Connections	Influence, Network, Links, System

Table 1 – Initial Literature Review Search Terms

2.1.2 Inclusion and Exclusion Criteria

All sources and databases that were searched are in English, and were not limited by time, however most papers and studies were published in the last 30 years. The range of publications that were used in this literature review consist of published books, academic journals, peer reviewed research theses and government publications.

Excluded from this literature review, but included elsewhere in this research are social media sources, interviews, web based material and commercially produced marketing research. These sources are noted via the data collection exercise that gathered data sources and empirical evidence used to construct models and theory. This evidence base is separate to the literature review as it forms part of the analysis phases of this research. However, findings from within the literature review are important, such as identification of entities and probable structure of VDDS processes and guided the data collection strategy.

2.2 Overview of Software Vulnerability

Prior to entering the main review, it is worth providing general security definitions from which to work from. When first investigating software and cyber security, two key concepts emerge, risk and the environments that software operate within. Risk in general terms is defined by Chavas as:

> "representing any situation where some events are not known with certainty" (Chavas, 2004, p.5).

This definition whilst useful as a general definition of risk, it does not capture the nature of software, nor the nature of any impacts (positively or negatively). Therefore, a more nuanced and contextual definition is required. The ISO/IEC international committee via a well-known international standard offer a definition of information risk that defines information as an asset, and the components that generate risk:

"the potential that a given threat will exploit vulnerabilities of an asset or group of assets and thereby cause harm to the organization. It is measured in terms of a combination of the probability of an event and its consequence" (ISO/IEC 13335-1, 2004, p.4 s2.19).

This definition extends that of Chavas (2004), and captures three key aspects of information risk; a) potential of an event happening, b) exploitation of vulnerabilities and c) consequence. These concepts are critical when we consider the factors of software vulnerabilities growth as they potentially drive the vulnerability discovery system. Both definitions use an implicit assumption that risk is essentially a measure of uncertainty, and that this uncertainty leads to a negative impact. This assumption is pervasive throughout security literature and is a generally accepted tenet of most studies cited within this thesis (Dunn, 2005; Rescorla, 2005; Schweitzer, 2003). Within the idea of risk, ISO/IEC unpack further the constituent parts of information risk:

“Definition One: Asset, *anything that has value to the organisation;*

Definition Two: Threat, *potential cause of an incident that may result in harm to a system or organization;*

Definition Three: Vulnerability, *a weakness of an asset or group of assets that can be exploited by one or more threats”.*

(ISO/IEC 13335-1, 2004, pp.1 & 4 ,s2.2, 2.25, 2.26)

Given these definitions and the context of information risk, it is possible to frame the topic of software vulnerability. Software vulnerabilities can be defined as mistakes within source code used to compile the software, or can be introduced via the configuration and implementation of information systems (Fidler, 2014; Frei et al., 2010; Radianti and Gonzalez, 2007a; Scambray and Shema, 2009, p.49). The mistakes that are made within the software are defined as software faults, or bugs, and were initially considered benign in nature (Lewis and Hilton, 2015). Software vulnerabilities on the other hand, whilst closely related to software faults (and in some cases may indeed be identical), critically differ in the malicious way in which the vulnerability could be used to subvert a security control or countermeasure (Aslam, 1995; Lewis et al., 2015). Litchfield (2002) provides a good example whereby a sample webpage included within the Oracle database server maybe used to bypass security controls.

2.3 Anatomy of a Vulnerability – The Buffer Overrun

At this point in the review, it is worth describing the technical nature of software vulnerabilities, and characterise how they occur. Simply put a vulnerability overwrites data at a specific location within computers Random Access Memory (RAM) with potentially malicious instructions, the location of which may be used for critical system or security functions (Lewis et al., 2015; Moshtari et al., 2013). Howard and LeBlanc (2002) describe how a buffer overrun occurs:

“…when a buffer declared on the stack is overwritten by copying data larger than the buffer…user input [is] passed to a function and the result is that the

return address for the function is overwritten" (Howard and LeBlanc, 2002, pp.127 & 441).

Within the Howard et al (2002) description there a number of key principles that warrant explanation, in particular buffer, stack and function. A software buffer is an area of RAM that is used to hold instructions from a software application, this is generally a fixed size, which is defined by the software developer (British Computer Society, 2002, p.290) The stack is a list of areas within memory, normally linked, that are 'pushed' or 'popped' to store data. (British Computer Society, 2002, p.253) Finally, a function (or subprogram) is a set of instructions which are used to perform a specific task, yet is not a complete application of program (British Computer Society, 2002, p.213). The result of copying, deletion and replacement of legitimate functions within potentially key areas of the software security systems is the key aspect of software vulnerabilities and why they are of intense interest to software vendors, security researchers and cyber criminals. Vulnerabilities come in many forms, however the most well-known is styled as the 'buffer overrun vulnerability' (Lewis et al., 2015). This is defined by CWE (2016) as:

"Improper Restrictions of Operations within Bounds of a Memory Buffer" (Mitre, 2016).

Buffer overrun vulnerabilities are critical security issues, and are most prevalent of all vulnerabilities despite being simple to prevent and mitigate once discovered (Allodi and Massacci, 2014; Arbaugh et al., 2000; Murtaza et al., 2016).

2.3.1 Fault and Vulnerability Discovery Models

Vulnerabilities do not exist in a singular way, with groups of differing classes of vulnerability occurring within a software system (Ahmad et al., 2011). As such significant efforts have been made to predict and understand the progression of vulnerabilities within software (Alhazmi et al., 2005; Neuhaus et al., 2007). Analogous to vulnerability discovery models are Software Reliability Growth Models, (SRGMs) are a common parametric technique to indicate the number

of failures that may be encountered after a system goes live (Lin, 2011; Littlewood, 1979). SRGMs provide an approach to count and estimate the number of potential remaining faults within a software system (Kapur et al., 2015). SGRMs provide a good basis for the prediction of 'naturally' occurring faults (i.e. faults that are not maliciously induced) within software. Analogous to SRGMs are vulnerability discovery models (VDMs), which use the same basic principles of SRGMs and have been shown to provide good accuracy. Studies by Lewis et al. (2015), Woo et al (2011) and Kapur et al. (2015) have validated these classes of model in popular software products such as the Microsoft windows series of operating systems, web browsers and popular database systems. (Kapur et al., 2015; Lewis et al., 2015; Woo et al., 2011)

Vulnerability discovery models or VDM's are a special class of a fault discovery model which look to predict the increase of faults within software. A fault within software is concerned with the manifestation of an error (or bug) within the software, with the activation of the error via an input value, which then in turn produces an incorrect output (Musa and Okumoto, 1984). With respect to vulnerabilities Browne et al. (2001) coined the term 'Vulnerability Exploitation Model', whereby the study investigated the process of vulnerability discovery (Lewis et al., 2015). Browne et al. (2001) additionally presented a model that examined the trends within vulnerability exploitation specific software applications, namely the domain name service application BIND, and common gateway interface, CGI. Browne et al (2001) performed statistical analysis upon computer incidents using computer emergency response team data collected over a ten-year period. The conclusion suggested a linear relationship between elapsed time and vulnerability discovery, and concluded with a predicted number of vulnerabilities into the future (Lewis et al., 2015).

One of the initial vulnerability discovery models is known as the Anderson thermodynamic (AT) model, which was based on a reliability growth approach and uses time as the independent variable (Anderson, 2001a; Lewis et al., 2015). Anderson (2001a) argued that the rate of discovery is proportional to the number of tests that have been undertaken, and assumed no new

vulnerabilities are introduced via the software fix process. This concept is known as perfect debugging, with the converse known as imperfect debugging (Lin, 2011). Anderson (2001a) also suggested that a thermodynamic analogy using the concept of entropic decay best represents the amount of vulnerabilities left within the software system as a function of time (Lewis et al., 2015). However, subsequent studies have discounted this model due to the low predictive capability (Alhazmi et al., 2005, 2007; Nguyen and Massacci, 2012; Woo et al., 2011). Despite the ground-breaking aspect of Anderson's assertions it was ultimately disproved, however it gave rise to the concept of defect density, which influenced subsequent vulnerability discovery models, specifically (Alhazmi et al., 2007). Following the AT model Rescorla (2003) proposed a new method for examining vulnerability growth. Whilst the Rescorla study was not a true model for measuring the level of vulnerability with the Linux operating system, it did lead to the idea that one may be able to observe the amount of vulnerable servers as a function of time (Rescorla, 2003). Rescorla (2003) additionally proposed two further models, the Rescorla exponential (RE) model and Rescorla Quadratic (RQ) model (Lewis et al., 2015; Rescorla, 2005). The Rescorla study focused and evaluated OpenSSL v2.0, a free encryption library within Apache to secure web traffic and Redhat Linux 6.2 (Lewis et al., 2015). Both models utilised the National Vulnerability Database (NVD) datasets and demonstrated that once an exploit is released into the wild, and a patch to that vulnerability is released, the vulnerability of that deployed application exponentially decreases as a function of time, commonly known as exponential decay (Lewis et al., 2015). Rescorla based the RE model on a technique widely used within traditional reliability modelling, that of reliability growth modelling. Growth modelling is a general way to characterise the reliability of a system by employing a counting model, which records the number of events over a given time period. This counting growth process is central to the predictive aspects of reliability models and shows the progression of events.

2.3.2 Vulnerability Discovery Models and Growth Measurement

The accepted way to measure software reliability is via statistical engineering techniques used to measure and predict the quality, or robustness of software (Chowdhury et al., 2011; Garrison, 1993; Tamai and Nakatani, 2002). Software reliability models in general use probabilistic techniques to predict the growth of faults within software. This primary focus is to measure the reliability of software, and to measure the elapsed time prior to a mean time to failure (MTTF). This type of analysis is generally expanded to include time intervals of software to failure, and frequency of events (Roumani et al., 2015). The measurement of time to failure, while useful in friendly environments, does not take into consideration the malicious nature of security threats and are therefore imprecise. Software vulnerabilities are actively sought out, therefore have many external influencing factors of the discovery rate.

Vulnerability discovery models (VDMs) are grouped into two categories, effort based and time based (Younis et al., 2011). Time based models use elapsed time, generally in days, months and years, whereas effort is measured in CPU cycles abstracted to installed component or market share (Alhazmi et al., 2005). Within these models a counting method is used which focuses upon predicting the number of vulnerabilities that maybe present in software (Chowdhury et al., 2011). Most count based VDMs are univariate, with time or effort used as the independent variable. Discovery events are the dependant variable showing cumulative evolution of vulnerability discovered. A relatively small number of vulnerability discovery models exist when compared to software fault models; however, the models which do exist suggest a degree of predictive accuracy (Lewis et al., 2015; Roumani et al., 2015)

2.3.3 Alhazmi Malaiya Logistic Model

One of the well-known VDMs is the Alhazmi Malaiya Logistic model (AML) which was proposed in 2005. The AML model uses a time based counting method based upon cumulative measurement of vulnerabilities within a given software system (Alhazmi et al., 2005; Lewis et al., 2015). The model observed from collected data that three distinct phases of underlying vulnerability

discovery exist (Alhazmi et al., 2005). These phases of discovery are labelled ‘learning’, ‘linear’ and ‘saturation’, each following sequentially (Alhazmi et al., 2005; Lewis et al., 2015). Alhazmi and Malaiya (2005) hypothesised that the learning phase incorporated the familiarisation aspects of the vulnerability discovery phase. Alongside the familiarisation aspects, vulnerability discoverers were also learning to overcome any introduction of software security controls introduced by the software vendors such as Data Execution Prevention (DEP) and Address Space Layout Randomisation (ASLR) to prevent the successful activation of vulnerability code (Ablon et al., 2017, p.79). The AML model is represented by a symmetric logistic curve, transitioning from growth to decline halfway through the lifecycle of the software (Lewis et al., 2015).

$$\Omega(t) = \frac{B}{BCe^{-ABt} + 1}$$

Equation 1 - AML logistic Differential Model Source : (Alhazmi, 2006)

Following on from the initial learning phase is a secondary phase known as ‘linear’, when a period is encountered whereby the rate of vulnerability discovery tends to grow at an increasing and constant rate (Alhazmi et al., 2005, 2007; Lewis et al., 2015). Alhazmi and Malaiya (2005) argue this phase is when the software system is most prone to exploitation, due to the number of discovered vulnerabilities. This assumption is based upon a significant lag time between a vulnerability discovery event, and a patch being available and installed. In contrast Arora et al. (2010), cite numerous examples of the time lag between discovery and patch time reducing, and the potential risk therefore increasing thus suggesting the linear phase may be more complex than a simple linear increase.

The final phase, known as ‘saturation’, is where Alhazmi and Malaiya describes a decline in the rate of discovery. With this decline it is hypothesised to be an indication of the residual number of vulnerabilities left within the software to be found (Alhazmi et al., 2005, 2007; Woo et al., 2011). Alhazmi and Malaiya (2005) and Woo et al. (2011) argue that this decline in the final phase is linked

to the declining interest of hackers or crackers and increasing software security, although no evidence is presented.

A key aspect of the AML model is the notion of vulnerability density within software whereby the number of vulnerabilities is measured against the size of the software (Alhazmi et al., 2005). The vulnerability density, V_D , is given by the equation below, where S is the size or the software in source lines of code (SLOC) and V is the number of vulnerabilities discovered.

$$V_D = \frac{V}{S}$$

Equation 2 – Vulnerability Density Source: (Alhazmi, 2006)

In the PhD thesis of Alhazmi it is argued that the relationship between vulnerability density, V_D , and known vulnerabilities, V_{KD} , is related to the residual risk of potential exploitation as there would still be a number of vulnerabilities within the software, thus:

"It is actually the residual vulnerability density (depending on vulnerabilities not yet discovered) that contributes to the risk of potential exploitation. Other aspects of the risk of exploitation include the time gap between the discovery of a vulnerability and the release and application of a patch" (Alhazmi, 2006, p.16).

The AML model is the most extensively considered and validated, cited by many authors as the most consistent for vulnerability discovery prediction over time (Joh et al., 2010; Lewis et al., 2015; Roumani et al., 2015; Ruohonen et al., 2015, 2016; Woo et al., 2011; Younis et al., 2011). Many studies have confirmed the basic three phase model exists within additional datasets gathered from software vulnerability event data (Chen et al., 2010; Kapur et al., 2015). However, subsequent studies have shown that whilst the model provides significant predictive capability, there are inconsistent results depending upon the type of software and age that is under analysis (Lewis et al., 2015; Nguyen et al., 2012; Rahimi et al., 2013). For example, later versions of the Microsoft operating system Windows show that secondary and tertiary 'linear' and

'saturation' phases exist (Lewis et al., 2015; Woo et al., 2011; Younis et al., 2011). Additionally, numerous studies have confirmed the AML model closely describes the discovery process, however Joh et al., (2010) point to a subtle flaw (albeit not critical) within the AML model (Joh et al., 2010; Joh and Malaiya, 2014)

The AML model whilst useful assumes that that the vulnerability discovery process is symmetric in nature and follows a normal probability distribution. However, Joh et al., (2010) showed that in some cases the probability distribution is asymmetric or skewed (Joh et al., 2010, 2014). In the context of VDM's, skewness characterises the degree of asymmetry its mean value, with a negative skew displaying a shortened 'learning' phase, and elongated 'saturation' phase, suggesting different or modified processes at work within phases (Joh et al., 2014).

One deficiency within all studies is around the lack of investigation between VDM's and the hypothesised lifecycles. This is particularly notable given the pervasive nature of phasing within the vulnerability discovery models and the observed behaviour, it is unusual that none of the literature mentioned attempted to relate the vulnerability discovery lifecycle, and the phases of learning, linear or saturation.

Joh et al., (2010) suggest that the probability of observing discovery events is higher toward the beginning the software lifecycle, accelerating the onset of the 2^{nd} linear/saturation phases of discovery. The converse is also true with an extended learning phase, and shortened saturation phase. (Joh et al., 2010, 2014). Joh et al., (2014) provided predictive goodness of fit of scores for a Wiebull based discovery model against data form Windows 2003, Windows XP, Redhat Linux and Redhat enterprise Linux (Joh et al., 2010). The results suggest that whilst the Joh model fits slightly better than the AML model, the improvement in r^2 values is insignificant ~0.02 (Joh et al., 2010).

In addition to the work from Joh et al., (2010, 2014), Kim et al. (2007) proposed a multi-cycle or multi version (MV) model. Kim et al. (2007) studied major versions of Apache webserver and MySQL. Additionally, an aspect of code

sharing between the versions was investigated. The key differentiator from previous models is that the MV model takes into consideration new versions and patches that are added to the software over time via a process known as imperfect debugging (Lewis et al., 2015; Pham, 2000, p.126). The MV model takes a similar approach as the AML model, however where the AML models assumes a stable implementation, with no changes to the software the MV does not (Lewis et al., 2015). Kim et al. (2007) argues that addition of new code gives rise to observed secondary and tertiary 'learning and 'linear' phases of discovery. Kim provided goodness of fit statistics when fitting the MV model to Apache and MySQL data which resulted in positive results ($p>0.99$). A multistage superposition between subsequent versions of Apache and MySQL occurred, resulting in a more accurate prediction of vulnerabilities and relative rate prediction (Kim et al., 2007).

Finally, Younis et al., (2011) proposed an augmentation to the Joh et al., (2010) asymmetric model using a concept known as 'folded distribution' (Younis et al., 2011). Younis et al., (2011) examined four software systems Windows 7, Apple OSX 10.5.x, Apache 2.0 and Internet Explorer 8. Goodness of fit tests were applied to predictive trends, resulting in encouraging results with P values of >0.99. Younis et al., (2011) argued that software, which is earlier in lifecycle or with shorter 'learning' phases, is more suited to the folded model, whereas older more established software is suited to the AML model. This is primarily due to the early onset of 'linear' and 'saturation' phases, which adds symmetry to the cumulative distribution. This is in stark contrast to the MV model, which argues the inclusion of imperfect debugging is the cause of multiple 'linear' and 'learning' phases. Later in a study Younis et al., (2016) built on previous work from and Kim et al. (2007) showing a progression of thinking toward a multiple set of AML phases, that may suggest several differing behaviors within the VDDS. However, analysis stops short of speculating that these phases are of interest, or in any way related to the vulnerability lifecycle or entities within them. Many studies state that vulnerabilities should be considered a subset of software faults, as they share the same fundamental issue, that of a mistake caused by human (Kim et al., 2016; Vijayan et al., 2016) There is also

agreement that there is a level of duality, when a fault can be both a fault and a vulnerability (Kim et al., 2016; Vijayan et al., 2016). A philosophical discussion on this topic can be found in Andersons study on security vulnerability economics (Anderson, 2001a).

2.4 Software Vulnerability Discovery Process

The discovery of a vulnerability within software is akin to 'panning for gold', as the vulnerability is already in existence, it is the technique, detail and expertise that uncovers the vulnerability that is key. Studies have claimed that a software vulnerability goes through a number of distinct phases, characterised as a vulnerability lifecycle (Clark et al., 2010; Frei et al., 2010).

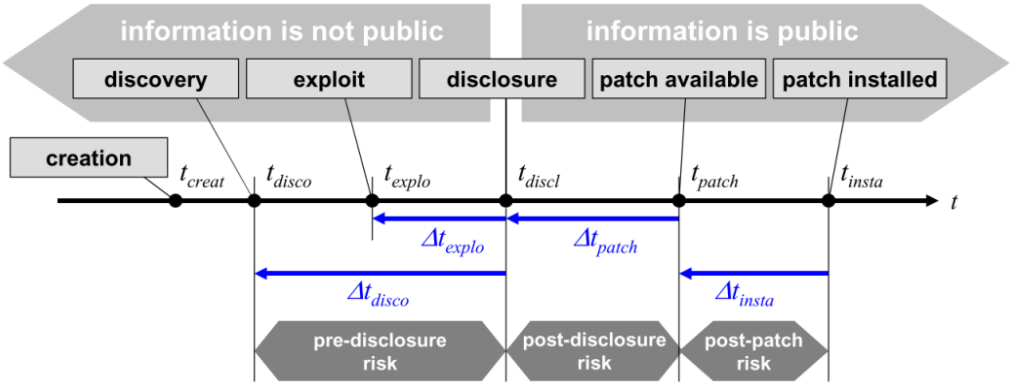


Figure 2 – the lifecycle of a vulnerability Source: (Frei et al., 2010)

Moreover, phases are argued to be distinct periods of time whereby differing events occur, and sub-processes are triggered. This claim is substantiated by Shahzad et al., (2012) who states that key phases within the lifecycle are characterised by differing behaviours of actors within the VDDS. There is consensus among authors that the lifecycle always starts with a vulnerability or bug being introduced into the software, which, in some cases may lie dormant for years. In the case of three well-known vulnerabilities, 'shellshock', 'Heartbleed' and 'goto fail' all lay dormant and undiscovered for over 24 months. (Banks, 2015; Delamore and Ko, 2015; Larsson and Sigholm, 2016). This is important when the issue of vulnerabilities is framed in the context of discovery, but not disclosure, as a separate set of actions occur.

The second phase within the lifecycle is typically the discovery phase, whereby software that contains an undiscovered vulnerability is subjected to robust probing and testing (Klein, 2011, p.3). This undiscovered vulnerability is detected via specialised discovery tools. These tools range from bespoke applications used for software security testing such as SPIKE and peachfuzz, to general programming tools such as debuggers (Klein, 2011, pp.6–8; Sutton et al., 2007, p.351). Vulnerability discovery tools are difficult to operate as they are generally designed to be utilised by highly skilled discoverers and software developers. However, the complexity of these tools has been shown to be reducing over time which may in part allows allowed potentially new entrants to the discovery system to influence the discovery of vulnerabilities. (Klein, 2011, p.181).

Once initial discovery phases have occurred, there is considerable disagreement as to which phase in the lifecycle follows, as can be shown in table 2 below. Frei (2013a) argues that the next step in the lifecycle of a vulnerability is that of exploitation, where measured time between discovery, exploit and patch is a factor. The assumption here is that that discovery of the vulnerability is via a malicious actor, driven to cause harm. Whilst this is appropriate, Frei (2013a) fails to consider the discovery of vulnerabilities by so called ‘white hat’ researchers, and internal discovery teams within the software vendor, which are deemed out of scope. In contrast to Frei, Ruohonen et al., (2016) hypothesises a seven step process which changes the sequence of events, placing disclosure directly after discovery. According to Ruohonen et al., (2016) this modification makes the assumption that the vulnerability is disclosed to a vendor, prior to any malicious exploit development. The main difference between suggested models is the sequence of events between patch, exploit creation and disclosure, however not consider the different modes of disclosure (Arbaugh et al., 2000; Cavusoglu et al., 2007; Li and Rao, 2007; Marconato et al., 2012; Ozment, 2007; Radianti et al., 2007a, 2009; Sood, 2009). In the study presented by Ozment (2007) there is the notable exception to this as a stated model includes the ‘injection’ of the vulnerability into the software prior to release. This injection phase is assumed to be within the software development

lifecycle, however this is not made clear. Within all these studies is the tacit assumption that the vulnerabilities that are discovered and result in a disclosure, and that the process is a smooth progression. However, no consideration is given to the time taken to navigate all the stages within the VDDS, and no studies have made efforts to measure the time that is taken between a vulnerability entering and ultimate exit of the VDDS.

Study	**Vulnerability Lifecycle Sequence**	**Number of Phases**	**Type of Model**
(Frei, 2013a; Frei et al., 2010)	Creation, Discovery, Exploit, Disclosure, Vendor Patch, Patch Installation.	6	Linear
(Ruohonen et al., 2016)	Birth, Discovery, Disclosure, Patching, Publication, Exploit Dev, Death	7	Linear
(Arbaugh et al., 2000)	Birth, Discovery, Disclosure, Correction, Publicity, Scripting, Death.	7	Linear
(Cavusoglu et al., 2007)	Introduction, Identification, Vendor Patch, {3rd Party Coordination, Vendor Patch}	4 (optional 2)	Linear
(Li et al., 2007)	Release, Discovery, Exploit, Vendor Patch, End	5	Linear
(Marconato et al., 2012)	Discovery, Disclosure, Patch, Exploit	4	Linear
(Sood, 2009)	Discovered, Made Public, Known to vendor, Vendor Notification, Security tool updates, Vendor Patch, Patch Known, Patch Install	8	Linear
(Ozment, 2007)	Injection, Release, Discovery, Disclosure, Public, Patch, Scripting	7	Linear
(Radianti et al., 2007b, 2009)	Creation, Discovery, Disclosure, Black Market Trade, Exploit, Patch	6	Network
(Takanen et al., 2004)	Discovery/Reaction, Correction, Disclosure, Nullification	4	Network
(Okhravi and Nicol, 2008)	Introduction, Discovery, Exploitation, Disclosure, Public Exploit, Patch, Patch Test, Patch Deploy	8	Linear
(Okamura et al., 2009)	Birth, Discovery, Disclosure, Correction, Publicity, Scripting, Death	7	Network

Table 2 - Comparison of Vulnerability Lifecycle Phases.

A further area of disagreement within the literature is centred around the disclosure phases within the lifecycle. All studies consider the disclosure phases as important, yet all identified different paths the vulnerability may follow the VDDS (Arbaugh et al., 2000; Concas et al., 2007; Frei, 2013a; Radianti et al., 2007a; Sood, 2009). The final phases of the lifecycle are centred around the installation of vendor patch, effectively removing the vulnerability from the software (Cavusoglu et al., 2007; Frei, 2013a; Radianti et al., 2007a; Sood, 2009). A detailed analysis of the disclosure phase is discussed below. All suggested vulnerability lifecycle models are subjective as they describe the journey which an individual vulnerability undergoes from a narrow point of view. The assertions from studies lends itself to a network of potential bifurcation of paths through the VDDS, each with hypothesised incentives, ethical choices, influences and timescales attached.

There is general agreement that the process a vulnerability follows is linear, progressing from creation and discovery to the eventual remediation of the issue. However, both Radianti et al. (2009) and Takanen et al. (2004) describe the differing entities and roles that are played within the VDDS and argue that the process is not linear in nature, that the lifecycle a vulnerability may contain loops, complex information flows (hidden of otherwise), and time delays.

Whilst the nature of the vulnerability discovery and disclosure system appears to differ considerably between authors, this could be due to the narrow research question that is being tackled. All studies, again with the notable exception of Ozment (2007) deal with ‘zero-day’ vulnerabilities as they are the highest impact, however other classes of vulnerabilities exist. So, called ‘one-day’ vulnerabilities which are vulnerabilities that are known about publicly, yet not patched are still critically important due to the impact of them (Ozment, 2007, p.87).

2.5 Software Quality Related to Vulnerabilities

Several studies have put forward theory describing software quality metrics that are used to quantify the quality of software by counting the occurrence of faults (Elish and Elish, 2008; Janes et al., 2006; McCabe, 1976; Nagappan et al.,

2006; Succi et al., 2003; Zhang et al., 2007). Most relevant to software vulnerability are studies that examine the correlation of software complexity, cohesion and coupling (CCC) metrics and the Chidamber-Kemerer (CK) object orientated metric suite have been applied to software security (Chowdhury et al., 2011; Lewis et al., 2015; Moshtari et al., 2013; Shin et al., 2011). One of the most widely used metrics to understand the complexity of code is Halstead's software science complexity metric (Halstead, 1977, p.127; Lewis et al., 2015). This metric is used to estimate number of errors or faults within the software by calculating the complexity within the source code, and difficultly in debugging (Lewis et al., 2015). However, Hamer (1982) and Shen (1983) argue that there is ambiguity of what can be classed as a operand and the derivation of the formula regarding effort required to debug a piece of code is very subjective and based upon unproven psychological theory. Hamer (1982) also draws attention to a lack of rigorous experimental design, the usage of non-probabilistic methods was unsuitable and a serious lack of experimental data within Halstead's (1977) study. Nevertheless, Halstead provides a good foundation for quantifying software quality, this is despite being published over 30 years ago as they serve as a timely reminder of the issues faced today.

A competing software metric known as the cyclomatic, or conditional complexity metric was proposed in the late nineteen seventies (McCabe, 1976). McCabe's cyclomatic complexity metric describes and measures the control flow of a program. McCabes technique utilises directed topologies, or graphs, to count logical decision points within the software. McCabe (1976) states that a program with a low level of complexity is easier to produce, maintain and ultimately will have a lower amount of faults (Lewis et al., 2015; McCabe, 1976; Watson and McCabe, 1996). However, Sheppard (1988) argues that McCabe's metric is of very limited utility due to the simplistic approach to decision point counting, the apparent dissonance with established software development practices and a seemingly inability to deal with modern development approaches e.g. object orientated.

Despite obvious shortcomings complexity as a general measure of software reliability has persisted with numerous theories and empirical studies refining and validating McCabe's and Halstead's initial complexity hypotheses (Chidamber and Kemerer, 1994; Douce et al., 1999). More specifically, studies have evaluated the effectiveness of CCC metrics and their utility for predicting software vulnerability. Neuhaus et al. (2007) determined correlation between software features such as code complexity and buffer usage and the number of vulnerabilities. Chowdhury et al. (2011) attempted to validate the usefulness of CCC metrics in vulnerability prediction with good success (Lewis et al., 2015). Chowdhury et al. (2011) established a significant relationship between the number of vulnerabilities present within the Mozilla Firefox web browser and CCC metrics with over a 70% success rate (Lewis et al., 2015). Alongside this Nguyen et al. (2010) established a relationship between component dependency graphs (CDG) and the density of vulnerabilities within the Javascript engine of Mozilla Firefox. Shin et al. (2011) built upon the Chowdhury study by exploring complexity, code churn and developer activity (CCD) as metrics for understanding the propensity for code to have vulnerability (Lewis et al., 2015). Moshtari et al (2013) generated CCC metrics from a range of software types, and performed classification activities upon the dataset to their establish predictive quality, with again significant results and predictive accuracy (Lewis et al., 2015).

The literature that investigates the twin concepts of software quality and software faults, however is seems that multiple authors treat the issue of fault, and indeed vulnerabilities within software as an engineering problem. Perhaps this is due to the historical nature of software and software quality originating from mathematically orientated disciplines. Therefore, all studies suffer from the serious shortcoming as they assume that issues can be engineered out, and do not address the inevitability of discovery of issues.

2.6 Software Vulnerability Disclosure

Many Studies have stated that the final or penultimate phase within the VDDS is disclosure (Arbaugh et al., 2000; Cavusoglu et al., 2007; Frei, 2013a; Frei et al., 2010; Li et al., 2007; Marconato et al., 2012; Okamura et al., 2009; Okhravi et al., 2008; Ozment, 2007; Radianti et al., 2009, 2007b; Ruohonen et al., 2016; Sood, 2009; Takanen et al., 2004). Vulnerabilities within software, and the techniques that are used to identify, or discover a vulnerability within software is largely uncoordinated and globally distributed. Once discovered by a vulnerability discoverer a number of potential disclosure strategies are available ranging from doing nothing, privately selling the vulnerability, exploit creation or simply disclosing the details to the general public (Arbaugh et al., 2000; Concas et al., 2007; Frei, 2013a; Radianti et al., 2007a; Sood, 2009).

The decision to follow a particular approach is argued to be highly subjective to the individual and is based upon previous experiences. (Cavusoglu et al., 2007; Takanen et al., 2004). The two main courses of action open to vulnerability discoverers are full disclosure and coordinated disclosure (Maurushat, 2013, p.5). The full disclosure strategy is an approach where vulnerability details are released to the general public without modification or dilution normally via online channels (Maurushat, 2013, p.5). It is argued that this approach was born out of frustration with the software vendor community as it was felt that vendors were not doing enough to prevent and fix vulnerabilities within software (Maurushat, 2013, p.6; Schneier, 2007). Conversely the coordinated disclosure movement advocates a closed security dialogue, whereby the discussion of vulnerabilities in a private manner allows software fix process to occur in a controlled and reduced risk environment (Maurushat, 2013, p.6). However, this closed approach however has come under question in recent years due to the vendors' inaction to produce a fix to issue and hostile actions toward discoverers. A recent alternative to a private coordinated disclosure movement, which is predicated on a closed approach, is paid coordinated disclosure. This strategy promotes private dialogue with the software vendor, enabling a fix to the issue to be created prior to public release however is mediated by a third party (Bradbury, 2007; Hackerone, 2016). This coordinated vulnerability

strategy is considered the first, and by some authors the only strategy to adopt when a vulnerability has been discovered as harm is reduced.

2.6.1 Full Disclosure of Software Vulnerabilities

The full disclosure movement was born out of two distinct views on what to do with a vulnerability, one philosophical in nature, the other frustration. The philosophical approach to full disclosure comes from the notion that information and software should be open, and in the case of vulnerabilities the wider the discursive audience the better. The second however negates a number of key aspects of vulnerabilities, namely; the harm to the users of the software and professional conduct of the discoverer (Matwyshyn et al., 2010; Takanen et al., 2004). Full disclosure is defined as:

"…detailed discussion and announcement of computer security vulnerabilities: what they are, how to exploit them, and how to fix them" (Arbaugh et al., 2000).

Combined these two facets bring to bear the ethics of vulnerability disclosure. Matwyshyn et al. (2010) argue that the first concern of any individual is to avoid harming others when it is clear in the case of full disclosure the vulnerability details will be used by actors, who may decide to create malicious software for criminal intent. This is supported by evidence of the myriad of viruses, malicious software and exploitation kits that exist (Caballero et al., 2011). Whilst not all malicious activities stem from the full disclosure process, a significant proportion do according to Matwyshyn et al., (2010). Clark et al. (2010) state that between 49% and 66% of malicious software is derived from this type of disclosure stance. Proponents of the full disclosure stance argue that forcing the vendors to act, due their inaction, is argued to be less harmful in the long run as it forces up software quality due to potential reputational damage to vendors (Martin, 2000a; Schneier, 2008). Nonetheless, this approach is not without criticism, as it effectively hands over critical information about weaknesses within software to malicious actors, potentially affecting millions of end users globally.

2.6.2 Coordinated Disclosure

The disclosure of vulnerabilities is considered to be in most cases a complex set of steps that need to be managed carefully as the potential negative impact on information system could be significant (Bradbury, 2007; Tang et al., 2016). An alternative to the full disclosure strategy exists known as coordinated, or ethical disclosure. Coordinated disclosure is defined as:

"...timely, fair, and accurate disclosure of the existence of security vulnerabilities that put consumers, business partners, and the social system at risk, thereby enabling these affected parties to mitigate their exposure to information risk" (Matwyshyn et al., 2010).

The confidential disclosure of a vulnerability to a software vendor once discovered allows the vendor to produce a fix to the issue that has been uncovered, therefore mitigating risk of exploitation (Arora et al., 2010). However, this mitigation is based upon the assumptions that vendors will both create a fix for the vulnerability and the vulnerability is not rediscovered independently during the coordinated disclosure period (Ozment, 2007, p.90). Whilst the rediscovery of a vulnerability is largely an uncontrollable event, production of a fix is firmly within the control of the software originator. It has been widely documented that the process which vulnerability discoverers enter into, and the time which the vendors take to process the received vulnerability information is extremely variable (Algarni and Malaiya, 2013; Clark et al., 2010; Fidler, 2014; Ruohonen et al., 2016; Schneier, 2007). This variability manifests itself at one extreme as efficient, welcoming and consists of responsive communication with vulnerability discoverers and quick production of a fix to identified issues (Damontoo, 2012). At the other extreme, discoverers are often ignored, ridiculed or threatened with legal action culminating with very significant time periods between initial disclosure and a fix being produced, if at all (Cavusoglu et al., 2007; Clark et al., 2010). It is claimed by studies that this apparent volatility, coupled with the potential for punitive actions from originators has forced the discoverer to adopt a full disclosure strategy, despite the willingness to disclose without recompense or reward (Sowa et al., 2009).

2.6.3 Vulnerability Markets

Given the variability of disclosure, and apparent dissonance within the VDDS a new mechanism was sought to minimise the risk from uncoordinated vulnerability disclosure and apparent lack of engagement of vendors – vulnerabilities markets. Markets for vulnerabilities are split into two main types, legitimate, illegitimate (Anderson, 2001a). Legitimate software vulnerabilities markets are a relatively recent phenomenon which have been the subject of many recent studies (Ablon et al., 2014; Anderson et al., 2013; Böhme, 2005; Egelman et al., 2013; Fidler, 2014; Goncharov, 2014; Grier et al., 2012; Lewis et al., 2015; Miller, 2007; Schechter, 2004). Bohme (2005) suggests that four legitimate markets exist; bug challenges, vulnerability brokers, exploit derivatives and cyber insurance. The utility of legitimate markets has been the subject of scrutiny since the initial concept of rewarding vulnerability discovery with payment was initiated in 2007 with the ultimate establishment of schemes such as Hackerone and Bugcrowd (Bradbury, 2007; BugCrowd Homepage, 2017; Hackerone, 2016). Illegitimate, or black markets are defined by Ablon et al. (2014) as:

"the collection of (skilled and unskilled) suppliers, vendors, potential buyers, and intermediaries for goods, or services surrounding digitally based crimes".

Ablon et al., (2014) go on to state that black markets are organised for the purposes of cybercrime, and optimised as such, with the intent of the participants within the market as the main driver:

- Individuals or small groups: intent is for financial gain;
- Organised criminals: intent is for financial gain;
- Nation State: Intent is to monitor, exploit or attack threats;
- Cyberterrorists: intent is to degrade, destroy, disrupt, deny or deceive;
- Hacktivists: intent is for notoriety or visibility.

(Ablon et al., 2014; Lewis et al., 2015)

2.6.4 Illegitimate Markets (Black)

Notably a series of studies have speculated on the structure and motivation of vulnerability black markets (VBM). Published between 2007 and 2010 these studies utilised a System Dynamics approach to understand the VBM and provided insight into the potential dynamics of underground black markets (Lewis et al., 2015; Radianti, 2010a; Radianti and Gonzalez, 2006; Radianti et al., 2007b, 2009). Additionally, the studies dealt with various scenarios that showed potential policy effects upon black markets (e.g. increased risk of capture from law enforcement). The studies initially focus on VBMs as a closed system with no exogenous influences or factors (Lewis et al., 2015; Radianti et al., 2006, 2007a). The proposed initial model was subsequently improved upon to include the fluctuations of vulnerability researcher reward and ethical strategies, creation of exploitation software, disclosure, supply and demand, patching. The series of studies culminated in a study outlining the social behaviour within online black markets, (Lewis et al., 2015; Radianti, 2010b)

Each study advanced and documented a series of potential structures that black markets can adopt and the dynamics within them. Whilst undoubtedly useful, several shortcomings are assumptions are present within the studies which in some areas undermine the validity of the research. Models presented in Radianti et al. (2007a) and Radianti et al. (2009) omit three critical aspects of vulnerability discovery. The first is that the inherent software quality that gives rise to the initial set of vulnerabilities within software or the software vendor themselves is not addressed. This critical omission undermines the model as it does not take into consideration different types of vulnerability as the model treats each vulnerability as a discrete event having the same severity and characteristics (i.e buffer overflow, XSS etc). Within this is an assumption that more vulnerabilities means more patches and exploit development. Secondly, the capability of the vulnerability researcher is omitted. This is equally critical as it has been noted in several studies that a small ‘elite’ of researchers are responsible for the discovery of a disproportional amount of vulnerabilities. Furthermore, factors such as vulnerability discovery tool availability and discovery techniques being sharing between researchers are not addressed.

Finally, and potentially most significantly an assumption is made that the discovery process follows a linear process over the lifespan of the software with no time delays. The omission of any detail on time, versions or debugging assumptions are a useful as simple approximation; however these have been shown to be inaccurate when describing real-world discovery events (Joh et al., 2010; Kim et al., 2007; Younis et al., 2011, 2016).

Alongside this, is the notion of nation state stockpiling vulnerabilities for usage within intelligence and military organisations. This is particularly acute, when we consider the WannaCry malware, and alleged theft of US government developed vulnerabilities (Ablon et al., 2017; Goodin, 2017b). It is suggested that vulnerabilities were developed by the US government, and weaponised to to be used within malware development (Goodin, 2017a).

2.7 Chapter Summary

This chapter provided a review of the literature that surrounds the topic of vulnerability discovery and disclosure. In summary, it has been shown from this review that software vulnerability and associated models, frameworks and theory is an important, yet a small subset of the wider computer security field. As such the number of studies that have been conducted is relatively small in comparison to older, more established disciplines such as cryptography. Consequentially, the literature search that was undertaken uncovered a lack of in-depth socio-technical orientated empirical studies, and no grounded system studies employing mixed method approaches to software vulnerability discovery and disclosure. It is suggested that the existing literature is scarce due to the technical nature of the previous authors focus, and comparative immaturity of the domain.

Since the series of studies published by Alhazmi and Malaiya (2005) their AML model has become the standard for representing the evolution of vulnerability with software systems. The AML model has been adopted, enhanced and validated by several studies, all of which accept the tacit assumption that software evolves in a similar fashion over time, and this evolution consists of one or more of the three phases stated. This assumption has led to obvious

analogues between software reliability growth models and vulnerability discovery models. However, no studies have taken steps to investigate the underlying factors that cause this observed growth, and have only speculated on the reasons.

In general most studies characterise the vulnerability discovery and disclosure process as a linear set of events, and fail to address the question as to how the vulnerability discovery process operates at sufficient depth (Clark et al., 2010; Frei et al., 2010; Ozment, 2007; Shahzad et al., 2012; Sood, 2009). However, studies do provide insight into parts of the discovery process, and linkages between differing phases. The critical flaw in most studies is the general claim that the phases follow a sequential flow, and that these phases are distinct, yet none have investigated this from a wide scope. Furthermore, methodological issues within literature raise questions on the diversity of data that these conclusions have been reached. For example, in Clarke (2010) the National Vulnerability Database is used in conjunction with commercial closed dataset, however only a subset of exploited vulnerabilities is used to base their conclusions upon. A further issue with the literature is around the claims made by authors in follow up activities. In all studies, the activities stop at a time defined by the researchers, normally due to availability of data. This is also compounded by studies as no follow up studies are undertaken to assess the validity of claims made, with the exception of Alhazmi et al. (2005) series of papers (Alhazmi, 2006; Alhazmi et al., 2005, 2007; Woo et al., 2011).

It is stark that the only appreciable foray into any investigation into a systems approach to vulnerability discovery are the Radianti studies and a single presentation at the hacking conference Blackhat 2015 (Moussouris and Siegel, 2015). These studies are the only appreciable example of investigating the vulnerability discovery mechanisms from a systems perspective. Even so, Radianti et al. (2007) only considered the black-market trade of vulnerabilities alongside the legitimate market. Whilst the Radianti findings are cogent, they lack detailed empirical evidence as to how the system works, and why interactions exist as they do. Furthermore, the structure of the system will have

changed significantly due to new markets, discoverers and policy, rendering Radianti's (2007; 2010) studies out of date. Although studies undertaken by both Radianti et al., (2007; 2010) and Alhazmi and Malaiya (2005) have examined the behaviour of vulnerability discovery there has not been a definitive conclusion as to what is driving the growth of vulnerability discovery. The investigation of disclosure and discovery, coupled with a systems approach seems obvious, yet there has been no significant literature in any quantity published. This failure has been addressed albeit in a rudimentary manner by the practitioner community.

A large corpus of non-academic literature does exist outside of traditional resources. This type of literature is used to form the basis of the thematic analysis, and exists due to the heavily practitioner focus of vulnerabilities. Furthermore, the impact of time upon the VDDS has not been investigated in any depth.

Given the lack of investigation and studies on systemic approaches to the vulnerability discovery and disclosure this study will build upon and contribute to the literature and address the lack of research in this area. In the chapter that follows, a mixed methods approach is provided that sets out the investigative approach to understanding the factors that underlie the vulnerability discovery and disclosure system.

3 Research Methodology and Data Collection Overview

The following chapter presents a rationale and methodology for investigating the vulnerability discovery and disclosure system, this includes techniques for collecting, analysing and modelling. The problem of discovery and disclosure is assumed to be both a complex yet tractable one (Malone and Malone, 2013; Rahimi et al., 2013). Coupled closely with this description is the accepted fact that the vulnerability discovery and disclosure system operates as a socio-technical system, with influences arising from participants within the system (Banks, 2015; Castelfranchi and Tan, 2001; Polatin-Reuben et al., 2013). Therefore, to investigate not only the structure of the system, but the nature and behaviour of the system we must employ several analytical techniques to draw insight and conclusions from collected data.

Consequentially, we can define this investigation as both data driven and inductive in nature. As such it is useful at this point to restate the research questions, outlined in in the initial chapter of this study:

What are the constituent parts, dynamics and structures of the COTS legitimate vulnerability discovery and disclosure system?

And;

What are is the potential impacts of strategies employed to reduce vulnerability centred risk?

3.1 An Ontological View of the VDDS

Outlined in chapter two, many researchers have utilised mathematical models, engineering approaches and deductive data collection techniques to investigate and infer behaviour of vulnerability (Alhazmi, 2006; Kim et al., 2016; Radianti et al., 2006, 2007a). However, none framed their investigations in an ontological manner and addressed the philosophical aspects of the VDDS, thus providing a arguably myopic view on the VDDS. Consequentially this investigation has adopted an ontological construal, specifically subjectivism. Subjectivism is defined by Saunders et al. (2009, p.111) as:

"..social phenomena are created from the perceptions and consequent actions of social actors".

As the VDDS is socially constructed phenomenon, which is assumed to be constantly evolving, the subjective approach is well suited as it roots any conclusions that are drawn from the VDDS as constructed from the reality of participants within the VDSS. Alongside the ontological underpinnings adopted by this investigation, philosophical pragmatism has also been adopted as to fully explore the VDDS.

3.1.1 Epistemological Enquiry

According to Saunders (2009) research can capture data and interpret it correctly, however the essence of the system may be lost if we do not consider why we collect it. This is of relevance when we take into consideration the focus of this research – vulnerability growth – and previous studies. Therefore, epistemologically speaking consideration of the approach that is taken towards the investigation of the VDDS, or more specifically what constitutes acceptable knowledge of the VDDS must be made. To provide insight into the phenomena of interest Saunders et al. (2009, p.108) argue that it is crucial that the first task of any research is to choose a suitable philosophical stance when considering the nature of the research. Presently, the approach that has been chosen to investigate the VDDS is constructive, or explicitly, social constructivism. This stance allows for the building of theoretical edifice to assist in the interpretation of events and actions, whilst still considering the fluid nature of the VDDS. The reality of the VDDS is that it is viewed via social lenses, perceptions and choices, which influence the outcomes of the system of vulnerability discovery and subsequent disclosure.

3.2 Research Approach – Building Theory

As the philosophical underpinnings of this enquiry are firmly rooted in epistemological interpretation and understanding, an abductive research enquiry has been chosen. Abductive investigations place greater emphasis on the explanation of events from potentially incomplete data alongside,

exploration and interpretation and construction of theory (Akcam et al., 2011; Kock, 2007; Massie, 2015, p.45). Currently theory and understanding that surrounds the VDDS exists within a mathematical, or inductive paradigm, which is derived from the engineering disciplines upon which modern computing and information systems have originated. However, given the advent of the internet and enabling technological focused systems that exist online to become social constructs, we must now adopt a different, broader abductive approach to the construction of theory.

3.2.1 Methodology – Observation of VDDS Phenomena

The selection of an appropriate methodology, and philosophical approach to investigations aids both rigour and depth, and provides key insight into the phenomena of interest. As such the methodology that has been chosen to investigate the VDDS is phenomenological in nature due to the passive observational character of the data and the opaqueness of the system. A phenomenological approach is defined as:

“defined meaning for several individuals of their lived experiences or a concept or phenomenon, describing what participants have in common as they experience a phenomena”.

(Creswell, 2013, p.58)

Given this definition, an initial aspect of the VDDS is the number of unidentified elements within it. Therefore, initial questions to consider are centred around control mechanisms of the VDDS, participants within it and any behaviours exhibited. As stated, central to this research is a constructive research approach, combined with an observational phenomenological stance as we are observing a phenomenon. Hence when approaching the problem on the scale and complexity of the discovery and disclosure system one needs to take an expansive and exploratory philosophical and approach to it, yet appreciate the ontological nature of the phenomena.

The construction of a theory as to how the VDDS operates is the focus of this research. Therefore by using a constructive approach to both problem

identification and production of a solution the impact can be measured on the current states of affairs (Pasian, 2015, p.94). This approach is in contrast to a pure phenomenological enquiry as to do so would limit construction of theory to a simple of the study of a phenomena, and loose potential richness (Creswell, 2014, p.85; Vagle, 2016). The abductive nature of this research, coupled with the stance of problem identification and resolution have been integrated. This is due to the opaque nature, and incomplete state of data describing the system that surrounds discovery and disclosure. These approaches provide a way to construct a theory, or partial theory, as to what is driving an observed phenomenon – the growth in disclosure of vulnerabilities with in software. When coupled with the abductive approach this dual method provides a powerful philosophical underpinning as it allows for the construction of theory from observing participants of the system interacting. It is accepted that as modelling is generally considered an abstraction of reality, therefore, the way in which we approach the problem is an important consideration, as the approach is the embodiment of the techniques used to model it.

3.3 Investigative Method – Mixed Methods

So far the philosophic and methodological approaches to the research have been discussed, providing mental construct to investigate the VDDS with the pragmatism and rigour required. The constructive and phenomenological approaches that underpin this research are important, as they provide a basis for continued investigation, however a practical framework is required. Given the importance of exploration and investigation that is central and *critical* to this research as it follows that analytical and practical research frameworks embodying this must be chosen to fit the mental model.

To fully investigate the VDDS both quantitative data and qualitative observations must be made. Using both qualitative and quantitative data together gives a richness of perspective upon the VDDS with the embodiment of this approach known as mixed methods (Creswell, 2014, p.86). A mixed approach incorporates elements of both qualitative and quantitative analysis, permitting the use of two distinct forms of data providing deeper insight into the

research area (Creswell, 2014, p.89). By utilising this mixed approach, it allows the exploration of the research problem with much greater granularity, and from two potentially conflicting viewpoints. Once adopted the mixed method approach requires the selection of techniques to be used within the framework, specifically within qualitative and quantities analysis phases.

Mixed methods when used within academic research are commonplace within the humanities, yet when consideration is given to their use within ‘hard science’ disciplines such as computer science and engineering it is rare when contrasted with existing studies considering the VDDS (Alhazmi, 2006; Alhazmi et al., 2005, 2007; Rahimi et al., 2013; Woo et al., 2011).

3.3.1 Mixed Methods - Approach

The mixed method approach lends itself well to the understanding of complex and ill-defined issues, such as the VDDS. A key strength in using mixed methods is the ability to map analytical relationships between key aspects of data and analytical stages (Creswell, 2014, p.90). The identification of areas of importance early in the research process provides an anchor point so areas can be revisited and explored further downstream. This ability to provide insight and build upon initial areas allows for a more holistic investigation to take place. The mixed method approach involves the collection of both qualitative and quantitative data, analysis of both which is integrated via merging and connecting both sets of data and informed by philosophical underpinnings (Creswell, 2014, p.565). Within the mixed framework, four differing approaches, or paths, can be selected dependent upon the quantity or quality of information available. An overview of the different approaches, and rationale is provided in Table 3 below.

Convergent Parallel	Both qualitative and quantitative data is collected and analysed in parallel, with subsequent stages of comparative analysis and interpretation.
Explanatory Sequential	Quantitative data collection is undertaken along with analysis, which is followed up with qualitative data collection and analysis and a final interpretive stage.
Exploratory Sequential	Qualitative data collection and analysis is undertaken which builds to a collection of quantitative data and interpretation stages.

Table 3 - Mixed methods approaches (adapted from (Creswell, 2014, p.224))

The specific approach adopted for this research is *Exploratory sequential*, with a bespoke system dynamics modelling, simulation and testing stage incorporated into a final analysis stage. The rationale for choosing an explanatory sequential configuration of the framework is due to incomplete or unknown availability of data and currently ill-defined structure of the VDDS. The exploratory approach allows for the themes and properties of the system to 'emerge' throughout the initial qualitative analysis stage, which is refined to establish a foundation upon which to build the subsequent quantitative data gathering and analysis stage. A further justification for this approach is the extraction of themes and interactions between entities, allowing for the construction of qualitative diagrams, to illustrate the structure of the VDDS. This in turn provides a basis for selecting and exploring key variables within the secondary quantitative phase of the process.

The final stage, incorporated within the analysis is the construction and testing of a model, derived from the previous data collection and analysis steps. This validity test provides a critical step to build confidence in the model by testing and refining it with empirical observations. Whilst the explanatory sequential method provides structured data gathering and analysis stages this is only part of the solution. The final bespoke stage that extends the framework defined by Cresswell (2014 p. 224) is the ability to model and simulate a system. This is achieved by adding two additional steps to the framework, modelling and simulation. Taken from the system dynamic approach these two supplemental steps complement and extend the mixed methods approach as it takes previous

analysis, based upon collected evidence to formulate the dynamic hypothesis upon which a formalised map of the causal relationships can be created, modelled and simulated.

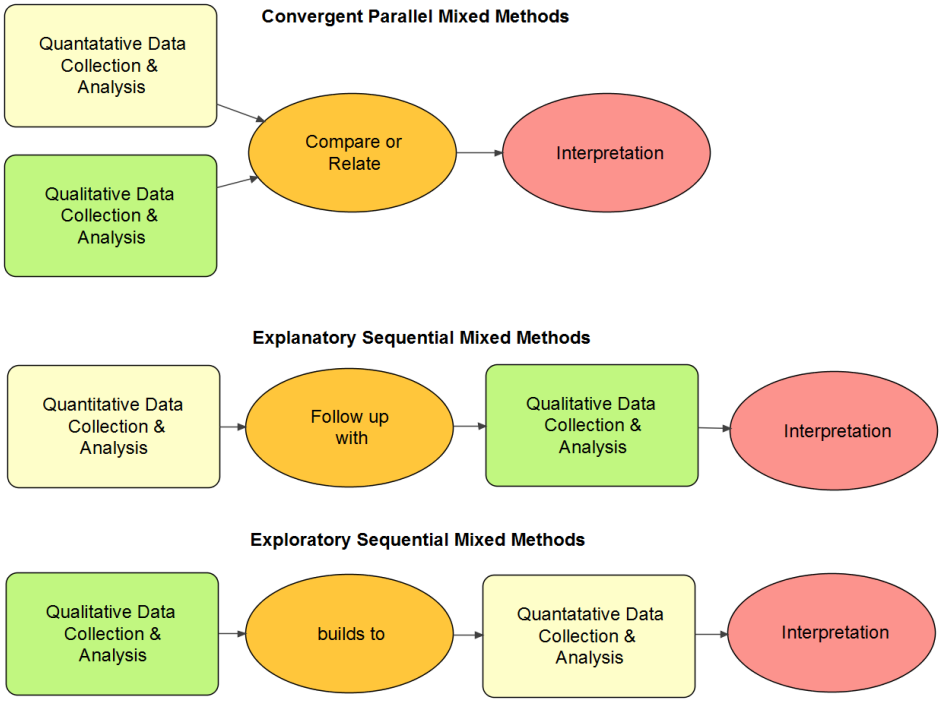


Figure 3 – Overview Research Flow Adapted from (Creswell, 2014, p.574)

Centred upon the notion that researchers report reoccurring patterns and themes that are within the data, thematic analysis directs that these patterns are codified in a rigorous and systematic manner as being important to the phenomena under investigation (Braun et al., 2006) This approach provides a framework that constructs insight and structure from seemingly disparate and incoherent data. However, the key here is not the volume of the data items, although this is important, it is the quality of veracity of the data item, known as 'keyness' also needs to be considered (Clarke and Kitzinger, 2004).

The theme of exploration and investigation is central and *critical* to this research as it allows for themes that may have been regarded as folly or may have been dissuaded against to be explored. Both approaches are within the exploratory mixed research framework thus enabling both qualitative and quantitative data to be collected, analysed and used. Given the abductive and phenomenological

approach of this research and the critical nature of concepts emerging from the data, it is key to adopt a specific role when approaching the problem on the scale and complexity of the discovery and disclosure system.

3.3.2 Enhanced Research Steps

One of most powerful aspects within the mixed methods framework is provision to choose the most appropriate analytical techniques for inclusion within the quantitative and qualitative stages. However, as this analysis has expanded the traditional framework, and expanded with two additional steps an explanation how this is arranged is required. In total six steps exist within the enhanced investigation into the VDDS. The steps are: qualitative data collection; qualitative analysis; quantitative data; quantitative analysis; model construction and simulation analysis. Each step within the process maps to the mixed method framework, which is outlined below in Figure 4 below, with steps five and six added.

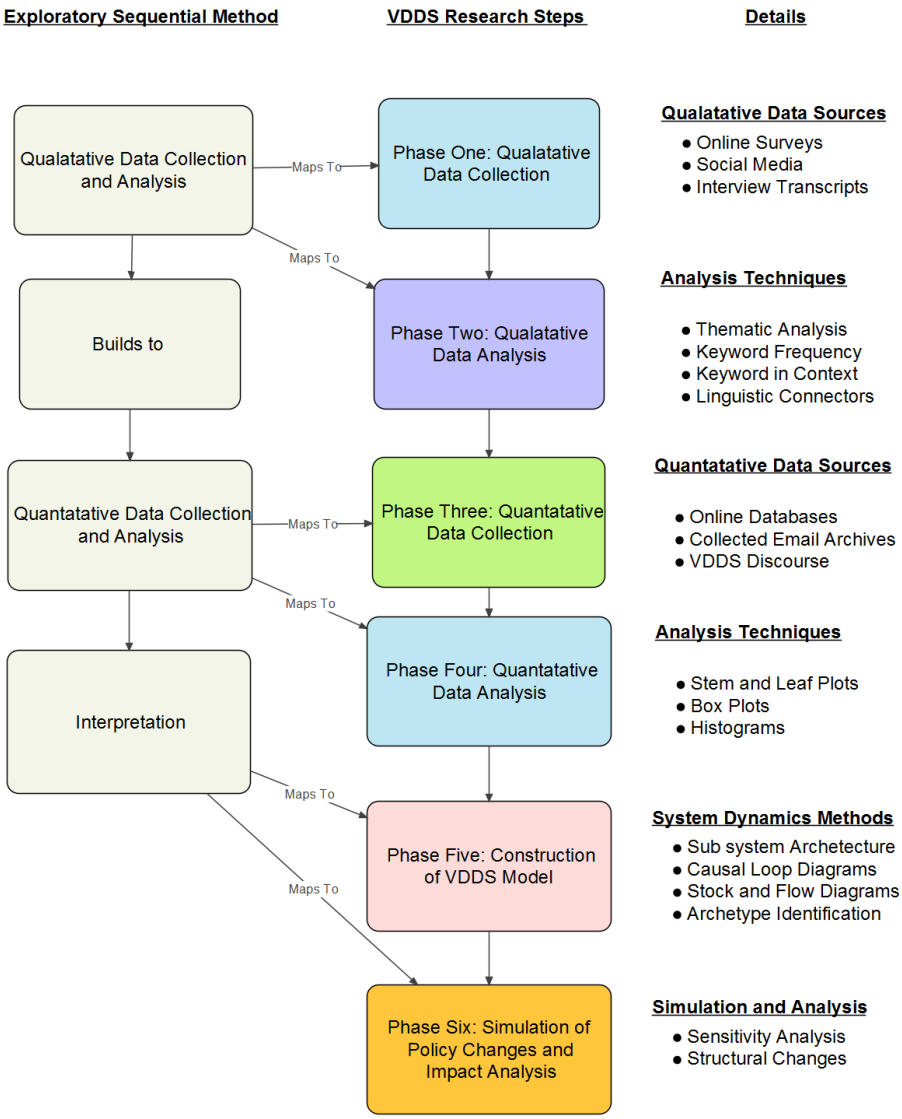


Figure 4 - Mixed Methods Mapping to Chosen Analytical Techniques Adapted from (Creswell, 2014, p.86).

3.4 Analysis Techniques

Within each investigation step is a set of techniques that have been chosen to build and lead to the next step within the analysis. The initial qualitative analysis technique that has been chosen are thematic analysis methods set out by Braun and Clarke (Braun et al., 2006). This technique provides a set of tools for the identification and reporting of patterns and themes within a data corpus, and provides a semi-organised structure to allow interpretation (Flick, 2014). The initial phase of analysis within the thematic framework is to familiarise one's self with the sources and the content of the data source, followed by coding and

identification of themes. As the exploratory nature of the research codes, themes and theories that are generated are tightly coupled to the data and are therefore abductive in nature as they are data driven and emerge from the investigation.

Secondly, the quantitative analysis framework that has been chosen is Exploratory Data Analysis (EDA). EDA is a collection of methods and techniques such as frequency distributions, visual graphs and measures of central tendency. This framework is well suited to recognising new information about the vulnerability discovery system as it allows us to explore and provide key inferences about the data to include in the creation of the model (Tukey, 1977).

3.4.1 Thematic Analysis

In keeping with the exploratory and abductive nature of this investigation, thematic analysis draws from the same philosophical movement. Thematic Analysis advocates and relies upon an inductive approach to constructing theory and mental models as to how an observed phenomenon behaves. This is in turn closely connected to the data and the analysis performed upon it (Braun et al., 2006). Widely used within the social and psychological disciplines, thematic analysis grew out of a wider field known as grounded theory, whereby one tries to establish patterns from within the data, yet is constrained by a theoretical ideal. Thematic analysis provides a 'process' on how to describe and synthesise the data, and indicate previously unknown relationships between emergent patterns and themes.

Thematic Analysis has been characterised as a poorly demarcated and rarely acknowledged analytic method which provides a systematic way to classify and codify qualitative data (Braun and Clarke, 2006). However, this critical appraisal is not due to the inadequacies of the method, but are due to the unstructured way in which the thematic method guides the analyst toward a working model (Braun et al., 2006). The thematic analysis framework provides a flexible research approach that produces rich, detailed and complex account of the phenomena under study. Braun and Clarke (2006) suggest a six-phase

approach to undertaking a thematic analysis approach to qualitative data analysis, which whilst at first inspection maybe considered linear is in fact a recursive process. The six phases are outlined in Table 4 below (Braun and Clarke, 2006).

Phase	Description
Familiarizing yourself with your data:	Transcribing data (if necessary), reading and re-reading the data, noting down initial ideas.
Generating initial codes:	Coding interesting features of the data in a systematic fashion across the entire data set, collating data relevant to each code.
Searching for themes:	Collating codes into potential themes, gathering all data relevant to each potential theme.
Reviewing themes:	Checking if the themes work in relation to the coded extracts (Level 1) and the entire data set (Level 2), generating a thematic 'map' of the analysis.
Defining and naming themes:	Ongoing analysis to refine the specifics of each theme, and the overall story the analysis tells, generating clear definitions and names for each theme.
Producing the report:	The final opportunity for analysis. Selection of vivid, compelling extract examples, final analysis of selected extracts, relating back of the analysis to the research question and literature, producing a scholarly report of the analysis.

Table 4 – Thematic Analysis Process Overview (Reproduced from Braun and Clarke, 2006, pp 87).

Within each phase of the Thematic Analysis process the act of searching and iteration occurs actively provoking continued identification of patterns and meaning within the data. Furthermore, the action of reading and re-reading provides a way to become intimately familiar with the data (Braun et al., 2006). Once the initial phases of the thematic method have been completed, initial codes that describe the investigated phenomena (in our case vulnerability discovery and disclosure) are generated. These codes form the basis of the analysis, which in turn provide insight into the themes and reoccurring categories. The final phases of the analysis method provide time to reflect upon the generated codes and themes, thus allowing for the creation and review of a thematic map to define identified themes.

3.4.1.1 Coding and Theme Identification

Thematic Analysis relies heavily upon the act of coding, and therefore the choice of which coding paradigm is adopted when performing analysis is crucial. Flick (2009, pp.306–322) suggests two main approaches to coding, inductive or deductive coding, with the former being data driven and codes emerging from the data, and the latter being formed around a predefined idea. In both cases the choice of coding paradigm provides evidence which in turn helps construct themes and identify patterns within data. Identified patterns and themes may then be used to create a model of the phenomena under investigation.

With respect to this research the adopted coding approach is inductive as it matches the philosophy of the research, and is therefore driven by the data. An inductive approach ensures that themes which are identified are strongly linked to the data itself (Patton, 1990). The inductive coding approach was chosen as it encourages a deep understanding of the research data that is collected, provides a clear link to the identified themes and patterns, and ultimately theory that is created. The inductive approach when applied to the VDDS provides strong linkage to identified themes allowing for data items themselves to be identified and queried and revisited if required (Patton, 1990; Taylor, C Gibbs, 2005).

Once a coding method has been selected, several guidelines are available which suggest the ‘best’ or most appropriate way to code, dependent upon what is under investigation. Ryan and Bernard (2003) cite several areas to focus upon when coding, set out in Table 5 below.

1	**Behaviours, specific acts**	**Seeking reassurance, Bragging**
2	Events – short once in a lifetime events or things people have done that are often told as a story.	Wedding day, day moved out of home for university, starting first job.
3	Activities – these are of a longer duration, involve other people within a particular setting.	Going clubbing, attending a night course, conservation work.
4	Strategies, practice or tactics.	Being nasty to get dumped. Staying late at work to get promotion.
5	States – general conditions experienced by people or found in organisations.	Hopelessness "I'll never meet anyone better at my age" settling for someone who is not really suitable.
6	Meanings – A wide range of phenomena at the core of much qualitative analysis. Meanings and interpretations are important pars of what directs participants actions.	
	a. What concepts do participants use to understand their world? What norms, values, and rules guide their actions.	The term 'chilling out' is used by young people to mean relaxing and not doing very much.
	b. What meaning or significance it has for participants, how do they construe events what are the feelings.	Jealousy "I just felt why did she get him".
	c. What symbols do people use to understand their situation? What names do they use for objects, events, persons, roles, setting and equipment?	A PhD is referred to as 'a test of endurance' (because finishing a PhD is a challenge).
7	Participation – adaptation to a new setting or involvement.	About new neighbours "In my new house I have to keep my music down at night as the neighbours have young children".
8	Relationships or interaction.	Seeing family " Now my sister lives in the next road she visits more and we've become much closer.
9	Conditions or constraints.	Loss of job (before financial difficulties), moving away (before lost contact with old friends).
10	Consequences.	Confidence gets dates, positive attitude attracts opportunities.
11	Settings – the entire context of the events under study.	University, work place, housing estate.
12	Reflexive – researcher's role in the process, how intervention generated the data.	Probing question "How did you feel"?

Table 5 – Coding Behaviours Adapted from (Ryan and Bernard, 2003)

Using the criteria outlined in Table 5, we can adopt a systematic approach to coding the gathered data items and structure them in an organised way. Massie (2015) developed a coding card that provide an aid memoir around what to include, exclude and examples of data that was coded. This approach allows for consistency and comparison with the coding process as it reinforces why the previous codes were selected.

3.4.1.2 Computer Assisted Qualitative Data Analysis (CAQDAS)

Coding tools can be used to manage any collected data, specifically computer based tools. Computer Assisted Qualitative Data Analysis (CAQDAS) software is a relatively recent phenomena, and are used as a set of computer based tools that allow the researcher to manage their data (Bazerley and Jackson, 2013, p.2). These tools add a level of automation to the analytical process, by potentially increasing efficiency in searching and analysis methods. Many CAQDAS tools exist, and range in sophistication, however one of the more common is QSR Nvivo. Nivivo is a CAQDAS software package that facilities for the capture, coding and analysis of multiple forms of data (Bazerley et al., 2013, p.56). Nivivo v10.0.641 SP6 was used to code all qualitative data that was collected.

3.4.2 Exploratory Data Analysis

In common with both the phenomenological research approach and the thematic analysis method, a complimentary quantitative method of analysis, which shares the same exploratory stance was chosen - known as Exploratory Data Analysis (EDA). EDA was first codified as an approach by Tukey who argued that data analysis is detective work, either numeric, counting or graphical in nature, which provides indications of how a system may behave (Tukey, 1977) Furthermore, Church (1979) states that EDA is a highly visual approach that is used to discover potentially new information about the data by utilising simple methods such as frequency distributions, visual graphs and measures of central tendency (Church, 1979). The EDA method is well suited to uncovering new information about the VDDS as it allows the exploration of key inferences about the types and behaviours of data to include.

The philosophy of exploration is central to EDA in stark contrast to traditional confirmatory data analysis (CDA), whereby inference or estimates are made about the data and system under scrutiny (Tukey, 1977). Moreover, Tukey (1980) provided an iterative evolution to EDA, which allows phasing of the types of question to ask when conducting an explorative investigation.

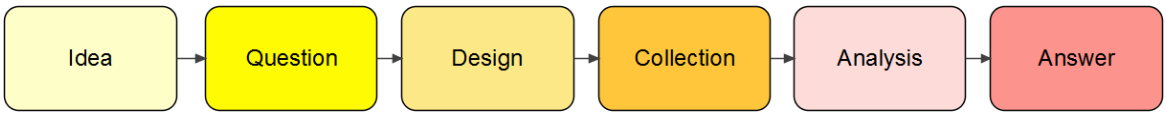


Figure 5 – Suggested EDA Exploratory Steps Source: (Tukey, 1980, p.23)

The EDA process allows the examination of data to take place without any preconceived ideas, and allows the behaviour of the phenomena under investigation to emerge from data (Martinez et al., 2011, p.3). Two typical techniques that are used to investigate data within the EDA framework are pattern discovery and visualisation (Martinez et al., 2011, p.6). Pattern discovery is the process of uncovering the structure of the data, and ultimately the phenomena that is producing the data, with techniques such as principle component analysis and cluster analysis. (Martinez et al., 2011, p.7). Whereas visualisation using statistical techniques such as tree maps, histograms and scatter plots provide ways to look at the shape, distribution or correlation of the data that may uncover important aspects of data (Martinez et al., 2011, p.6).

3.4.3 Modelling Choices and Techniques

Once both thematic and exploratory data analysis stages of the research framework have been undertaken and yielded results, the final stages are to build a model and simulate the vulnerability system. The building of a testable systemic model, rather than an reductive model, provides the capability to test differing scenarios. These scenarios take the form of changing the conditions upon which the simulation will run. A simple example of a scenario is changing the quantity of vulnerabilities which exist within a software application.

There are many approaches that permit the creation of conceptual theory, however most do not offer the capability to test differing scenarios choices, nor provide an analytical framework to build and test a model. Therefore, the choice of a modelling technique must both embrace the exploratory underpinnings of this research and provide robust tools to create and test any model. Several approaches were investigated, each with specific strengths and weakness, with the final choice being System Dynamics. System Dynamics tools and

techniques provide both the flexibility to construct a data driven inductive model and the ability to test different scenarios (Sterman, 2000).

3.4.3.1 Soft Systems Methodology

An alternative to Systems Dynamics is Soft Systems Methodology (SSM) developed in 1981 by Peter Checkland (Checkland, 1981). SSM draws upon systems theory to create a practical methodology which could be applied to ill-structured or 'soft' real world problems. Checkland (1981) defined SSM as a holistic paradigm – that of whole systems, concerned with the wholes and properties (Checkland, 1981, p.13). The philosophical underpinnings of SSM are interpretive, or hermeneutic, and give rise to the central tenet of SSM's ability to deal with real-world problems, the role of the individuals' world view (weltanschauungem). Once established this world view provides the backdrop for using a technique known as rich pictures. Rich pictures allow the analyst to bound the problem, but provide insight into how the system functions. Furthermore, this holistic view provides unique perspectives on how systems may be interconnected, and how solutions may present themselves (Jayaratna, 1994, p.178). This underlying philosophy is an effective tool for exploring political and social analysis, however once the initial findings are presented there is limited guidance on how to take things forward toward a simulation or other tests.

3.4.4 System Dynamics Overview

When one considers a system on the scale of the vulnerability discovery and disclosure system an appropriate and robust method that provides an understanding of that system is critical. System Dynamics deals with the simulation and interaction between objects in dynamic systems (Forrester, 1958). System Dynamics also provides a robust platform to bring together both qualitative and quantitative data that has been collected, utilising techniques such as causal loop diagramming, stock and flow models and time series simulations. System Dynamics also provides an analytical framework to analyse elements of a system that are organised for a purpose. and emphasises the actions of information, action and feedback (Coyle, 1996). The notion of

feedback loops is crucial within System Dynamics, and takes two forms: positive and negative. Coyle describes both loops as:

“A positive feedback process, or loop, is one which acts to reinforce a change in a system level….a negative loop is goal seeking, that is it tries to move a level toward a desired target”.

(Coyle, 1977, pp.38–40)

The System Dynamic approach also allows the investigation of how different variables affect how the system behaves over time. It allows the creation of a dynamic model as the artefact of interest, with the value of the artefact demonstrated via the execution of the model under differing conditions.

3.4.4.1 Model Construction and Simulation

The final phase within the mixed methods framework is to build upon the analysis undertaken in the previous steps and construct a conceptual model of the vulnerability discovery system. This is a critical step as it allows the visualisation of the system and identify key elements and information flows around the system. The initial representation of the system is via Causal loop diagrams (CLDs) which are based upon the thematic analysis, highlighting themes, subthemes and feedback loops that are present within the system. In the case of the vulnerability discovery and disclosure system there are several hypothesised interrelated system archetypes, information flows, causal links that come together to form a larger model.

Using the CLD technique to visualise structures and identify loops that are derived from analysis is a key aspect of this investigation. The interaction of two or more balancing and reinforcing loops is suggested to cause identified behaviours within systems. These archetypes may cause oscillations, exponential growth or ‘s-shaped’ growth within a system.

System Dynamic modelling has been used to understand real-world systems and problems since its creation in the 1960’s (Forrester, 1975). As part of the evolution of System Dynamics, many complex systems have been shown to

produce similar behaviours although the systems under scrutiny are very different; and are known to confirm archetypical dynamic hypotheses and explain generic system behaviours (BenDor and Kaza, 2012; Senge, 1990, pp.379–390)

In the context of the VDDS there are a previously observed influences and interactions within the system that are hypothesised to makeup a diverse set of loops and archetypes. However, the main behaviour that has been shown to be exhibited by the system is the S-shaped growth curve (Alhazmi et al., 2005). This S-shaped curve, known as logistic, suggest that the underlying process that causes this growth to occur is a combination of both balancing and reinforcing loops which initially drive growth, then reduces growth when a limit has been reached. Using simple archetypes is a possible way to capture the underpinnings of seemingly impenetrable levels of complexity, and the interactions of the participants who exist within the VDDS. As stated the VDDS is hypothesised to be made up of entities, interactions and delays. However, to move further and begin to understand the systemic behaviours how the VDDS operates, and crucially how variables impact each other, we must construct a model of the VDDS.

3.4.4.2 The VDDS as a Dynamic System

Within the final phase of this research and to accurately model the VDDS we must first adopt a specific nomenclature to describe the aspects of the VDDS correctly. To this end the terminology used by Voinov (2008) has been adopted when entities, relationships and interactions are described. Initially we start with describing an *element* which is considered to be a building block of a system, and is considered to have both *properties* and *features* (Voinov, 2008, p.7). Features are a distinctive property of system (Peter interacts with Paul), whereas a property is an attribute (ie colour) of a feature/ Interaction is defined again by Voinov by describing the type of relationship that may exist between elements; specifically flows of *material* and *information* (Voinov, 2008, p.9).

These definitions of interaction are key when modelling the VDDS as we are essentially looking to understand the rules of the system as a whole (Voinov,

2008, p.7). The representation of flows that are present within a system, combined with the elements are the important essence of the VDDS as they represent the disclosure choices that are made, and rewards that are given. In common with other systems that exist, several interactions have been identified within the VDDS, and these sets of processes clearly operate to facilitate the discovery and disclosure of software vulnerabilities.

Adding to Voinovs' (2008) definitions we can further characterise potential processes by using system dynamic modelling vocabulary defining two further aspects of a system, *state* and *resource.* Resources within System Dynamics are defined as is anything that has value which can be transferred from one element to another (Sterman, 2000, p.127). Alongside resource, state of a particular resource in the context of system dynamics can be defined as "any accumulation of that resource" (Wolstenholme, 1996, p.12). By applying and characterising a system in these terms it is a small, yet important leap to introduce a further term, *stocks*. Stocks within System Dynamic models are measurable amounts of a resource at any given, state and time (Wolstenholme, 1996, p.12). Applying a systems approach, nomenclature and vocabulary we can define in a broad sense the VDDS as:

'The VDDS converts raw vulnerabilities (State1: Undiscovered Vulnerabilities) into refined vulnerabilities (State2: Discovered Vulnerabilities) to be either sold for profit (State3: Money) or disclosed for free (State4: Knowledge)'.

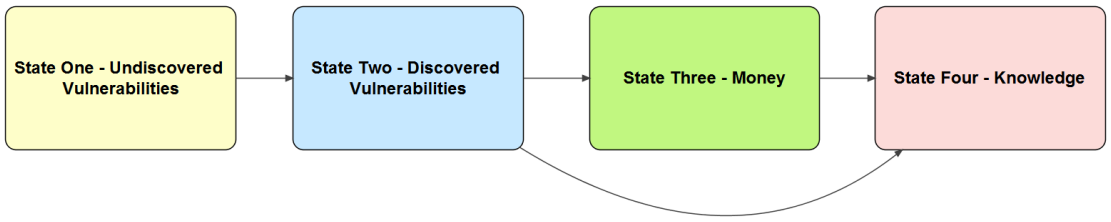


Figure 6 – State Transition of Deriving Value from Vulnerability

A key aspect of the transition between states are the rate in which the conversion between states occur (Wolstenholme, 1996, p.12). The rate at which the conversion occurs are represented within the system dynamic modelling

process as rate variables and are key to understanding the behaviour of the system. In particular, we can see that vulnerabilities transition between states linearly.

3.4.4.3 Why Systems Dynamics?

The construction theory and model that is intended to represent real-world phenomena must start at first principles, or as close as realistically possible if current theory is not adequate. It is also reasonable to assume that the any model will be continuously improved and future generations will represent reality more accurately. Given these assumptions, many modelling approaches exist to assist and suit differing classes of problem when trying to understand the behaviour of a system (Kelly et al., 2013). System Dynamics provides a robust approach for bringing together both the collected qualitative and quantitative VDDS data by utilising techniques such as causal loop diagramming, stock and flow models and time series simulations. System Dynamics deals with the simulation and interaction between elements within in a dynamic system and crucially integrates the concept of feedback whereby elements may influence each other, in positive and negative ways. (Coyle, 1996, p.2; Forrester, 1958). Coyle (1977) describes this influence in terms of processes or loops:

“A positive feedback process is one which acts to reinforce a change in a system level….a negative loop is goal seeking, that is it tries to move a level toward a desired target” (Coyle, 1977, pp.38–40).

The System Dynamic concepts outlined by Coyle (1977), Wolstenholme (1996), Sterman (2000) and Forrester (1975) for the investigation of how different structure, resources and rates affect how the system behaves over time provides a very powerful investigative framework. The framework provides tools for the creation of dynamic models, characterisation of the artefact being investigated with the behaviour of the artefact demonstrated via the execution of models under differing conditions (Sterman et al., 1997). System Dynamics also provides an analytical framework to assess elements of a system organised for a purpose and emphasises the information, action, feedback paradigm (Coyle, 1996, p.5). This concept of feedback is crucial when we consider the systemic

nature of the VDDS, as it these types of flows that ultimately govern how the system behaves and how it will continue to do so in the future.

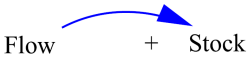
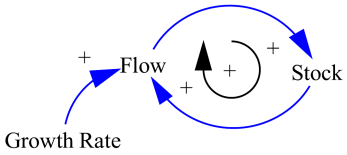
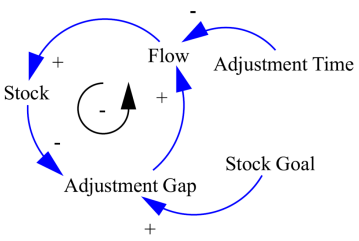
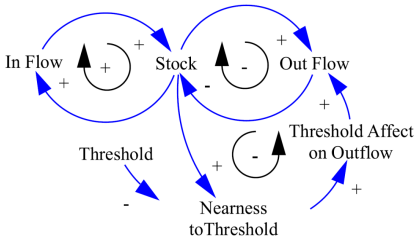
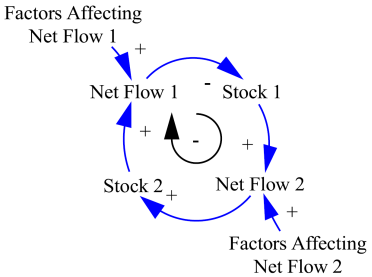
3.4.4.4 The System Dynamic Modelling Process

To model any system or process the analysts must adopt a process as to how to go about investigating the phenomena (Sterman, 2000, p.85). Consequently, several process steps have been put forward by authors outlining general steps that are followed in constructing a System Dynamic model. The most comprehensive is outlined by Sterman (2000), which brings together processes steps from both Coyle (1996, p. 11) and Wolstenholme (1996, p. 26-19):

> **Step 1: Problem Articulation** (Theme Selection; Key Variables; Time Horizon; Problem Definition)
>
> **Step 2: Formulation of Dynamic Hypothesis** (Initial Hypothesis Generation; Endogenous Focus; Mapping)
>
> **Step 3: Formulation of a Simulation Model** (Specification; Estimation; Tests)
>
> **Step 4: Testing** (Comparison to Reference Modes; Robustness Under Extreme Conditions; Sensitivity)
>
> **Step 5: Policy Design and Evaluation** (Scenario Specification; Policy Design; What if...; interaction of Policies)
>
> (Sterman, 2000, p.86)

As stated by Senge (1990) there are six modes of behaviour that have been identified within the System Dynamics movement. The archetypes are characterised by feedback loops and interactions of those loops in one or more ways. Furthermore, modes of behaviour have been characterised, and by using influence diagramming and codified so that they can be readily identified (BenDor et al., 2012; Sterman, 2000, pp.264–291). Of the fundamental systemic archetypes that exist all are grouped around the growth and decline of a value or an observed variable, for example the growth of bacteria in a petri dish (Goodwin, 1970). These dynamic system archetypes are postulated to

exist within the VDDS, taking the form of loops of information or movement of resources within the system. The commonly found archetypes, along with structures that are found within systems are outlined in Table 6 below.

	System Archetype	Governing Equation	Diagram
1.	Linear Growth	$\frac{dS}{dt} = k$	
2.	Exponential Growth	$\frac{dS}{dt} = Ks$	
3.	Goal Seeking Growth	$\frac{dS}{dt} = \frac{C-S}{k}$	
4.	Logistic Growth	$\frac{dS}{dt} = kS\left(1 - \frac{S}{C}\right)$	
5.	Oscillations	$\frac{dS}{dt} = ks$ $\frac{dS}{dt} = krS$	

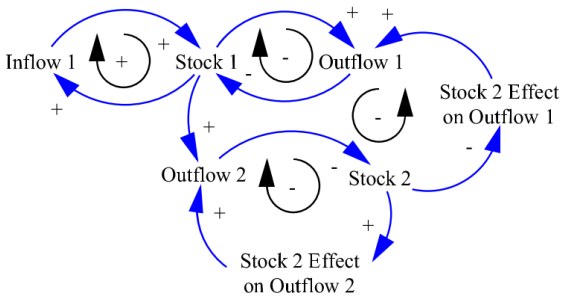
6.	Overshoot and Collapse	$\frac{dS}{dt} = k_i S - k_o S \frac{R}{C_R}$ $\frac{dS}{dt} = K_R S$	

Table 6 - Systemic Archetypes Adapted from (BenDor et al., 2012; Senge, 1990) ***Note: S = stock, C= goal/carrying capacity, k = constant, Ks/Kr = S or R related constants.***

3.5 Data Collection Method

The initial phase of the exploratory sequential framework is to collect and analyse qualitative data which then in turn provides a foundation for further quantitative data collection and analysis. Traditional thematic literature reviews provide a good foundational step towards answering these questions on the initial collection aspects, however due to the nature of the investigation, a combination of both literature and practitioner based knowledge is required.

According to Creswell (2014) a data collection approach to purposefully select data on participants, documents and material is crucial and will assist in understand the problem and research question (Creswell, 2014). Additionally, Miles and Hubeman (1984) state that four aspects are important when considering what data to collect and where it may reside. These consist of a) the setting where data maybe collected, b) actors which are involved in the phenomena under investigation, c) events that surround the problem, and d) processes that surround phenomena. Table 7 below outlines the four categories or data that were collected, and the context upon which it was collected.

	Observational	Interview
Setting	Social Media and Email Distribution Lists.	News blogs or 3rd party emails.
Actors	Vulnerability Researchers, Software Vendors, and Vulnerability Brokers.	Vulnerability Researchers or Blogs.
Events	Discovery of a Vulnerability or Disclosure activities.	Disclosure of Vulnerabilities or participation within black markets.
Process	The process of discovery or disclosure.	Disclosure or selling of a vulnerability and discourse surrounding activities.

Table 7 - Categories of Data and Contexts

3.5.1 Qualitative Data Collection: Reliability and Validity

The reliability of both the data that is collected and the analysis that is generated from the data is critical. This is even more important when we consider the foundational role data forms within this research. Flick (2014, p.387) states that the reliability of data should consider two particular aspects, the genesis of the data and the procedures that are followed to collect the data. This is alongside analytical bias introduced by the researcher is a critical consideration when investigating a phenomenon that can be influenced or observations can be biased toward the researchers preconceived ideas.

Observations of all data collected for this investigation are historical in nature and therefore cannot be influenced by the researcher. Thus, researcher bias is mitigated insofar as the raw data that is collected. However, using the first of Flick (2014) criteria, that of genesis, there is probable researcher bias as the selection where the data has been collected from. However, consideration must be given to the fact that there are limited places to collect vulnerability discovery data items. Addressing the second criterion, collection procedure, vulnerability discovery is overall an online activity, where participants and processes occur on the internet. Therefore, it is possible to observe interactions between researchers and other participants as Garfinkle (2009) states “via their internet footprint” (Garfinkel and Cox, 2009, p.1). Several actions are recorded in various forms ranging from social media posts through to discussions via email and forums. These records of online interactions provide key evidence into how

researcher and other participants behave. Procedurally the following was used to apply consistency around the collection of data:

- **Step one:** Identification of location of data items;
- **Step Two:** Selection of data items, via keyword search within the data repository;
- **Step Three:** Collection of the data, via NVivo Ncapture;
- **Step Four:** Importation into the Nvivo software package.

The search methodology that was used to gather data utilised the research question and was used to limit the scope of the search to vulnerability to focus upon related areas, such as disclosure discourse. The systematic search methodology selected, synthesised and appraised data and information that is relevant to the question. The main tools and platforms that was used to identify key data items were as follows:

- Google;
- Google Scholar;
- British Library, EthOS;
- CiteUlike;
- Seclists.org;
- Reddit;
- Searchblogspot.com;
- Hackernews.com.

The keywords that were utilised to search for data items are outlined in Table 8 below.

Software AND vulnerability	**Software AND 0day**	**Software AND Hacker**
Vulnerability AND quality	**Application AND vulnerability**	**Application AND Hacker**
Vulnerability AND Discovery	**Software AND Exploit**	**Software AND Security**
Software AND Vulnerability AND ethics	**Vulnerability AND Disclosure**	**Vulnerability AND Tools**

Table 8 – Keywords for Searching

3.5.2 Initial Data Retrieval Steps

A number of data retrieval steps were undertaken in the initial stages to establish where the most appropriate data was located. Consequentially, the eligibility criteria for inclusion within the research was initially limited to the search terms listed in Table 8. However, a small set of useful results were returned, therefore the search terms were widened thus helping identification of social media and specific repositories of VDDS discourse. Using Braun and Clarke's (2006) taxonomy for classifying data types data was arranged, labelled and assigned to categories thus enabling a more systematic data collection to occur. For example dataset in the context of this research is defined all social media posts within a set period of time, whereas a data-item is defined as singular social media post. Finally, a data extract is a coded subset of a data item highlighting a unique or common theme. The taxonomy is outlined in Figure 7 below.

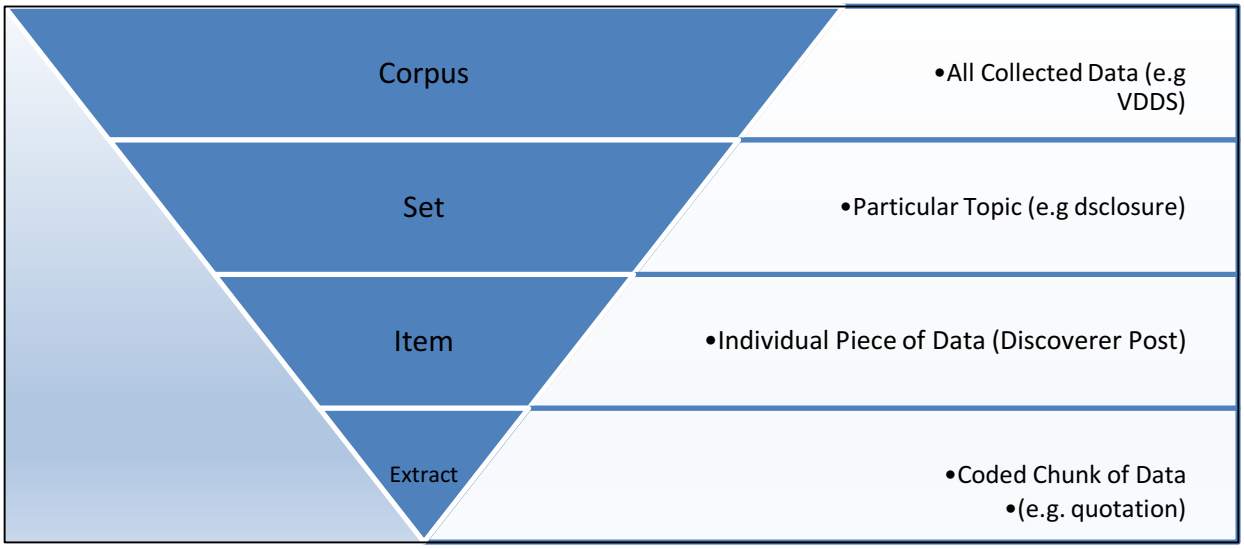


Figure 7 – Thematic Data Analysis Refinement (Source: Braun and Clarke, 2006, p. 79)

3.5.2.1 Third Party Interviews

Resulting from the keyword searches a considerable number (>50) of interviews were retrieved that were performed by journalists from popular blogs or news sites. These interviews provide evidence on the experiences, discourse and actions that researchers undertake when uncovering and disclosing vulnerabilities. Interview data-items provide context and evidence as the

participant of the interviews are subject matter experts, or significant actor within the VDDS.

3.5.2.2 Commercial Reports and Popular Media

A further source of evidence are commercial reports created and distributed by organisations who participated within the VDDS. Due to the nature of vulnerability discovery many commercial organisations have vested interest in producing marketing material the area of vulnerability and software security in general. As such commercial reports are a source of information on the discovery and disclosure processes, entities that are involved and potential dynamics that are present. Specifically reports on the frequency and prevalence of zero day (0day) vulnerabilities, and potential discovery techniques are described in detail. These types of commercial reports also contain interesting insights into organisational stances toward disclosures and the potential environmental factors. Furthermore, several commercial reports have undertaken investigative work to describe the black-market dynamics of vulnerability research and trading. Popular media news outlets provide commentary about software vulnerabilities and security in general, with varying degrees of accuracy and bias. Nevertheless, they are still practical sources of data that provide insight into the vulnerability discovery system.

3.5.2.3 Third Party Semi-Structured Interviews with Subject Matter Experts

During the main literature review several studies were identified as important, these are known as the Radianti studies. As the research was deemed valuable contact was made with Radianti and original primary information was obtained via email, and word document (Radianti, 2010a, 2010b, Radianti et al., 2006, 2007a, 2007b, 2009). This primary material is in the form of a structured survey and semi structured interviews whereby Radianti asks specific questions and requests clarifications from vulnerability researchers. The surveys provide detailed information on how the researcher entered both the black and grey markets and the movement between the two. Additionally, the surveys provide detailed data on the ethical considerations researchers take when selling a vulnerability.

3.5.2.4 Further Survey data

Many surveys were initiated by the author to provide an overview not readily available via normal open source research. To assist in the collection of the data an online questionnaire was created and requests for information were sent to:

- General call for respondents to complete the survey via twitter and linkedin.com
- Targeted call for respondents to complete the survey within community groups on linkedin.com
- Request to CPNI to circulate to Security Researchers Information Exchange (SRIE)
- Bespoke interview requests to Microsoft, VUPEN, Hackerone and Endgame Inc

Unfortunately, but not surprisingly, the response rate was low to the questionnaire and no responses were forthcoming from Microsoft, VUPEN, Hackerone and Endgame. The author presented to the CPIN SRIE which was greeted in a favourable manner, but yielded no survey responses. The Survey detail can be found in appendix A, and analysis found in chapter four.

3.6 Ethical approval

Due to the nature of this research several ethical considerations have been considered. These considerations include the storage or personal information, informed and recorded consent, ability for the participant to withdraw and risk associated with researching the areas of vulnerability discovery. Ethical approval for this research study was applied for and approved by the Science and Engineering Research Ethics Committee (SEREC) on the 19th Aug 2014, attracting the reference 116-2014. Ethical Approval documentation can be found Appendix A

3.7 Chapter Summary

Thus far this investigation has argued that the VDDS is a complex issue, requiring a new approach to characterise and investigate the dynamics of it. As such three techniques, have been selected to assist in the investigation, Thematic Analysis, Exploratory Data Analysis and System Dynamics. All techniques will be used within a mixed methods research framework, founded upon a exploratory phenomenological philosophy. Practically speaking, the mixed methods approach lends itself well to comparing differing perspectives which can be drawn from qualitative and quantitative datasets. For example, initial results from Alhazmi and Malaya (2005) were presented from a quantitative and statistical perspective providing an mathematical view on the observed behaviour (i.e., sigmoidal cumulative growth of vulnerabilities over time). However, when subsequent observations were made within later studies these initial observations were revised. It is therefore suggested that the use of triangulation methods inherent within the mixed method approach provide an erudite approach to understanding the VDDS and provides a richer, complete and coherent picture of observed phenomena.

To provide confidence in inductively constructed models, the accuracy of data under scrutiny, and the variables themselves is critical – particularly when a complex System Dynamics model is constructed. One key issue when we consider models that describe vulnerability discovery processes is that all data is observational in nature. We can observe events occurring and infer relationships between entities, but we cannot make absolute claims of precision, nor existence. Consequentially, the breadth of data that has been collected and analysed was drawn from a wide set of quantitative and qualitative sources. The main qualitative sources are from open sources, such as social media platforms, subject specific forums, academic studies and professional news pieces. These sources provide narrative and social discourse into what is taking place within the VDDS. For example, a longitudinal analysis of the attitudes toward vulnerability discovery and the evolution over time may yield important evidence around the rate of vulnerability creation and discovery.

Alongside this, several closed sources have been used for example personal interviews, forums and commercial reports.

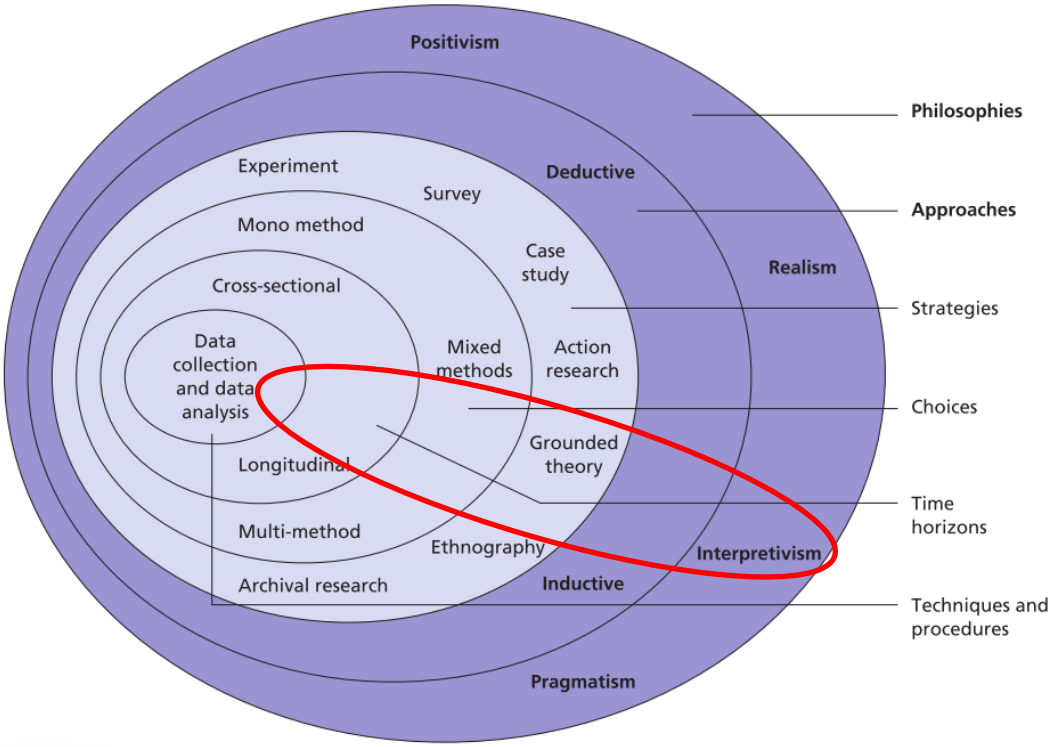


Figure 8 - The Research Onion Source: (Saunders et al., 2009, p.108)

Due to the nature of the research topic the datasets upon which conclusions are drawn is disparate and incomplete, hence abductive. Therefore, two specific data analysis frameworks are offered for use, EDA and Thematic analysis. These provide the tools to analyse both the qualitative and quantitative data that has been gathered. The data corpus is extremely varied, ranging from structured interviews and social media posts, through to empirically observed market share of operating systems. As with any research methodology, the mixed methods approach is not without its limitations and challenges to the researcher. Whist mixed methods provide a solid approach from both quantitative and qualitative perspective, this effectively doubles the data that is required. It also requires a level of fluency in quantitative and qualitative data collection and analysis. Moreover, this doubling of data and analysis is also time intensive, especially when we consider the fidelity that is required for the later system dynamic modelling stages. Indicated by the red oval is the collection of

world views, philosophies, methodologies, methods and approaches the research has adopted. In the next chapter I present the principle qualitative analysis of the collected datasets, items and coded extracts. The collected data was analysed using the techniques from within the Thematic Analysis framework. These Analytical procedures and results are described in detail within chapter four.

4 Qualitative Data Collection and Analysis

In the following pages I present the usage of the Thematic Analysis framework and techniques outlined in Braun and Clarke (2006). This is alongside supporting analysis used to draw out concepts and themes from collected data. An overview of collected data is provided along with analysis of the validity and credibility of the data sources. This is followed by an explanation of the codes generated inductively from the data and a codified thematic structure of the VDDS. Alongside the structure, a detailed description of actors that generate interactions and dynamics that accompany them are offered. Finally, a causal structure of influences is provided as a basis to form the basis for a theoretical model of vulnerability discovery and disclosure to be constructed.

4.1 VDDS Overview

The VDDS has been characterised as a complex yet, tractable problem, inhabited by entities that interact over time (Malone et al., 2013; Rahimi et al., 2013). Therefore, the first phase of constructing theory, and subsequent model of the VDDS is an exploration of the human and social aspects of the system. To do this Thematic Analysis was employed as the technique of choice within the qualitative stage of the mixed methods approach. The mixed method approach provides differing lenses to observe the system and triangulate between, thus providing further confidence in the analysis (Creswell, 2014, p.207). Once collected data was arranged and categorised into data sets, items and extracts and finally coded with any observed patterns being noted as a potential theme. Once noted, codes were analysed and drawn out from the data items, leading to the formation of a potential themes, relationships and interactions.

4.1.1 Structuring the VDDS

Data collected about the VDDS is both unstructured and heterogeneous in nature as it is predominantly derived from entities who do not conform to a specific standard (Katal et al., 2013). Hence, once data was collected organisation and structure was required to make sense of the data and provide

a framework to undertake analysis. The Thematic Analysis approach provides a method that provides structure to data by allocating, or coding, data items with descriptive tags, highlighting relationships between data items. This approach allows for the identification of sub-themes and themes to emerge from within the data (Braun and Wilkinson, 2003; Braun et al., 2006).

4.1.1.1 Data Item Descriptions

Social media and blog data items contain rich seams of information whereby vulnerabilities are discussed, debated and interactions between entities are described and defined. Specifically social media platforms are seen to be more technology focused such as Reddit, Hackernews and 4chan are used for this purpose (4chan, 2017; Reddit, 2017; Ycombinator, 2017). Open social media platforms such as these are a valuable source of data and to the discourse around vulnerability discovery and disclosure as they provide a historical record, and environment to discuss vulnerabilities. This is alongside the need to discuss current and new tools for discovery.

A further primary source of evidence is email, which has been used for several years. In the context of vulnerability discovery large email archives exists, specifically a well-respected email distribution list known as ‘Full Disclosure’ (Seclists.org, 2002). The full disclosure email distribution list is *the* place to disclosure security vulnerabilities to the community and wider public. The full disclosure list is a further valuable source of evidence to support or refute hypotheses stated about the discovery process and - importantly - the disclosure process.

Alongside social media, a further medium that is used to communicate between vulnerability researchers and other participants within the vulnerability discovery system is speciality web forums. These web forums can be publicly available for comments, or closed requiring authentication to view posts. Given the nature of the subject matter, it is probable that a lot of discussion around the trading, discovery and production of software vulnerabilities is within these closed fora. Moreover, given the proliferation of anonymization tools such as ToR or L2P a lot of the discussions that occur are unobtainable for analysis. However, open

forums such as Reddit, and other bespoke open forums provide insight into the market participants and their interactions.

A further source of evidence that was used in constructing theory was the usage of interview transcripts. Interviews with vulnerability researchers or those who are involved within the brokering, disclosing or purchasing vulnerabilities is an invaluable source of information about how the system behaves and is structured. The interviews that have been analysed in this section are semi-structured, and were undertaken by third parties, typically conducted by social media outlets, and subsequently published on the web.

A significant number (>1000) of reports and publications have been produced by commercial organisations, which provide opinion and insight on the vulnerability discovery system (Ablon et al., 2014; Frei, 2013a, 2013b; Lewis and Baker, 2013; Miller, 2007; O'Gorman, Gavin McDonald, 2012). Typically reports of this nature are marketing and sales orientated, produced to enhance the image of the company, and are not peer reviewed or academic in nature. Nevertheless, they do provide insight into the discovery and disclosure system as most of the content is provided via researchers who have links into the community or are vulnerability researchers themselves. Furthermore, the introduction of commercial third parties such as vulnerability brokers provide first hand insight into the VDDS. Therefore, the usage of commercial reports as a secondary source of information is valid, with special attention given to the credibility of the source and any biases present.

4.1.2 Overview of Collected Qualitative Datasets

Within this research the total data corpus consists of seven distinct types of qualitative and quantitative data. For example, within the social media dataset there are in total 364 data items, 1413 data extracts, from 1998 unique actors, complemented with 2,746 emails published by the full disclosure email archive. For the purposes of this research data items are defined as discrete social media posts (including comments), commercial reports, interviews, blogs, emails or news articles. All data items were exhaustively read, and re-read as to gain the most accurate meaning and context from the data.

The initial step within the analysis phase was to first choose which of the identified data sources to use, whilst being pragmatic on how much data can be collected and processed. Yin (1994, p.17) advocates collecting data from the widest range of sources, yet being practical, thus providing confidence in the validity of the collected data corpus and analysis undertaken. By embracing this concept of *pragmatic wideness*, the collected data was acquired from as wide and extensive range of sources, whilst being realistic on the availability of data and the time available to collect it. An overview of the data sources is provided in Table 9 below.

Data Source	**Description and Bias**
1.Blog posts from security community	**Vulnerability discovery is covered heavily by both security commentators and vulnerability researchers themselves. As such a large volume of data is available around the actors and relationship that are involved within discovery system.**
Typical Data Item Description Blog posts are generally considered a new phenomenon and are a product of the web 2.0 paradigm. They are published via personal or corporate mechanisms, and therefore can be in most cases considered a primary source of evidence, unless they are re-posting opinion from other sources. The blogs that have been selected are from known vulnerability research organisations or individuals. Where credibility cannot be independently confirmed, this is nonetheless considered as researchers may not be affiliated to a commercial organisation, as the blog poster may have considerable technical knowledge and experiences. Potential bias may include both marketing and sales pressures, feuding with rivals, posting for notoriety and untruthful posts. Validation of blog posts is difficult, unless peer review comments are post within the blog, this can provide a level of validation as to the authority of the author.	
2.Open source web forums and news groups	**Experiences around vulnerability discovery, and disclosure are regularly discussed on popular online webforums such as Reddit.com or 4chan.com. In addition specific hacker forums have been included within the evidence.**
Typical Data Item Description Webforums and their predecessor newsgroups are major sources of information about software vulnerability. They are generally used by technically literate users, and have a large user base. The popularity of newsgroups has declined over the past decade. Due large user base, the newer webforums can considered to be a primary source of information. The credibility of webforum posters is again variable, as posters may express an opinion rather than actually undertaking or being involved within a vulnerability discovery. Similarly the bias of posters can be heavily civil libertarian in nature. However, this type of post can be a good source of evidence.	
3.Interviews performed by third parties	**Interviews performed, and posted in blogs and webpages.**
Typical Data Item Description Third party interviews are Secondary sources of evidence and data. They are conducted as the interviewee has been validated as a person who is directly involved in vulnerability discovery, and has demonstrated their credibility.	

These interview are unstructured as they	
4.Email archives	**Archives of vulnerability discovery and disclosure are regularly posted to a open source distribution list known as 'full-disclosure mailing list' found at seclists.org. This is considered the defacto forum to disclose vulnerabilities.**
Typical Data Item Description Full disclosure email list is again a primary source of data both from a qualitative and quantitative basis. The email distribution list has thousands of subscribers, which allows security researchers to disclose to a large public distributions any vulnerabilities that they have discovered. Typically it is used as a quasi-official channel to disclose a vulnerability if all other avenues of resolution have been exhausted. The other way in which the distribution list is utilised is to 'go-pubic' without any prior discussion with software vendors or coordination organisations.	
5.Commercially produced vulnerability reports	**Vulnerability discovery and research is primarily undertaken via commercial organisations.**
Typical Data Item Description Due to the nature of cyber security and the monetarisation of vulnerability discovery a significant proportion of research that is undertaken is by commercial organisations. This research is generally undertaken as a marketing exercise and published to create hype or awareness of a current issue or trend. Ultimately this type of research is based upon real world observations, however must be considered in the context of the organisation that it is published by.	
6Third party structured interviews	**Several structured interviews have been used that were sourced directly from (Radianti, 2010b). Furthermore these interviews are primary sources as the interviewee is a vulnerability discovery and disclosure actor.**
Typical Data Item Description Interviews that were collected are structured in natures, and consist of email questions and responses between Radanti and anonymous respondents. Each data item is in the form of email, and was provided by Radianti during March 2015, but cited in studies from 2008 – 2010.	
7.Third party Semi-structured interviews	**Several interview transcripts were collected. These took the form of magazine articles, blog posts or videos. All interview are scripted, and are considered to be popular in nature.**
Typical Data Item Description Interviews with vulnerabilities researchers and brokers are undertaken by popular bloggers or specialist new outlets. These articles provide insight in to how the vulnerability discovery and disclosure system operates from different actor perspectives. Generally, these articles are between 200-500 words and consist of experiences within the system.	

Table 9 – Qualitative Data Items Descriptive Overview

4.2 Initial Analysis Steps

The first an arguably most important steps of thematic analysis is to methodically read, and re-read the data items and code each item as demonstrated in tables 10 -15 (Braun et al., 2006; Ryan et al., 2003). This research has adopted an abductive approach to the investigation of the VDDS, which is in turn strongly links conclusions to the data. All data was collected was coded over a 6-month period, and initially consisted 254 codes, which was sorted, arranged and categorised into a final list of 93 codes which consist of 'holding codes' that describe a collection of codes, and actual code holding data extracts (Saldana, 2016, p.9). Codes were sorted, resorted and categorised when new data was added. This allows a level of flexibility within the categorisation and structure of the codes, whilst retaining the codes themselves. During the coding and reading phase initial comments or observations were noted, any interesting actors or obvious systemic structures. These were noted within the software application *Nvivo*, and 'white boarded' to aid in the construction of the emergent themes. Due to the exploratory nature of the research the structure of the codes initially took the form of a list, which was then transformed into a hierarchy of categories providing arrangement that made logical sense. However, codes that were observed, whilst arranged into discrete groups did not provide a rich analytic picture of interactions across actors and information flows.

4.2.1 Coded VDDS Examples

Once collected data was processed and entered the Nvivo analysis package. Drawn from these thematic cases a parallel activity of constructing a high-level overview of the data items, noting any observed structures, information flows and meta codes that were generated from within the analysis. For clarity, codes are defined as:

"Codes identify a feature of the data (semantic content or latent) that appears interesting"

(Braun et al., 2006)

Examples of codes, and the cases that they were drawn from is presented in Tables ten through fifteen. Each presented example is extracted from the main data corpus and provides an exemplar of generated codes that emerged from the data. Alongside codes, analysis of the data item is provided to add context to the example and any bias apparent within the source. Finally, any observed structures or interactions that are present between entities is also provided.

Identified interactions are categorised in one of three ways. For example, the interaction between the vulnerability discover and the online vulnerability community. Whilst this may not seem a strict one to one relationship, the community has a indirect influence on the disclosure stance taken when a discovery takes place. Hence interactions within the VDDS attract the following definitions:

- **Mutual relation** - Direct communication between actors occurs via electronic of physical means, can be unidirectional or bidirectional (e.g. selling of a vulnerability to a broker, soliciting advice from community or ethically disclosing to a software vendor)
- **Contextual** – Indirect communication between entities (e.g. awareness of other actor's actions and experiences)
- **Environmental** – The actors are aware of indirect or direct influences on their behaviour (e.g. regulation or law enforcement activities)

Each interaction has been chosen to demonstrate the potential influence or impact the interaction may have. Each interaction is in almost all cases a bi-directional transaction between two or more entities within the VDDS. These cases are based upon the coding cards outlined by Massie (2015, p.66).

Data Item: BLOG1 – A thirst for regulation?	Open Codes
Analysis The post was posted on slate.com website, and was created on the 16/1/2013. The post is written by Ryan Gallagher and is aiming to explain the actors and relationships within the marketplace for vulnerabilities. The Emphasis is centred upon altering that this market is unregulated, and around the repercussions of this. Actors within the blog post are generally named as brokers (ZDI, Tippingpoint etc) with the occasional reference to bug bounties. One key point is around regulation set out under the wassenaar arrangement limiting the trade of 0day exploit software.	• Cyber Criminal Usage • Markets (Bug Bounty) • Markets (Broker) • Markets (Black) • Researcher (age) • Markets (Broker) • Usage (State Sponsored)
Observed Structures and Interaction Flows Mutual Relation - discoverer interaction with 3rd party brokers (bidirectional) Contextual – awareness of international regulation	
Reference: (Gallagher, 2013)	

Table 10 – Coded Example Blog

Data Item SOC1: An ethical dilemma, with negative experiences	Open Codes
The post this post exists on the popular reddit.com website that is used to discuss news, entertainment and general topics and was posted on 20/1/2011 in the /r/netsec/ forum there are 83 replies. The post is by member ***"i_didnt_do_that"*** and is seeking to understand how to disclose a serious vulnerability within a social networking site and what the issues are. The replies and discussion that follows centred around experiences in disclosing vulnerabilities to vendors, with most being posts being negative in nature. A further area that is discussed is the usage of the differing disclosure strategies (full and responsible) being used to 'shame' vendors into fixing vulnerabilities. A number of posts cite ethical dilemmas on what to do when a vulnerability is discovered. It is suggested that *"it only takes being burned once to make corporate higher-ups want it to never happen again"* (I_didnt_do_that, 2011) The users repeatedly mention that the users should suffer not from a vulnerability that wasn't their fault, and if customers knew the company maybe shamed into fixing things.	• Motivation (altruism) • Interaction with vendor (processes) • Feelings (uncertainty) • Disclosure Stance (full) • Disclosure Stance (Coordinated)
Observed Structures and Interaction Flows Contextual – Research discussion with community (bidirectional) Contextual – Ethical consideration of disclosure strategies (unidirectional)	

Environmental – Sentiment towards software originators drawn from social media (unidirectional)
Reference :(I_didnt_do_that, 2011)

Table 11 – Coded Example Social Media

Case COM1: Symantec – The Elderwood Project	**Open Codes**
This report was published by Symantec, well known global security organisation in Sept 2012. It is a technical marketing document, that's outlines the usage of 0day vulnerabilities by cyber criminals within malware. This was one of the first documented usage of malware using 0day exploits in both an organised and repeated manner. Alongside this, it showed that that potential transactions are taking place as different malware samples were analysed what we're using the same vulnerabilities but for very difference purposes. This could indicate a level of organisation and communication between vulnerabilities researchers.	• Discovery Techniques • Cyber Criminal Usage • 0day vulnerability usage
Credibility, validity and bias As this is clearly a report that was created by a corporate security organisation interested in selling its products, one must keep this in mind when reading it. However, the researchers that wrote the main analysis section of the report seem to be credible, as offer technical insight into the usage or 0days within malware. Therefore this analysis could be used in conjunction with the other reports to estimate the structure of the cybercriminal vulnerability involvement.	
Observed Structures and Interaction Flows Contextual – Research discussion with community (bidirectional) Mutual Relation - discoverer interaction with cyber criminals (bidirectional)	
Reference (O'Gorman & McDonald, 2012)	

Table 12 – Coded Example Commercial Report

Case EMAIL1: Finding Vulnerabilities?	Open Codes
This is an email trail from the distribution list 'Vulnerability Development', managed by seclists.org. The initial email was sent on the 5/5/2002, by a user known as kaipower@subdimension.com and there are 20 replies. The user is looking to establish known techniques for finding vulnerabilities within software. Long and comprehensive replies from the community are given ranging from the types of general tools, techniques and the level of education and experience required. In general the replies are position and helpful and suggest a way forward in a nurturing and community based manner.	• Vulnerability Discovery Tools • Discovery (techniques) • Researcher (experience and skills) • Tools (fuzzing) • Market Penetration • Researcher (Kudos) • Motivation (altruism)
Observed Structures and Interaction Flows Contextual – Experience and skills of discovery process (unidirectional) Contextual – Tools to discovery vulnerabilities (bidirectional) Contextual – community pressure for discovery and disclosure practices	
Reference **:** (Kaipower, 2002)	

Table 13 – Coded Example Email

Case INT1: Benjamin Kunz Mejri Interview Softpedia	Open Codes
This interview was published on the website "vulnerability magazine" on the 21/10/2011. It is centred on the issues that are around the security industry, specifically vulnerability disclosure. The interviewee explains the methods and targets for vulnerability discovery, the rationale for doing so and why they release the data publically. Two major theme throughout this are the altruistic nature of what they do and the interactions with both other researchers and software vendors.	• Emotion (Anger) • Emotion (Love) • Motivation (altruism) • Disclosure Stance (full) • Disclosure Stance (Coordinated)
Credibility, validity and bias The interviewee is the founder of a major vulnerability discovery organisation and has been researching vulnerabilities with his teams for around 5 years, making the interviewee a credible source. The validity of his claims are also of high value due to the nature of experiences. However, he can considered to be slightly bias due to the altruistic nature of his work as they give any all findings for free.	
Observed Structures and Interaction Flows Mutual Relation - discoverer interaction with 3rd Party broker platform (bidirectional) Mutual Relation - discoverer interaction with software originator (bidirectional) Mutual Relation - discoverer interaction authorities (bidirectional	

Reference: (Eduard Kovacs, 2011)

Table 14 – Coded Example Unstructured Interview

Case STR1: Interview on Vulnerability Markets: A white hat perspective	**Open Codes**
The interview was conducted by a researcher in (see Radianti, 2010) Norway in June/July 2009, discussing anonymously the emergence and usage of legal and black markets for vulnerabilities. The interview looks at the reasons as to why researcher use legal markets and why black markets could be attractive. The research suggests that the biggest motivation is money, and that generally researchers are very ethical in nature. Usage if ZDI and Tippingpoint are cited as models for legitimate marketplaces. The discussion then moves on to the lifetime of a vulnerability, which is estimated to be from 1 week to potentially years. This is extended to discuss the fumbling of the MS08-067 Microsoft vulnerability, The interviewee goes on to discuss the time to discover vulnerabilities within software, the conclusion is that it is a highly variable task and is dependent upon software quality. Finally, the interviewee discusses disclosure polies and suggests that full disclosure should only be used 'as a last resort'.	• Disclosure Stance (Coordinated) • Motivation (Money) • 0day vulnerability usage • Markets (Broker)
Observed Structures and Interaction Flows Mutual Relation - discoverer interaction with 3rd Party broker platform (bidirectional)	
(Radianti, 2010a, 2010b, Radianti et al., 2006, 2007a, 2009)	

Table 15 – Coded Example Structured Interview

4.3 VDDS Code Hierarchy

Codes generated inductively from within the datasets were arranged to reflect the prominence, and perceived importance of the codes. Specifically, code instances were tallied to establish the frequency of occurrence. This in turn was used to construct a coding hierarchy, forming the initial exploration of potential themes and structure within the VDDS. Each level indicates a higher level of abstraction, providing a grouping, of clumping of codes surrounding a potential theme. From the social media dataset the following high level aspects were derived from the data:

Vendors (software originators)

- Disclosure of vulnerabilities, and the emotions and behaviours surrounding the disclosure such as fear, anger, hate and confusion pervade this grouping. Within the disclosure grouping is also the concept of time and the quality and experiences of interaction and disclosure of a vulnerability, this group alone accounts for **51.17%** of all coded data extracts.
- Software Quality, is a major grouping of codes, with the sub-codes of standards and education featuring within in the grouping. Aspects that are included within the group as the education level of the software vulnerability discoverer and education of software originators to understand the VDDS.
- Interactions, communication within the VDDS, and between entities is a code, accounting for **1.56%** of all coded data extracts.

Market Forces

- Including codes related to black markets, vulnerability brokers, bug bounties, auctions and the opinions surrounding these markets is very prevalent accounting for **30.15%** of all coded data extracts.

- Motivation the aspect of recognition and altruistic kudos is very acute within the VDDS alongside the reward via monetarily recompense accounts for **6.58%.**
- Discovery Process codes include the tools that are used in the discovery process and security software development practices that are used, accounting for **1.06% of** all coded data extracts.
- Community included ages, skills and leadership within the VDDS community showing up in **0.14%** of data extracts.

Remaining coded extracts making up 9.34% account for lower level codes. The coded map outlined in Figure 9 shows a complex set of interactions between entities characterised by both the movement of information flowing around the VDDS and resources (money, vulnerabilities, tools etc). Within Figure 9, the themes are highlighted as yellow rectangles, with sub themes highlighted in blue circles. Finally, codes and sub codes are represented by pink diamonds. Links and highlighted relationships derived from the data are shown as lines between themes and codes. A list of raw codes is provided in Appendix A.

4.3.1 Survey Responses

Within the investigation a survey was conducted to illicit clarifications and statements from the VDDS community. Ten responses were received, which add to the contextual environment of the VDDS, details of survey questions are available in Appendix A. One particularly interesting response was received around educational level, with 8 out of 10 responses citing Masters level or above education, with 1 N/a and 1 undergraduate. This suggested that the level of education that vulnerability discoverers attain is significantly above general. Additionally, the length of time a vulnerability discoverer has been involved in searching for vulnerabilities is 5 years or less, with 5 responses out of 7 (3 did not answer). Again, suggesting a relatively short lifespan. Furthermore, all discoverers worked within a team, with 7 out of 7 positive responses.

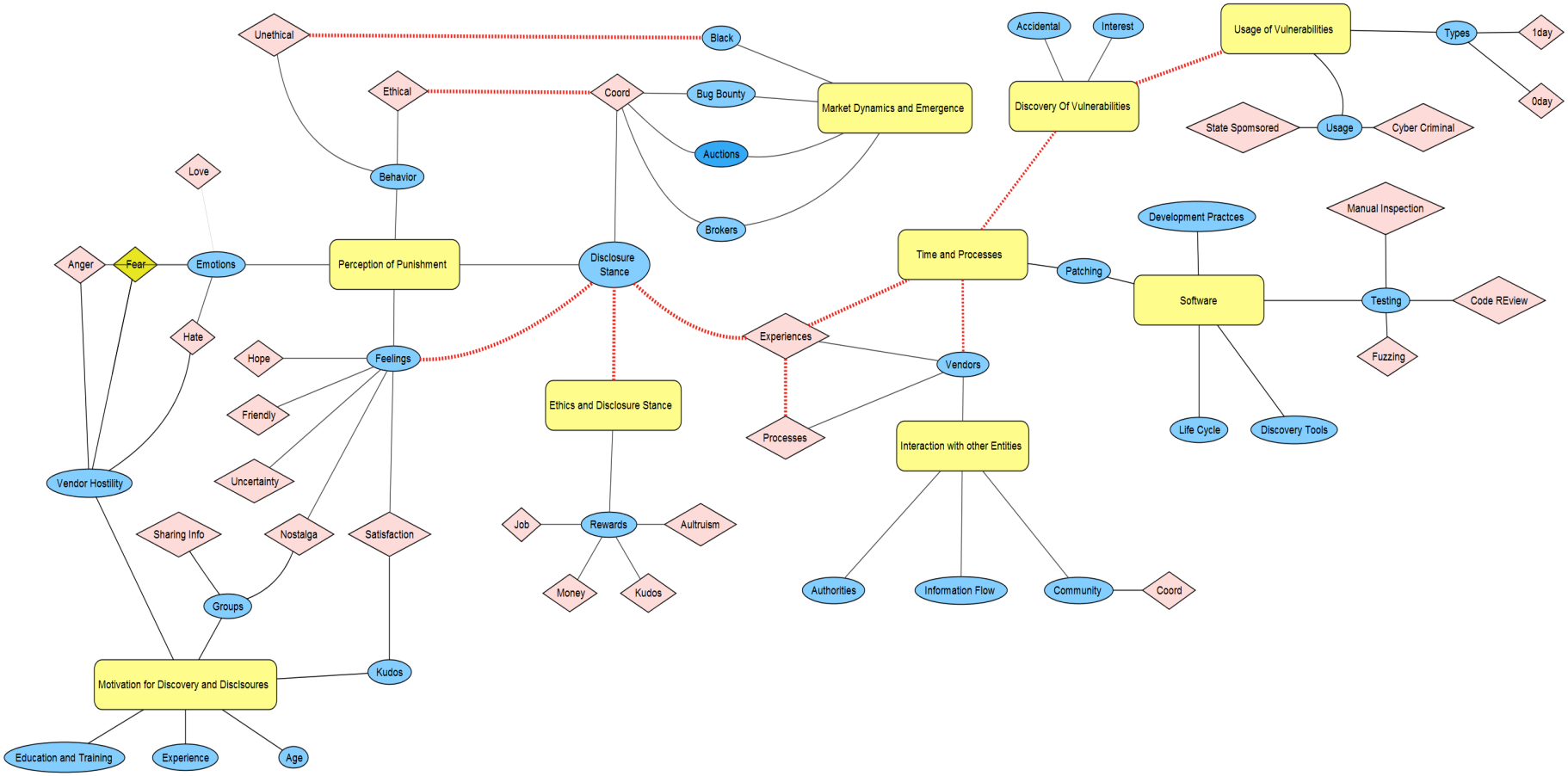


Figure 9 – Initial Inductively Generated Coding Hierarchy

Evolving (by re-reading) and categorising the coding structure yielded interesting and previously uncovered relationships and interactions. Key relationships that were not evident during the initial coding activities, but were apparent during the subsequent analysis steps, were centred around the following factors:

- **Relationships between disclosure stance and rewards.** Initially the relationship was considered an interaction between discoverer and software originator, however there is an influencing factor known as 'monetary reward' that influences the disclosure stance.
- **Time delays and Length of Interactions.** Delays between both vulnerability discoverer and software originator are acute. Significant delays exist between the time that a vulnerability is discovered and the public disclosure of the vulnerability. Normally interactions are time bound by disclosure policies.
- **Experiences** disclosing vulnerabilities to software vendors.
- **Emotions** felt towards the software vendor and the strong tension between fear and ethics when disclosing.
- **The rational choices discoverers make,** most if not all discoverers make rational and information choices about what the wish to do with the vulnerability details.

Once notable aspect is centred around rewards. The reward that a vulnerability discoverer may gain by selling the vulnerability, therefore potentially changing their disclosure stance is of interest. For example, the recent popularisation of so called bug bounties (Hackerone, 2016) is a manifestation of economic incentive scheme to reward researchers for finding vulnerabilities. Furthermore, most of the full disclosure events have been driven by the actual or perceived hostility to discoverers who discloses the vulnerability by the software vendor. This change in stance by the discoverer suggests that they may be more inclined to adopt a coordinated disclosure stance, or sell the vulnerability illegitimately. Finally, a significant area of concern is the ethical dilemmas which

discoverers discuss and reflect upon when deciding which disclosure strategy to adopt.

4.4 Identification and Structuring Candidate Themes

Having set out the low-level codes and constructed a coding hierarchy, I will now turn to the consolidation and identification of themes. There are many techniques which may be used to identify themes derived from codes (Ryan et al., 2003; Strauss, Aselem. Corbin, 1997, p.36). However, the choice is reduced when we take into consideration the inductive and explorative approach of this research. Therefore the analytical techniques used within this research are outlined by Ryan and Bernard (2003), who advocate analytical scrutiny around repetition, indigenous categories, metaphor/analogies, social conflict, status, control and linguistic connectors, although not all are used. Ryan and Bernard (2003) also suggest several techniques to process data items and extracts into themes and categories.

Ryan and Bernard (2003) outline a number of techniques that assist in the processing of low level code, and the production of higher level themes and sub-themes. The criteria used in selecting which analytical techniques to use is, argues Ryan dependent upon the type of data collected (Ryan et al., 2003). Suggested techniques range from identification of repetitions, transitions, similarities and differences, word frequency list and Keyword in Context, linguistic connectors. For a more detailed explanation of all techniques see (Ryan et al., 2003).

The chosen techniques that are most appropriate for the VDDS are Keyword in Context, frequency lists and linguistic connectors. Frequency lists are simple to implement, and provide signposts toward potential themes due to the duplication of words, indicated by an increase in frequency. KWIC was chosen as it provides views of potentially complex and rich insight into linguistically orientated data, and is simple to implement (Culy and Lyding, 2010). Finally, linguistic connectors, suggest causal or conditional relationships between things, or groups of things.

These three techniques were applied to Social media data items, email and interviews. Assisted by the Nvivo v10 software package Nvivo provided the automation of a number of these activities, such as keyword frequency lists and keywords in context processing.

4.4.1 Keyword Frequency List of VDDS Corpus

Ryan and Bernard (2003) state that the first step in exploring data is to construct a word frequency list. Therefore, a such a list was generated across the complete data corpus, essentially performing a corpus wide pre-processed map and structure. The map included all common English words used within normal everyday language such as ‘and’, ‘the’ and ‘this’ - these words were excluded from the analysis temporarily. Further pre-processing was undertaken, which resulted in a further word frequency list. The generated list was of low quality and included several words that required removing from the data, such as ‘www’ and ‘permalink’, these were also added to a stop list within Nvivo as they indicate metadata collected via the raw collection process. Once these pre-processing activities had been undertaken, data was in a form appropriate for analysis. Words were ordered based on rank. Emerging from the frequency list were keywords, such as ‘days’ (No.4), ‘information’ (No.5) and ‘Market’ (No.7) thus indicating a frequency baseline from the entire corpus.

To provide further insight a second frequency analysis was constructed from coded data items, rather than the whole corpus, which provided a focused comparison. The subsequent list was of improved quality, and provided additional insight as it showed a similar number of commonly associated VDDS words such as ‘vulnerability’, ‘security’ and ‘vulnerabilities’ alongside English common joining words.

A final pre-processing stage was performed which yielded many nouns and adjectives such as ‘disclosure’ (No.2), ‘software’ (No.3), and ‘fix’ (No.4). Several unusual words featured highly within the coded data extracts, potentially suggesting a higher interest in these topics. An unusually high ranking word is ‘time’ (No.9), which is not a typical word associated with security and vulnerabilities. Not included within the top 10 list is are words such as ‘people’

(No.17), ‘because’ (No.20), and ‘responsible’ (No.32) due to space constraints, as indicated in Table 16 below. Within the analysis, words are drawn out in terms of uncommon or frequent use within vulnerability discourse. This word frequency list provided an initial point to perform further analytical techniques, again outlined by Ryan and Bernard (2003).

Whilst useful, keyword frequency lists do not provide strict comparative results, as to how different or important identified keywords are. Consequently, to indicate how the generated frequency lists compare to none vulnerability related discourse word frequencies were generated from two well-known general word frequency lists from Brown and Wordlex. Worldlex, consists of 104.2M analysed words from Twitter and blog posts) and the Brown University Standard Corpus of Present-Day American English with 1.01M words (Francis, 1965; Gimenes and New, 2016). The results of both comparisons with Wordlex and Brown are provided in Table 16 below.

All Sources					**Coded Data Items**				
Corpus Rank	**Brown Rank**	**Wordlex**	**Word**	**Word Count**	**Corpus Rank**	**Brown Rank**	**Wordlex**	**Word**	**Word Count**
7	1187	1205	**security**	7703	6	1187	1205	**security**	316
11	11038	11550	**Vulnerability**	6248	13	11038	11550	**vulnerability**	252
16	N/A	34997	**Vulnerabilities**	4241	14	N/A	34997	**vulnerabilities**	183
18	247	196	**Days**	3914	22	22168	7693	**Disclosure**	155
20	363	481	**Information**	3363	24	N/A	2828	**Software**	166
21	N/A	2828	**Software**	3661	28	363	481	**information**	155
28	660	663	**Market**	2786	29	46721	9350	**Vendor**	152
34	22168	7693	**Disclosure**	2507	31	26722	6769	**Vendors**	147
43	6687	1036	Computer	2310	**33**	NA	4345	bug	146
46	16983	11135	Bounty	2286	**35**	8697	10614	exploit	140

Table 16 – Keyword Frequency Ranking

Within Table 16 we can see that words such as ‘Disclosure’, ‘Market’ and ‘Bounty’ score highly, with these words indicating potential themes. Alongside this words such as ‘Vendor’, ‘Days’ and Disclosure’ also suggest that these words are important within the VDDS discourse, and they may suggest themes.

4.4.2 Keyword in Context Analysis

Having defined and measured the frequency of occurrence within the data corpus, and coded data items, identified words were used within the next phase of analysis, the context of the most frequently identified words. The usage of keywords to derive contextual meaning from disparate or unrelated data has been used within information science for over 40 years (Parnas, 1972). Keyword in Context (KWIC) is a technique used within social sciences to give meaning and context to identified frequent words within datasets (Ryan et al., 2003). Using the identified words from section 4.4.1 all ranked words were used to establish context that surround that word. This was undertaken by utilising both the whole of the data corpus as the search space, and specific coded examples. In the case of this research +/- 5 words on either side were used to derive a meaning or establish context.

As expected within the coded data items the keyword ‘vulnerability’ is frequently seen, therefore warranted further investigation. The keyword ‘vulnerability’ is typically centred around three aspects; the discovery of a singular vulnerability, technical discussion of vulnerability characteristics or what the subsequent actions once found are. This keyword is a potential anchor for further analysis, as exemplified by a data excerpt from a VDDS participant showing both the anchor keyword ‘vulnerability’ and the context of discovery, type of vulnerability, location, interactions and entities.

“Back in July, I searched for and discovered a cross-site scripting ***vulnerability*** *on facebook.com, as well as what I would describe as infrastructure flaws that give me the ability to steal httpOnly authentication cookies used by Facebook. So far I have kept all the details to myself”.* (Facebookxss, 2010)

Furthermore, keywords such as ‘disclosure’ and ‘information’ were used together to potentially indicate overlaps and relationships between keywords. Once identified contexts were sorted and named, via a technique known as ‘cutting and sorting’ and noted within Nvivo. The technique is well documented as an analytical method that allows structure to be applied to processed data, albeit in discipline other than computer science. (Barkin et al., 1999; Braun et al., 2006). Using the most commonly identified words from table 16 above, a KWIC ‘cut and sort’ was produced that identified a number of relevant contexts. These contexts indicate current zeitgeist from within the VDDS, as they are tightly coupled with the collected data items produced by participants. For example, KWIC keyword ‘market’ is surrounded by the context of discoverers use of the black or illegitimate market, speculation of how prices of vulnerabilities within both legitimate and illegitimate markets have increased and the differing ethical considerations when selling details of a vulnerability. Identified contexts are presented in Table 17 below, and describe a rich set of contexts that exist within VDDS.

According to Tecsh (2013) the usage of KWIC and frequency analysis allows for the condensation and distillation of interpretation of data (Tesch, 2013, p.139). This condensation provides a level of abstraction to carry forward, and look for linkages between contexts, drawn from the KWIC analysis. For example, looking at Figure 10 below a textual overlap between vulnerabilities and information is present. Similarly, there is overlap in context between disclosure and markets. These overlaps suggest that a relationship or interaction between concepts, and the entities that cause the interactions to occur. The derivation of Figure 10 from KWIC analysis provides a signpost and further evidence for the relationships within the VDDS.

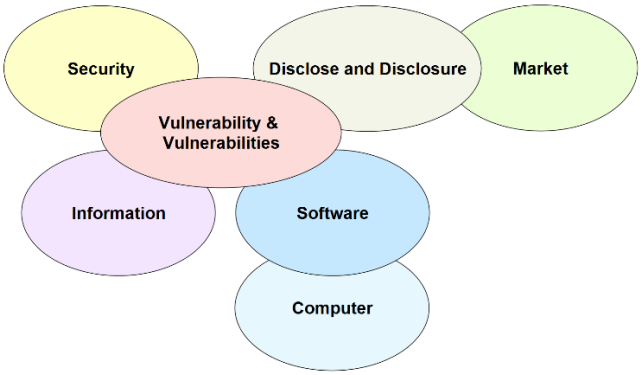


Figure 10 - Keyword in Context Overlaps.

KWIC Keyword	Identified Contexts
Security	• Individual Researchers and soliciting advice from security community. • Security of software and products created by vendors.
Vulnerability and Vulnerabilities	• Discovery of vulnerability and reporting process. • Issues reporting vulnerabilities to vendor. • Moral issues surrounding disclosure of vulnerabilities. • Vendor process issues and communication. • Differences between disclosure experience between vendors. • Bug bounties and reward schemes.
Information	• How to share information between entities. • Dissemination of potentially sensitive vulnerability information that could cause harm • Usage of information that is used to exploit systems. • Ethics and responsibility around releasing information about vulnerabilities within software.
Software	• Vendor testing of software. • Quality of software once released. • Tools and techniques for discovery of vulnerabilities within software. • Software as an entity, constantly evolving.
Market	• Black market maturity and inevitability. • Vendor negative attitudes pushing researchers toward the black market. • Perceived size of black markets. • Curiosity about the market for vulnerabilities. • Confusion on the ethics on what to do with vulnerabilities (Sell/Fully Disclose/Responsible) • Different market models (auction/bounty/trade) • Discussion of market forces of supply and demand. • Implications of a open market for vulnerabilities. • The failure of software vendors creating good quality software.

Disclose & Disclosure	• Different strategies for disclosure. • Forcing software vendors to act when disclosing. • Logical progression from coordinated disclosure to full disclosure due to vendor actions. • Punishment of users not software vendors. • Choosing the 'best' strategy for disclosure. • Usage of policy and timelines to ensure vulnerability is fixed.
Computer	• Entry into (computer) security field by researchers.

Table 17 – Keyword in Context Analysis with examples

4.5 Linguistic Connectors

The final technique used to process data is linguistic connectors, which are a method to indicate causal relationships, identify connections between entities and nature of their interactions (Ryan et al., 2003). Words and phrases such as 'because', 'since' and 'as a result' often indicate causal relationships, whereas time-orientated connexions may be conveyed with words such as 'before', 'after' or 'next' (Ryan et al., 2003). Casagrande and Hale (1967) provide a useful list of semantic indicators between words that have a relationship or entities that are related. Relationships such as attributive ones, where an entity is defined in respect to another entity and functional relationships whereby an entity affects another are of particular relevance to software vulnerability discovery and disclosure (Casagrande et al., 1967). Two pertinent examples of the utility when using linguistic connectors is the usage of the causal keyword 'because' within two relevant examples. The first is an interview with Benjamin Kunz (Eduard Kovacs, 2011) and the second, a social media post using 'since' when discussing a vulnerability within the Nissan Leaf electrically powered car.

Within the Kunz interview a statement is made around the software originators stance toward software quality and vulnerability discoverers. Kunz uses the keyword *because* twice to signify the strong causality between the software originator, and discoverer as a dual relationship. This duality is evident in the fact that Kunz characterises the connection as love/hate relationship as the originator does not wish to see their own flaws, yet when flaws are uncovered they are reluctantly indebted to the discoverer. The preceding words, 'hate' and 'us' signifies a group of people, and a very strong emotional experience that has

clearly impacted Kunz previously. This is typical of the type of language that is used when software discoverers discuss the process of disclosure.

“There are 2 options for the product vendor ... he hates us ***because*** he can not [sic] see his own flaws/mistakes/fails ... or he loves us ***because*** he can now see his flaws/mistakes/fails” (Eduard Kovacs, 2011). [emphasis added by author]

In the Nissan example, a social media contributor Mikestew, is discussing a vulnerability that has been uncovered within the Nissan Leaf electric car (Nissan, 2017). Mikestew, suggests that the rational for locking down (i.e. ensuring there can be no inspection of the code) is due to the poor state of the software. This causal relationship between obfuscation and poor code, is evident within the final part of the post, whereby Mikestew ends with a proposition that legal action may occur if vulnerabilities are uncovered within the Nissan Leaf software. This example emphasises the interaction between the quality of software, and the potential for punitive action that may take place.

“Car companies are terrible at making software. I'm not surprised that they want to lock down the ECU code, ***since*** it's probably awful and full of subtle bugs as well, and they don't want to get hit with tons of lawsuits when people find a bug in the firmware” (Mikestew, 2016).

The power of extracting linguistic connectors from within rich narrative data is self-evident when we look at the above examples. The subtlety of a single word, *since*, changes the dynamic and meaning of the discussion as it joins a potentially positive statement to a negative sentiment - ‘full of bugs’

4.5.1 Time Influences

Time is a crucial part of the VDDS that has been identified throughout both coding and frequency analysis. Time is suggested to influence all the entities within the VDDS and importantly influence the outcomes that are produced from interaction within the system. An example of the impact of time on a specific outcome is the stance which a vulnerability discoverer may take once a vulnerability has been uncovered. The influencing factors derived from the data suggest that both the disclosure stance and adopted policy for disclosure are

influenced by both the quality of interaction between discoverer and software originator. Crucially, the time it takes to perform actions such as recognition, payment or patching are as impactful. Moreover, this seemingly central role of time evidenced in recent policy decisions by large software and technology organisations that within which a strict time period is adhered to once a vulnerability is discovered and importantly reported (Apple, 2017; Google, 2017; Microsoft, 2017; Oracle, 2017).

When considering the role of time within the VDDS, it is also necessary to consider the impact of the perception of time passing, rather than the absolute passage of time. The perception of time passing within the VDDS, is influenced by the way in which vulnerability discoverers, software originators or other third parties are treated. If treated badly, or communication is stunted or poorly managed the passage of time is perceived to be slow, and resumption of a possibly harmful disclosure stance may resume. If the disclosure process occurs smoothly with minimal delays, or if delays occur and they are communicated and explained to discoverers then perception of time moves quickly, thus resulting in a more coordinated disclosure.

A major source of information of both the quality of the interactions between VDDS entities and the passage of time between specific interactions is the email distribution list, full disclosure. Full disclosure is a public email distribution list that has been in existence for over 15 years, and originally provided for the use of vulnerability discoverers who felt (either philosophically, or otherwise) the need to make vulnerability details public without consultation. However, the role of the full disclosure email list has now joined with an information dissemination channel, with the full disclosure of vulnerability details. An interesting addition to the full disclosure email list was a standardisation of, how to disclose vulnerability detail, as opposed to why. This is important when discussing the wider VDDS and influences upon the sentiment of software originators as it provides valuable information on the process of disclosure that discoverers take part in.

4.5.2 Section summary

Whilst methodological in nature, this section outlines the process and techniques of how themes were derived from collected data. The next section sets out the process that were followed to extract meaning from codes, frequency analysis, KWIC and candidate themes. Coding data and extracting meaningful insights from the data corpus, whilst invaluable, is the first stage of organising the data, to go further we must seek linkages, meaning and themes from within the data. Using the differentiating factors outlined by Ryan and Bernard (2003) to test the validity of identified themes we can build upon identified codes, and increase the confidence in any themes emerging from the data. Factors such as repetition, metaphor and analogy, linguistic connectors and syntax are again suggested to be useful in uncovering both higher abstractive codes and themes (Ryan et al., 2003). As stated in chapter three the inductive coding paradigm was adopted, whereby the data analyst reads and re-reads all data items, and codes emerge from the data and assigned as appropriate. During the coding process data is not differentiated by source of data or topic as not to bias subsequent findings, as the generation of cross cutting phenomenological themes is the primary focus of the analysis (Ryan et al., 2003). Using the coded data extracts, and contextual information about the themes we can see that several casual relationships exist, which are outlined and explored in the next section.

4.6 Analysis of Candidate Themes

Building upon the analysis derived from section 4.5, the focus now moves to a refinement stage for candidate themes. Refinement of themes provides both confidence and validity, and is based upon the judgement on the part of the investigator (Ryan et al., 2003). As such further synthesis and mapping was undertaken, providing a level of abstraction and consolidation of the identified themes. This consolidation is in part to assist in the mapping of themes and sub themes, so theory constructed using identified themes are useful and intuitive. Consequently, the consolidated candidate themes are presented, along with sub-themes and narrative.

4.6.1 Thematic map of the VDDS

As we are looking for how the VDDS behaves, specifically structures and interactions within the system, combining codes into similar groups or patterns is an important step (Murtaza et al., 2016). Using the nomenclature outlined in (Braun et al., 2006) an initial thematic map was created by grouping codes together, utilising contextual information gathered by KWIC analysis, frequency rank and identified linguistic connectors. In its unrefined form, the thematic map, whilst comprehensive does not highlight the central tenets of the VDDS. Therefore, integration of concepts and refinement is required to draw out the key themes through a process of consolidation and evidential review (Braun et al., 2006). This refinement process provides a supplementary step to collapse and combine the initially identified theme and sub-themes, such as 'emotions', 'discovery tools' or 'behaviours'. These themes collapse into a collection of sub themes and major-themes. The resultant thematic map allows for a sense of significance within the analysis to develop, and pragmatic sense checking to take place. For example, two identified candidate themes 'software' and 'discovery of vulnerabilities' feature less prominently within the analysis, than the ethical or moral decisions discoverers take when a vulnerability is discovered. Conversely, the importance of the interaction between the vulnerability discoverer and the software vendor who owns the software is extremely important, both in a positive and negative way. Therefore, ranking and consolidating themes was undertaken.

4.6.2 Refined and Developed Thematic Map

Braun and Clarke (2006) advocate two levels of review when considering refined and developed thematic maps. This review takes the form of returning back to the level of initial codes, and secondly at the level of the entire dataset (Braun et al., 2006). Thus, the developed thematic map contains reviewed, consolidated and analysed themes which after both returning to the raw data provide and using high analysis provide a clear internal consistency. By re-reading the data codes, and identifying clear distinctions we can reduce and

promote candidate themes, and add further context from codes and data items listed in Figure 9.

These consolidatory process steps provide a clearer overview of the key themes within the discovery and disclosure system. These process steps in turn provide a potential hierocracy and grouping of themes, derived from codes arranged and sorted previously, in Figure 9. Within Figure 11 a consolidated grouping of themes is presented. The consolidated thematic map is a result of moving between the coded extracts, and the entire data corpus, enriched with the analysis techniques, and contexts. The mapping shows a number of different aspects from within the VDDS. Specifically, themes are linked between previously unknown concepts, for example the relationship between the perception of punishment and the stance a discoverer takes when this perception of punishment is high. Furthermore, the transmission of these experiences, via historic social media or other communication channels is again a significant factor when choosing how to disclose. Finally, the link between new vulnerability markets, and the ethical considerations discoverers undertake is significant as it again influences the stance of the discoverer.

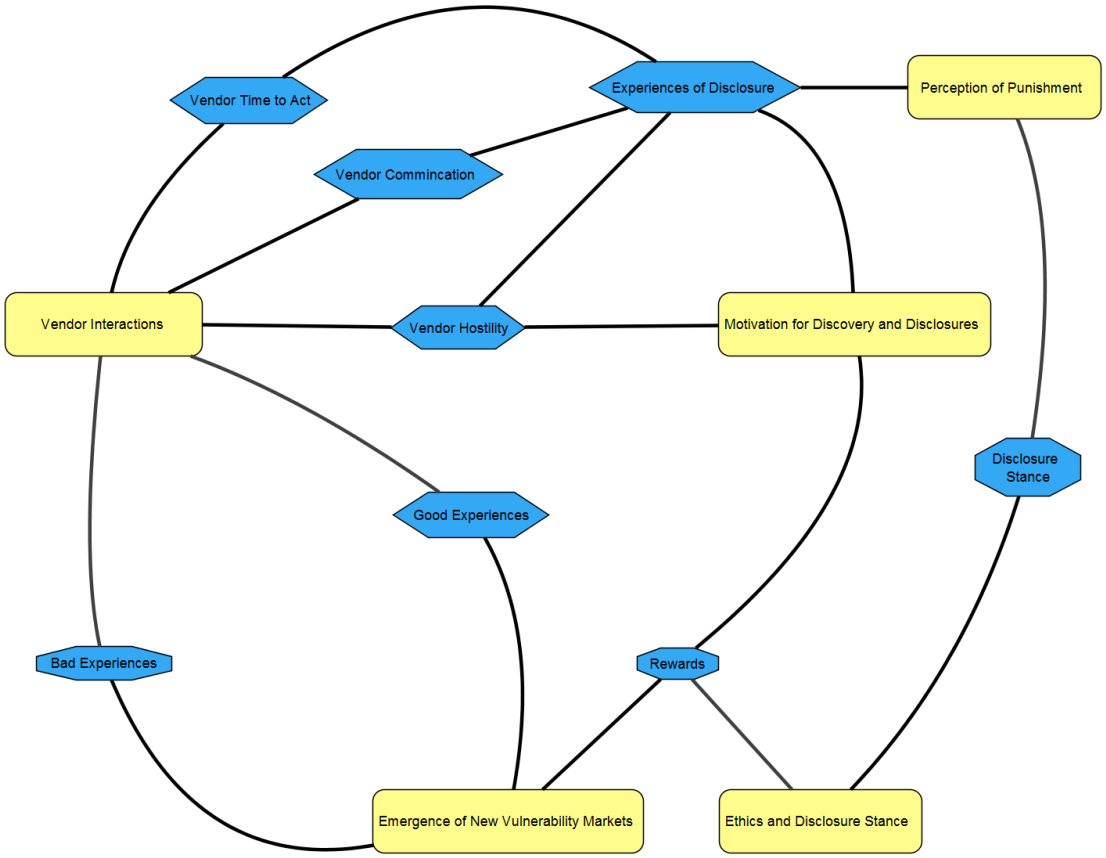


Figure 11 - Developed Thematic Map.

4.6.3 Candidate Theme 1 - Perception of Punishment

Throughout collected data and analysis is the repeated concepts of punishment and fear of punishment, specifically around actions of disclosure. Words and phrases such as “nasty”, “threat”, “repercussion”, “hate” and “That could turn nasty…” are examples of the way in which vulnerability discoverers or members of the community describe disclosure interactions. Using the social media dataset the most common code that is associated with this theme is Emotion (fear), consisting of data items representing fear of reprisal from disclosing a vulnerability to a vendor, which accounts for 46% of all instances within the punishment theme. The is followed by anger, 38% and hate 10%.

In one example **icambridge** comments on a post titled ‘what to do when a company refuses to fix a vulnerability I disclosed to them’, posted on the social media platform Reddit.

> *“That could turn real nasty, real quick. They could see it as a threat, report it to law enforcement claiming you're threatening to release”.* (Enoughalready and Icambridge, 2012)

Repeating words and phrases such as “nasty”, “threat”, “threatening” suggest an adversarial punitive relationship between both discloser and the recipient of the vulnerability information. This is further evidenced by a comment of a participant known as **enoughalready** discussing previous experiences when disclosing a vulnerability:

> “I was able to get in touch with a VP in IT, but I was unhappy with the results. I contacted a lawyer, and it basically came down to it's best to just drop the whole thing”. (Enoughalready et al., 2012)

Furthermore, in a different forum **Jessaustin** comments upon the probable course of action within a corporate environment whereby legal consequences are brought to bear:

> *“It seems likely that there was good-faith negotiation between the researchers and some reasonable people at FireEye. Then some dumbass executive (general counsel, perhaps?) got wind of the proceedings and decided to blow shit up”.* (Jessaustin, 2012)

Jessaustin suggests a possible dissonance between initial organisational members that are tasked with receiving vulnerability reports and acting upon them, and policy governance within the organisation. Phrases such as like *“blow sh*t [sic] up”* signify an escalation toward a punitive stance toward the discloser, possibly in an almost accidental manner:

> *“There are 2 options for the product vendor ... he hates us because he can not see his own flaws/mistakes/fails ... or he loves us because he*

can now see his flaws/mistakes/fails. Nothing between" (Eduard Kovacs, 2011). [Benjamin Kunz-Mejr Interview]

Using the data items that were coded under the Researcher (Vendor Hostility) code the top five frequently identified keywords (highlighted) are inform, site, care, fix and users. These keywords were used in the context of discussing the way in which vendors deal with a disclosure event. It is interesting that in the context of informing the vendor, discoverers refer to vendors as 'they' in context.

"Alright, so what do I do here? I want to **inform** *the developers, however I've been through this before, and it didn't go that well. They didn't* **inform** *the users. They didn't care. Just fix the flaw and move on, as if nothing had happened"* (I_didnt_do_that, 2011).

A detailed example analysed using linguistic connectors of the type of narrative that surrounds the disclosure of a software vulnerability is via a user known as Pak from Hackernews.com, with the context of a 0day vulnerability being disclosed to American Express with no response from the company, and subsequent releasing of the vulnerability via the full disclosure stance in the final paragraph of Pak's post:

"Sure, none of you **(1)** should be thrilled **(2)** about the situation because as technically-oriented people with generous motives **(3)** the system is not set up to serve you. But that's not a failure **(4)** of the system, except maybe from your own individual perspective. Believe me, AmEx has done **(5)** the cost-benefit analysis and they **(6)** are saving boatloads of money by having those rare well-intentioned hackers **(7)** listen to some hold music, because it is too expensive **(8)** to sort you out from the thousands of loonies that got a phishing email. Security breaches are an acknowledged risk and they are already prepared to absorb **(9)** their effects on multiple levels" (Pak, 2011) [numbers added for analysis]

(1) & (6) The use of the word "you" indicates a perceived separation or grouping between entities within the system. In conjunction with 6, the word "they" also suggests a separation between groups, this is almost

certainly suggesting an adversarial relationship between two groups within the system, most probably between ‘discoverers’ and ‘software vendors’.

(2) & (3) & (4) Being “thrilled” at the situation is a behaviour or action that is associated with the preceding action of full disclosure. The descriptive nature of the text in 3 and 4 posit the emotional investment which people are perceived to make within the system and the seemingly disingenuous perceived actions by software vendors.

(5) “Has done” indicates actions that have been taken, in this case undertaken an assessment of cost vs benefit.

(6) & (8) & (9) Actions again surrounding the activities within the system.

4.6.3.1 Analysis

What emerges from the data, in particular the notion that discoverers, researchers and hackers are not taken seriously by software originators, or when they are listened too the normal course of action is generally punitive in nature. Furthermore, there seems to be a distinct division between discoverer entities within the system, specifically an adversarial relationship between ethical and supposed unethical discoverers. An overarching feeling of persecution and lack of control seems to be present from within the VDDS, in particular when discoverers have tried to reach out in good faith. Moreover, there is an emerging theme that researchers who disclose vulnerabilities to software originators do, or more importantly *did* so for the right reasons, but as the level of perceived persecution has increased this ethical stance has been eroded over time.

The software vulnerability discovery community is as diverse as it is distributed across the globe. This is primarily due to the ubiquitous nature of the software that is under scrutiny. It is however still predominately a technical endeavour, with the need to understand how the software works, and the tools that allow the software to be investigated a significant part of the system. The knowledge and tools that are used for discovery are generally available to all and accessible by all who look for them. Tools are distributed via open hacker

forums or closed membership only forms which provide the user with limited instruction how to operate them. The ability to comprehend the knowledge and operate tools is a different matter altogether. The community, is a distinct meritocracy which rewards ability with kudos and praise. This in turn has created a paradoxical relationship between the need to do the right thing, and the need to survive in the real world. For example, the ethical aspects of the treatment of researchers is a key consideration in this. This is especially relevant when those researchers who have discovered a vulnerability, but have been badly treated by a software vendor subsequently change their ethical stance to a more negative position.

The perception of punishment impacts the sentiment that is felt toward the software originators, and exists throughout the discourse of the VDDS. This discourse resides upon social media platforms, where it is readily discussed and crucially archived for future use and citation by discoverers.

4.6.4 Candidate Theme 2 - Software Originator Interactions

Disclosure of vulnerabilities is closely linked to the interactions between the discoverer and software originator. A key finding is that the communication between originators and discoverers has been significantly overlooked previously, and in most cases can be considered to be negative. The interaction between these two entities is typically initiated by the discoverer, and motivated by the release of the vulnerability details. Once released a significant process ensues with delays and miscommunication pervasive through it. This is evident within a post from 2011:

> *"The initial conversation went very well with them and they took the matter seriously. However, they are just dragging their feet now and are no longer returning my emails for status updates".* (Sarphim, 2011)

The processes in place for informing software originators that the vulnerability exists within software are typically little more than a general support telephone number or email address. This is taken by discoverers as an important

statement about the importance software originators treat vulnerabilities within their software. Frequently, frustration is cited as from within the process details about the vulnerability becomes lost within organisational bureaucracy, with little or delayed external communication to the vulnerability discoverer. One example is a dialogue with a software originator known as Bullhorn, stretching between March 2014 and October 2014, a total of 7 months. The discoverer here is looking to disclose a vulnerability within a software portal, yet is met with delays:

> *"October 23, 2014 - After months without hearing a word in response, I decide to ping them again. This actually got the attention of their director of support".* (Arciszewski, 2015)

This is an example illustrating the lack of communication and dismissal of the discoverer and is example amongst hundreds. This has led to a feeling of resentment amongst the vulnerability discoverers and the VDDS community. This has in turn furthered an adoption of different disclosure stances or, in some cases caused an adoption of a different strategy all together (i.e. selling the vulnerability for profit). Evidence also shows that on several occasions a vulnerability has been disclosed to the vendor and this has been met with legal instruments such as cease and desist orders or threatening emails explaining that the researcher is in violation of a legal statute. Furthermore, law enforcement has been informed and have visited the researcher in question. Punishment stance has fostered an atmosphere of resentment and anger toward software originators, causing friction between discoverers and software originators.

However, recently these issues have become are less pronounced due to new disclosure initiatives created by large vendors such as Microsoft, Oracle and Google. However, historical negative examples are still cited to provide justification for a disclosure stance, typically full disclosure. Experiences whilst disclosing a vulnerability range on the one hand from a happy surprise (Damontoo, 2012) to disappointment and disillusionment because of the

conduct of the software vendor (Sarphim, 2011). These experiences are also magnified by the time within which vulnerabilities are removed from the software, and timeframes are constantly discussed to how much time should be given to the vendor to fix the issue. These range from no time at all to almost twelve months in some cases (Sintonen, 2017). Dependent upon which disclosure stance is taken the time frames are dramatically different, but crucially if the communication is good, and previous experiences are positive these deadlines imposed by the discovers are quite flexible (Sintonen, 2017).

4.6.5

4.6.6 Candidate Theme 3 - Ethics and Disclosure Stance

Disclosure of vulnerabilities within the VDDS are categorised into two main groups, full and coordinated, with polarising effects upon the participants of the VDDS. Proponents of full disclosures point to the fact that software vendors have a duty of care to customers of their software and they cannot be trusted to remove the vulnerability without being forced to do so via attacks or customer and monetary pressures (Damontoo, 2012; Eduard Kovacs, 2011; Matt4077, 2016). This is further amplified by the by negative stories that are distributed via social media and online fora.

As with most psychosocial systems the emotional state of a vulnerability discoverer and the social constructs that surround them is a critical factor in establishing the course of action the researcher may take. For example, when researchers consider which disclosure stance to take - either full or coordinated - this choice is directly influenced by either personal experience or, more commonly, via reflecting on community experiences published within social media whilst soliciting advice (I_didnt_do_that, 2011). Many examples of emotional states influencing the disclosure stance exist, such as fear, anger and uncertainty both initially and during the process of disclosing a vulnerability. The historical record of lived experiences during vulnerability disclosure is used and referred to, time and time again (Kdobb, 2012; Stormehh, 2012). There are isolated incidences whereby the discoverer has approached software originators directly, and been welcomed and in some cases rewarded, however

these were few and far between. However, with the advent of such schemes as HackerOne and Bugcrowd, these cases are becoming more commonplace, albeit via a coordinated, paid vulnerability-as-a-service offering.

The choice as to which disclosure stance the researcher takes is strongly related to the software originator actions, previous experiences (both personally and community) and personal ethics. There is evidence showing the solicitation of advice from the community because of previous bad experiences is influential. The evidence shows that during the initial time period of the vulnerability discovery movement (1998 – 2004) there was a sense of 'doing the right thing' with pressure being brought to bear upon researchers if they decide to 'cash in'. Recent cases have been documented around researchers and organisations being ostracised from the community due to this continuing ethical stance, however this is changing rapidly. One example form 2012 is from Cody Brocious, discussing the repercussions of full disclosure:

> *"I've taken significant flak from the security community and others for my lack of adherence to Responsible Disclosure. Hopefully this post will show that that's not always the responsible approach".*
>
> (Brocious, 2015)

The ethical motivation of what to do when a vulnerability is discovered is a hotly debated topic, with no clear majority. However, there is an indication that a preference to disclose responsibly (aka coordinated) at first, then move to a aggressive full disclosure stance if vendors are unresponsive or have a history of inaction.

Significant discussion surrounds the ethical considerations the vulnerability discloser considers when details of a vulnerability is uncovered. This shows that whilst vulnerability research is *demonised* in popular media as unethical or embracing anarchy, in general careful consideration is given to disclosure. When discoverers choose to divulge the details of the vulnerability they have uncovered, typically discoverers try to do the right thing. This attitude is widely

prevalent within the community and is generally accepted as the social norm. This is however changing rapidly with new economic rewards being introduced by software originators, third parties and accepted norms.

The vulnerability discovery community is motivated by ethics and continues to be so. As such the discourse that takes place within the VDDS when a new method of working or motivation for disclosure is introduced is instant and in some cases visceral. However, many participants within the community still believe that the right thing to do is to disclose vulnerability details ethically in a coordinated manner and do so for no reward other than the recognition of discovery alone. This approach is pervasive as it is seen to provide leadership and a sense of purpose other than just monetary reward. Examples of this altruistic behaviour are frequent, and in most cases do not provide anything other than recognition.

Moreover, there is an element of community pressure to show that they are not just a 'bunch of black hats' that provide valuable services to both the community, and end users in discovering vulnerabilities. An example of ethical standards upon which the VDDS holds itself to account is from a blog post from Martin (2000) discussing the responsibilities of how and when to disclose the details of a vulnerability:

> *"The adoption of Full Disclosure is an ethic, our responsibility and the duty of every security professional is to disclose the facts. How and when we disclose the facts is on the other hand the most crucial part of full disclosure. Many factors come into play, the seriousness and implications of the bug, how long has it been since it was discovered, how responsive is the vendor and how dependant are we on the vendor for a patch".* (Martin, 2000b)

The level of sophistication that is apparent here is startling, as software vulnerability discoverers have been characterised as reckless, with a disposition to favour anarchy than harbour a sense of social responsibility. Most striking within the data extract is the use of the word *duty:*

"*…our responsibility and the duty of every security professional is to disclose the facts".* (Martin, 2000b)

If we take the literal meaning of duty to mean 'moral or legal obligation', then the deontological context of the statement is laid bare (Ostler and Swannell, 1983). That is to say that by starting the vulnerability discovery process you are duty bound and obligated to share information about a uncovered vulnerability, potentially for the greater good. Again, this sentiment is evident when a level of self-regulation within VDDS around the right thing to do when discussing vulnerability disclosure:

> "*This is kind of lame. You sound like a real tool. First, you demand to talk to a company that you aren't paying in the medium of YOUR choice as opposed to the numerous methods they provided for you, and then post a hack that could directly harm people. I'm not 100% sure about the legality of posting something like this, but my uncle will, so I'll be sure to forward it to him in the amex legal department. I do know damn well that it isn't very ethical. I hope you're proud of yourself".* (Dan, 2011)

Here we see the context of a vulnerability being disclosed in a full and uncoordinated manner, with sentiments such are 'harm' and explicit comments on the ethical nature of the disclosure. This level of self-regulation is again startling, yet considered routine.

This level of self-regulation is considered to be pervasive, and provided the basis for adoption of policy statements as individuals and organisations approach disclosure – crucially with caveats on time limits and reversion to full disclosure (CERT, 2017; Initiative, 2017; Neff, 2016). This inclusion of policy within choice, again suggests a level of sophistication that is not widely cast within the public populist mind-set. There are however exceptions to this detailed consideration as the focus of this research is focused on the legitimate VDDS. It is inevitable that the vulnerabilities that are disclosed by discovers and subsequently patched by software originators will be used for malicious purposes (Bashar et al., 1997; Ritchey, 2011).

The evolution of both self-regulation and ethical sophistication within the VDDS has occurred over a long period of time in relative terms. The first recorded use of coordinated vulnerability disclosure policy was by the Computer Emergency Response Team (CERT) Software Engineering Institute, based at Carnegie-Mellon University in 2000 (CERT, 2017). The policy, which is largely unchanged today clearly states the conditions once details of a vulnerability have been disclosed – and importantly the penalties for non-compliance of these conditions.

4.6.7 Candidate Theme 4 - Motivation for Discovery

Turning to the process for uncovering vulnerabilities within software, there is suggestion that significant correlation between competency and the number vulnerabilities that are discovered exists. The process generally starts with the creation of the software, and unless the researcher is internal to the software originator the first time that vulnerabilities can be discovered within the software is at the time of release. Prior to the release of the software significant speculation occurs within the community specifically around the severity, quantity and ease of discovery this version will hold. Alongside this new tools that are developed are circulated and used to discover new vulnerabilities. The discovery of vulnerabilities is undertaken by individuals who use tools to uncover flaws in the code. This is either via analysis of source code (white box testing) or by injecting data into the software and observing a response (black box testing). Black box testing is typically used to identify vulnerabilities within software by external discoverers due to the unavailability of the underlying source code. Discussion and discourse around tools and techniques used by researchers is, as one would expect, a highly complex and technical topic. This discourse takes place primarily online, within educational settings or at physical meetings where researchers exchange information on new techniques of tools, for example DEFCON (Egelman et al., 2013).

Within the concept of motivation, the quality of software is an important consideration, often discussed by entities within the system. There is a

pervasive sentiment from specific groups of entities, that of end users and discoverers. The sentiment is centred on the ideal that more could and should be done around removing software vulnerabilities within the software development process, rather than post release. Discussions centre on the costs which increases in software quality would bring adapted through to the creation of markets. Furthermore, potential use of standards is also a topic that is frequently discussed with narratives around the incentives that are used to entice software vendors to secure software, to development standards. Software vendors are considered to be ultimately responsible for the software which they produce, therefore are responsible for the removal of vulnerabilities from the software. To do this post release requires the vendor to communicate with the security researcher (assuming the discovery was not made in house) to establish the details of the vulnerability and ultimately remove the issue from the software.

The motivation for discovery is linked to theme three, ethics and disclosure stance, and theme five, vulnerability markets. Both themes provide motivation to the discoverer, yet what is apparent is the dissonance between the reward for altruistic behaviours, and economic reward. This dissonance is discussed in the next section 4.6.7.

4.6.8 Candidate Theme 5 - Emergence of New Vulnerability Markets

The economics of vulnerability discovery is intertwined with all other themes that are present within the system, and arguably the concept that drives all other interactions. The economic forces that drive the system are a recent phenomenon and are manifest in the formation of three distinct market approaches to coordinated vulnerability disclosure rewards. These approaches are classified as bounties, third party brokers and auctions. Each market rewards the vulnerability discoverer with money, and in some cases marketing merchandise, but recognition of finding the vulnerability is mandatory in all cases.

The act of selling vulnerability, as stated previously is met in some circumstances with hostility. In each approach the level of contention occurring within the VDDS differs dependent upon the perceived piousness, or ethicalness of the market. This ranges from what are considered the most ethical, bug bounties through to the least ethical, third party brokers. Bug bounties are a recent phenomenon, where software originators challenge discoverers to find, and exploit their software for a reward. The second, vulnerability auctions have for the most part, given way to other markets, largely due to the success of bug bounty schemes such as Pwn20wn and Google ProjectZero (Google, 2017; Portnoy, 2010). Ozment (2007) first codified the vulnerability auction, with specific reference to a type known as a reverse Dutch auction with both perfect and imperfect information, guiding the bidders. Auctions are considered to forebears of bug bounties, and are now rarely used. Finally, third party brokers are considered to be the least ethical of the markets as they exist only to find vulnerabilities and sell them to the highest bidder (Radianti et al., 2006, 2007b). Each reward mechanism is perceived to have created a schism between vulnerability discoverers and software originators, as some feel that discoverers should be rewarded for the skills, effort and resources they have used to uncover the vulnerability, not money. Typified by this example from 2013:

> *"Lesson learned: Find a security hole, report it to Facebook, and they don't respond after two attempts? Sell it as a zero day. Incentives matter. And there is always money to be had somewhere else".*
>
> (Toomuchtodo, 2013)

Within the VDDS the advent of new market forces, coupled with the overwhelming perception to adopt an ethical stance when disclosing vulnerability details created a need for a new approach. These factors have shaped and influenced the creation of a so called 'alt-market' environment, and provided an alternative to underground black-markets. Auctions have been a feature of the free market model of vulnerability discovery since the formation of

the market itself and can take various forms. The auctions are regarded as a necessary part of the market and enable vulnerability researchers to get a market clearing price. This price is in most cases disseminated to other participants within the market, showing what the potential rewards are for vulnerability discovery. The converse of the auction is the bounty whereby a fixed price is offered for the vulnerability, and are typically put in place by the software vendor. In most cases these types of bounty offer a gradation of rewards for different types of severity, and the ability to bypass security controls.

A recent addition to the vulnerability market of the *legitimate* brokering entities that provided middle man and escrow services between the researcher and the vendor, or in some case nation states or defence contractors. Notable examples of brokers are the Zerodayinitative, Endgame and Zerodium (Endgame, 2017; ZDI, 2017; Zerodium, 2017). These types of brokers perform two functions that of providing anonymity and negotiation for a price for the vulnerability. This process allows the brokers to also extract value from the vulnerability process by producing threat intelligence on vulnerabilities. The role of brokers within software vulnerability discovery is again a polarising one, with arguments for the continued support and the abolition of them rife within the community. An example of a positive role for brokers comes from raw interview data from Radianti (2010b):

> *"I would rather see a hundred researchers selling to legal markets than having to see anyone to sell on dark markets just because they run out of money".*

Whereas, others see the profiteering of potential brokers, selling the details of vulnerabilities to the highest bidder unethical and potentially dangerous, with this blog extract citing concerns:

> *"Companies have sprung-up to reap the benefits of the money being poured into the exploits market with a business model around finding exploits and then sending them to the highest bidder (often intelligence agencies)".*
>
> (Borgen, 2013)

Brokers also exist in the black market where cyber criminals purchase and use vulnerabilities to infect and steal money from users via malware. As the creation of grey market brokers is a relatively recent one, the discussion around black market brokers is generally seen in a negative light, and could constitute the reason the community does not like brokerage for profit. The use of vulnerabilities, or more specifically the conversion of the vulnerability into an application that can be used to defeat a security control, a process known as weaponisation is a known nuanced and difficult debate (Bradbury, 2015; Fidler, 2014). The use of vulnerabilities was traditionally perceived to be by cyber criminals to steal private information such as credit card details or private personal information (Anderson et al., 2013). However, whilst this still maybe the case, the use of vulnerabilities is now no longer the privy of the cybercriminal as both law enforcement and government intelligence agencies now utilise the benefits of information system exploitation (Schneier, 2015, pp.146–150; Schwartz et al., 2016). An example of potential usage of a vulnerability for malicious reasons is from Borgan, who explains:

> *"If DarthBorgen is a "black hat" hacker he may use that exploit to steal from the company himself or he may sell it to a rival company what would use it in some illegal corporate espionage" (Borgen, 2013)*

Finally, the value of a vulnerability is a significant factor that is discussed at length within the VDDS. The ability to sell a vulnerability in a legitimate manner is a recent phenomenon, with the first auction site known as WabiSabiLabi being launched in 2007 (Bradbury, 2007). The discourse around selling vulnerability details is polarised, as some feel that as the effort has been expended you must be rewarded as time could be spent on other things, with this anonymous example from the raw data provided from Radianti (2010b):

> *"We agree that more and more researchers are interested. I would estimate that the biggest reason in selling to legal markets is that even researchers need to bring food to the table and since one vulnerability can be worth severals [sic] months pay there are several interested in it".*

On the other hand, people feel very strongly about the perception of profiteering from vulnerabilities as it is felt that a civil duty should be upheld, suggesting the kudos is a better motivator than money:

> *"A 0day pat on the head from 4chan is pretty solid nerd cred for the CV".*
>
> (JonnieCache, 2014)

The introduction of vulnerability disclosure platforms such as Hackone.com, Openbugbounty.org and Bugcrowd have provided both an ethically perceived alternative to the brokers, and the uncertainty of the auction. This new type of market instrument seems to be increasing the flow of vulnerabilities, in a controlled and rewarding manner.

An additional phenomenon that has recently sparked debate is the purchasing of vulnerabilities, specifically zero day vulnerabilities by nation states for assumed offensive means. A significant number of alleged brokers have cited occasions when they have been contacted by 'friendly' and hostile nation states to sell vulnerabilities affecting well-known software products. This has been further compounded by the rationalisation of an arms control treaty known as the Wassenaar arrangement, effectively putting vulnerabilities in the same

category as nuclear enrichment technology (Wassenaar, 2015, p.74). Moreover, with the recent outbreak of the WannaCry ransomware which purported to use an NSA derived vulnerability known as ETERNALBLUE, weaponised for criminal intent this issue may start to become more acute (Goodin, 2017b).

4.6.9 Cross Cutting Theme - Time and Delays

Within the VDDS the concept of elapsed time is a significant factor on all that is undertaken - from the amount of time that vulnerability discovery takes, through to the delays that exist when communicating with software originators to disclose vulnerability details. Consequentially, the time that elapses between steps makes up a complete interaction between entities, and forms a fundamental element of the discourse within the VDDS. Two principle delays occur within the VDDS, discovery delays and disclosure delays.

Within both delays, vulnerability discovery is situated within the initial stages of the VDDS. Initial delays occur due to the labour intensive nature, and technical expertise that is required (Holm et al., 2013; Sommestad et al., 2012). Therefore, it is postulated that the time that is required to attain a level of understanding and competence to discover a single vulnerability is interrelated to the number of vulnerabilities that can be found alongside entry into the VDDS. In addition to delays associated with vulnerability discovery, disclosure delays make up a large proportion discourse within the VDDS. Disclosure delay occurs toward the end of the VDDS process, however is not linear in nature. This initial delay - which is defined as the delay that occurs when a vulnerability discoverer chooses to make the details of the vulnerability public is significant. These types of delay are dependent upon the disclosure stance that is adopted.

However, in the case where the discoverer chooses to make the vulnerability public without informing the software originator, known as full disclosure, there is no delay. This is due to fact the discoverer has numerous communication channels that are open to them, ranging from globally available email distribution lists, through to blogs and social media. The impact of full disclosure upon software originators, end users and other entities within the VDDS has been studied at length (Huang et al., 2016; Joh et al., 2014; Ozment, 2007;

Sutton et al., 2007; Woo et al., 2011). Full disclosure is, and continues to be the last step in efforts to communicate and improve the security posture of a piece of software.

4.6.9.1 Coordinated Disclosure Time Delays

Diametrically opposed to full disclosure is coordinated disclosure. Delays within the coordinated disclosure stance are centred on the altruistic or direct method of disclosure. This is opposed to the paid coordination modes, for example via Hackerone or the ZeroDayInitiative. Significant delays occur around the interactions between the software originator, vulnerability discoverers and in some cases, national government organisations. These delays occur when the vulnerability discoverer chooses to disclose details of the vulnerability within a direct coordinated disclosure route, and is met with organisational bureaucracy. Delays within the paid mode, do exist however as there is a monetary transaction occurring, expectations on all sides are different, and therefore delays are minimised.

The usage of coordinated disclosure is underpinned by the notion that if the coordinated process does not complete successfully, the right to disclosure in an uncoordinated and full way is retained. This is also the policy of large disclosure platforms such as Hackerone, Bugcrowd and Google Project Zero (BugCrowd Homepage, 2017; Google, 2017; Hackerone, 2016).

4.7 Collated Disclosure Process Steps

The process of vulnerability discovery and disclosure is not a linear or simple process, particularly if we take the disclosure process in isolation from the rest of the VDDS. Nevertheless, the ability to visualise key processes that make up the VDDS is important. Consequently, an abstraction of the steps and flows that make up the disclosure process has been mapped, shown in Figure 12 below has been constructed from the thematic analysis and models. The mapping in centred around three entities involved within the VDDS, Vulnerability Discoverer, Software Originator and Disclosure Platform (inclusive of brokers and bug bounties). The process is mapped from the perspective of the

vulnerability discoverer, and initiated in the top left, with the discovery of vulnerability.

Step 1: Initially a decision point to choose a disclosure approach, full or coordinated. Depending upon which approach is chosen dramatically reduces the time that the information is released. If a full disclosure approach is taken, then after an internal process of documenting the technical details of the vulnerability is released via public blog or email distribution list such as full disclosure.

Step 2: If the discoverer chooses coordinated disclosure, a second choice on the part of the vulnerability discoverer is required to disclosure altruistically or via a paid route. Again, depending upon which route is taken, an initial interaction occurs between the vulnerability discoverer and either the software originator – in the case of the altruistic – or vulnerability platform in the case of paid coordinated disclosure. The initial interaction is fraught delay, especially the direct disclosure choice. Many examples exist whereby the discoverer tries to disclose details of the vulnerability to the software originators, yet is met with salience or worse, dismissed. Typically, 3-4 iterations occur with discoverers trying to make contact with the originator, with the resultant failure to establish dialogue ending in full disclosure. Furthermore, the elapsed time that is experienced can take in some cases weeks, and months from initial contact.

Step 3: Once initial contact has been made, the steps to verify the technical details of the vulnerability take place. This verification step is important to both the paid for and direct disclosure choices as the ability to reproduce the under controlled conditions within the disclosure platform test environments, or software originator environments. If the conditions are not met to assure the vulnerability is legitimate, the process stops, with these conditions being set by the originator and disclosure platform. Yet, if the vulnerability is considered to be important by the discoverer, then they may wish to revert to full disclosure or attempt to sell the vulnerability to a different disclosure platform and restart the process.

Step 4: Once the validity of the vulnerability is confirmed, the next step is establishing the time scale for disclosure to the public. The establishment of vulnerability disclosure policies to assist in this is now commonplace, and therefore from initial contact, should be known. Although depending upon the severity and technical details of the vulnerability this may be changed. One key factor in the determination of timescales and possible enlargement of them, is a communication feedback loop. This loop is constructed around the constant flow of information between the discoverer and software originator, in the case of direct disclosure, or between originator, disclosure platform and discoverer.

Step 5: If accepted, both the originator and the disclosure platform follow broadly similar steps with the requesting of vulnerabilities disclosure policies, and the assessment of the probably timescales to fix the vulnerability. On the part of the discoverer this is where most delays occur, and the most frustrating as communication with the software originator is generally infrequent and content free. On the other hand, if the paid disclosure path has been taken the disclosure platform can move this process along on behalf of the discoverer and facilitate communication between the two entities. Again, if the discoverer feels that the communication is not appropriate or the fix is taking too long, the right of full disclosure is reserved.

Step 6: Once the vulnerability has been remediated, then the task of remuneration, in the case of paid coordination occurs. This is done via the vulnerability platform, with the software originator not being involved unless the monies flow directly from them.

Step 7: Vulnerability patches are released, and acknowledgements are made public along with any patched software.

There are a significant number of delays that may occur within the disclosure process with multiple communication loops in existence, the potential for them to be elongated or for the process to go off track. As such the involvement of 3rd parties within the process is a welcome one. However, even with disclosure platforms involved within the process things still and can go wrong and the potential for time delays increases significantly. The main aspects of the delays are:

1. Initiation of communication with the Software Originator;
2. Vulnerability validation;
3. Patch creation;
4. Communication on agreement of disclosure;
5. Acknowledgement of discoverer agreements.

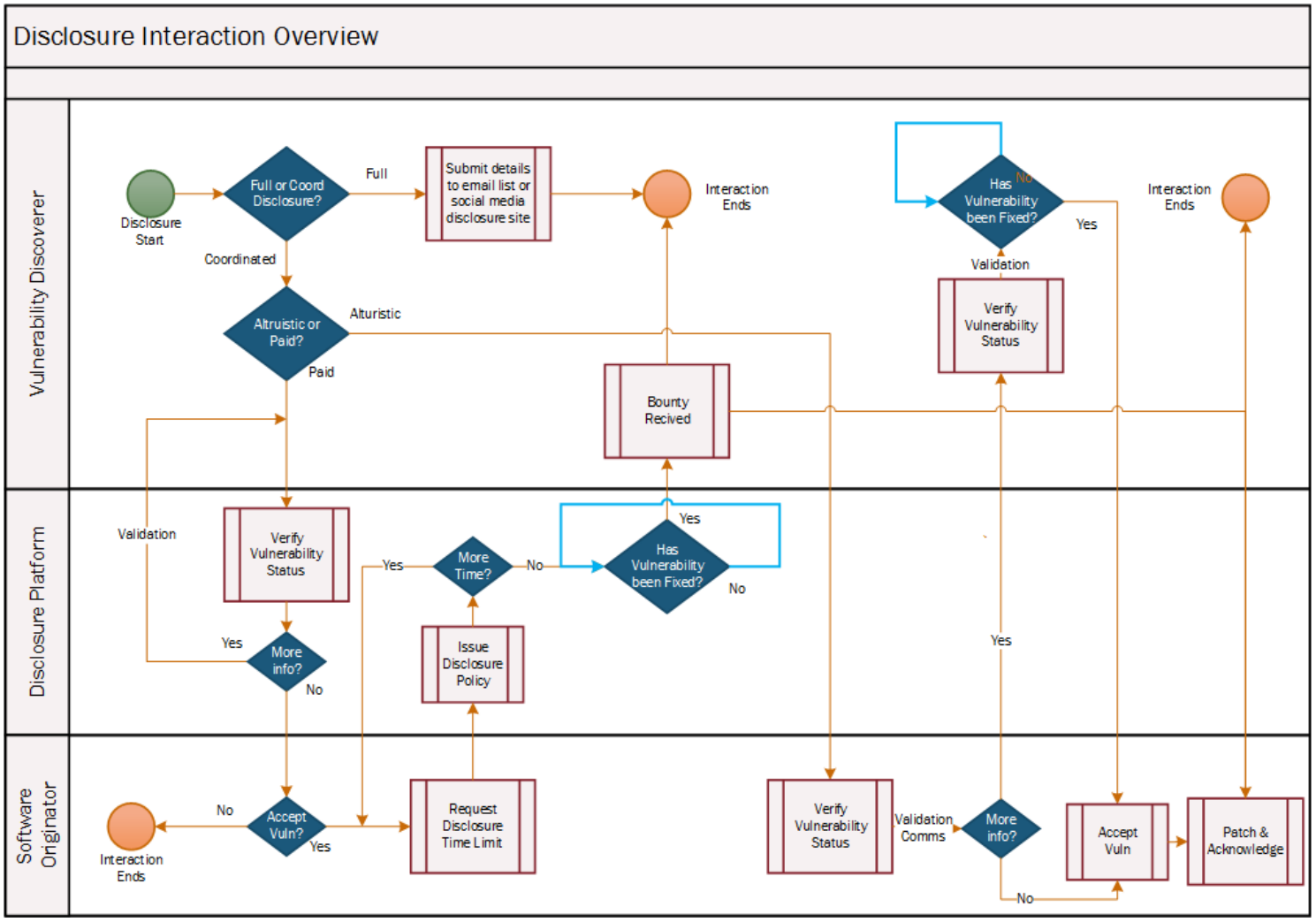


Figure 12 - Discoverer / Software Originator Interaction Process Diagram

4.8 Chapter Summary

Vulnerability discovery has been perceived to be a secret and malevolent act since this discovery of the first vulnerability within Sendmail (Stoll, 1989, p.391). It is argued that this discovery act gave birth to the modern malware movement we see today, with this single vulnerability inspiring a generation of hackers, crackers and criminals to search for vulnerabilities within software (Orman, 2003, p.7; Stoll, 1989, p.392). Alongside this vulnerability discovery and disclosure has been characterised as harmful for a number of years due to the prominence and criminal motivation of Blackhat researchers, and the Whitehat researcher looking to reduce the attack surface of a system or software (Ablon et al., 2014; Rescorla, 2005). Consequentially, this perception has led to the demonization of vulnerability discoverers by large organisations and governments, so when legitimately disclosed, discoverers are met with hostility.

However, this hostility has been replaced in some part, and the act of discovery is now seen in a more positive light, with in-house testing, the rise of white hat careers and vulnerability disclosure programmes, such as hackerone and google project zero (Google, 2016; Hackerone, 2016). Yet, given the nature of vulnerabilities most software vendors still wish to keep the issue confidential and do not wish to disclose detail (Bradbury, 2015). Indeed there have been cases where global organisations have come to metaphorical blows over the disclosure stance the other has taken (Chirgwin, 2017). Hence as a consequence of this secrecy a significant amount of the data that is available for analysis is hidden within software originators, vulnerability disclosure platforms and kept under non-disclosure agreements (Anderson et al., 2013; Bradbury, 2015). However, evidence is available in the public domain providing insight into the processes, interactions, actors and discourse that make up the legitimate vulnerability discovery system. Data that is publicly available is typically located in hacker forums, email or social media platforms or bespoke disclosure platforms.

There are aspects of the system that are in juxtaposition with each other. This is evident in the ethical considerations that exist within the VDDS, which are highly dynamic and change quickly. For example, the level of self-regulation within the VDDS is a major finding, as discoverers in most circumstances prefer to coordinate the discovery of vulnerabilities with software originators, only opting for a full disclosure stance as a last resort. Moreover, the overwhelming majority of sentiment that exists within the VDDS is superficially positive, coupled with a high level of sophistication when considering the moral issues when disclosing a vulnerability. Alongside this, new market structures and instruments have come into existence to facilitate ethical disclosure, and incentivise it. For example the creation of Hackerone, which is seen as the preeminent place to disclose ethically, and with recompense.

Additionally, negative factors influence current and potential future vulnerability discoverers to potentially adopt a 'full disclosure' posture, or if motivated to do so, seek monetary compensation for details of the vulnerability from either a 3rd party broker, vendor or in extreme cases the black market. These negative experiences cluster around traditional IT organisations, for example Cisco, Symantec and Hewlett Packard. Conversely, a small number of internet savvy organisations such as etsy.com and Google have been praised for clear vulnerability policies, communication and are held up by the community as exemplars. Past Experiences and emotion are key when choosing which path to take. The structures that exist within the VDDS are centred around four main entities, with the most important being the vulnerability discoverer themselves. The discoverer is the centre of all vulnerability discovery events, and if it did not exist then the vulnerability discovery system itself would not exist.

A key aspect of the VDDS is the level of asymmetry between entities and interactions, whereby the power of disclosure resides with the vulnerability discoverer. This is synonymous with the asymmetry of cyberspace (Geers, 2010; Moore et al., 2010). Therefore, the disclosure of vulnerabilities in a coordinated manner is preferable from the perspective of the software originator. However, almost half of the direct interactions between vulnerability

discoverers and software originators have some level of acrimony, with full disclosure taking place in some circumstances.

Vulnerabilities tend to follow a specific lifecycle whereby the vulnerability is created by the software vendor, the vulnerability is then latent within the software for a period of time, then it is discovered. Once discovered then an ethical choice is made, to either profit from the vulnerability (legally or illegally) or responsibly disclose the vulnerability to the software vendor. Once this decision has been made then the role of the vulnerability research effectively comes to an end, other than attracting notoriety of Kudos for the act of discovery. The final stages of the vulnerability are dependent upon the ethical choice which is made. If the vulnerability is disclosed responsibly then the software vendor will fix the vulnerability after a certain period of time, and release this to the software users. If a less ethical choice is made to sell the vulnerability to a broker or cybercriminal then exploits that use the vulnerability can be created and used within malicious software.

From a qualitative perspective, the VDDS, structures, interactions and dynamism have been characterised. This qualitative foundation provides a solid position to construct a quantitative model upon it. Therefore, in the following chapter several quantitative datasets have been collected to explore identified themes, and add numerical fidelity to them. Behaviours of both entities within the VDDS and the parameter space of the interactions between them will be investigated. Quantitative datasets have been collected, by using a set of techniques known as exploratory data analysis (EDA) explored and characterised.

EDA was first cited as a distinct framework in Tukeys' seminal work, where he proposed a philosophical approach to quantitative data analysis (Tukey,1977). Turkey provided an approach to allow the researcher to explore the data and utilise the techniques laid down in the EDA framework (probability plots, descriptive statistics and outliers etc). These techniques provide descriptive statistics about the data that has been collected and allow several techniques to be deployed on the data.

5 Quantitative Data Collection and Analysis

The identification of key themes drawn from the data is crucially important; nevertheless, it is only part of the investigation as to which factors contribute to the growth of software vulnerabilities. By building upon the previous Thematic Analysis this chapter presents an exploration of the numerical aspects of identified themes and concepts. This exploration is presented as a set of state variables, which are used to provide the basis for the construction and ultimate simulation of the VDDS theory and model. The EDA framework was used to investigate identified state variables, and to determine whether the themes derived from the previous analysis represent the VDDS accurately. Techniques from within the EDA framework were used to establish a historical *feel* and *shape* of collected data. EDA as a technique emphasises using visual aids and descriptive statistical methods to explore the data, and used previously as a technique to provide rigor to the analytical process used to derive System Dynamic models (Saunders et al., 2009, p.428).

The primary focus of this chapter is to establish the initial conditions of the identified state variables, and characterise their behaviour over time. Once considered, the systemic behaviour and potential interactions between state variables can be mapped. To do this, measurements such as arithmetic location, spread and shape of any probability distributions were completed to understand what if any parameters should initialise any simulations. For clarity, arithmetic location is defined as the point where the distribution is 'anchored', or which set of values best describe the entire set of values, shown by mean, median and mode statistics (Hartwig and Dearing, 1979, p.13). Spread is defined as the variability or dispersion of values throughout the observed values, shown by standard deviation statistics. (Hartwig et al., 1979, p.13) Finally, shape is defined as the distribution, modality, measure of skewness and outliers (Hartwig et al., 1979, p.13). All data is presented as historical reference modes, which show the behaviour of that state variable over a given time period.

5.1 Data Collection and Data Limitations

A range of data sources, discussed in chapters 3 and 4 have been used to construct key reference modes. All raw data were processed using a combination of data analysis tools MySQL 5.4.11 community server, Mathworks Matlab 2016b, Python 2.7.13 and R – Statistical computing.

During the quantitative data collection process three key limitations were observed about the data. These limitations include incomplete data, methods of data collection at source and source bias. In the first instance data that has been collected is almost certainly incomplete, hence the abductive nature of this research with incomplete data is a regrettable feature within the sciences, more so when undertaking an exploratory investigation of an online phenomenon such as vulnerability discovery and disclosure. It is however possible to compensate for the lack of complete data by imputing, or calculating any missing data with approximate replacement values.

The second limitation is centred on the collection methods and quality assurance that is used by third parties when collecting data from end users (e.g. ExploitDB). These methods are unknown and therefore must be viewed with an appropriate level of confidence. Again, it is possible to compensate for issues that arise from insufficient quality control of collected data. Where possible data that is used to inform state variable construction has been compared with at least one other similar dataset, and checked for consistency. Where this has not been possible, this is highlighted. Furthermore, it is reasonable to assume that third parties have made every effort to correct and deal with omissions and errors, and the quality of the data that has been collected from third parties is of reasonable quality.

Finally, a general limitation to all collected quantitative data used is the nature of the phenomena under investigation - it is globally distributed and opaque. As such, data sources that have been selected are the most prevalent, or popular, and therefore taken as the most representative of the feature that is being observed. For example, data representing sentiment toward software originators was collected from social media platforms such as Hackernews.com

and Reddit.com as this is where the richest data was located. Other sources of data exist, however are potentially less concentrated or not publicly available. Where data is missing or there is a potential bias or error within the data, statistical techniques that are used to compensate for these issues are noted. The data sources that have been collected are outlined below in Table 18, along with sources, associated variables and online location.

Table 18 outlines several properties about data that is used within the EDA framework. Variable name represents the descriptive aspect of the behaviour, and is a direct link to previous themes. Data source, lists the name of any locations of raw data, and appropriate detail. Location is the online web URL, and date accessed shows the time the data was collected and processed.

Variable Name	Data Source	Location	Dates Accessed or Collected
Variable A: Researcher sentiment	Hackernews.com; Reddit.com; Online blogs (See chapter 4 for details)	Online	November 2014 – April 2016
Variable B: Number of Vulnerability Discoverers within System	ExploitDB.com	www.exploitDB.com	December 2016
Variable C: Vulnerability Removal Rate	National Vulnerability Database	https://nvd.nist.gov/	December 2016
Variable D: Full disclosure and Coordinated Disclosure Ratios	Open bug bounty trading platform	www.openbugbounty.org	Sept 2015 -Jan 2017
Variable E: Time to fix vulnerability from coordinated disclosure	Multiple academic studies Full Disclosure Email Archive	http://seclists.org/fulldisclosure/	February 2015 – February 2017
Variable F: Monitory reward	Hacker One Bug Crowd	www.hackerone.com www.bugcrowd.com	January 2014 - December 2016
Variable G: Number of Bug Bounty Schemes	Hacker One Bug Crowd	www.hackerone.com www.bugcrowd.com	November 2013 – Apr 2017
Variable H: Vulnerability Discoverer Participant Activity	ExploitDB Open bug bounty reporting platform	www.exploitdb.com www.openbugbounty.org	January 2017
Variable I: Software Originator Market Share	W3Cschools Wikimedia Foundation Statcounter	http://gs.statcounter.com/ www.w3cschools.com https://analytics.wikimedia.org/dashboards/browsers/	March 2017
Meta Variable: Time and Delays	Full disclosure email distribution list Vulnerability Policies	http://seclists.org/fulldisclosure/	Multiple

Table 18 – Data Collection Sources and Locations

5.1.1 State Variable Selection from Themes

The System Dynamics modelling process requires the identification of key state variables that represent the phenomena under investigation. Therefore, extracting representative variables from the identified themes that best describes the VDDS is a critical step in completeness of theory and models. The pulling out of variables which numerically express the state of a system or sub-system is known as state variable exposition. State variables are used to describe the state of a system at a specific time, or over a defined period of time (Palm, 2010, p.229). Within System Dynamics these behaviours are known as reference modes, and describe how the state variable changes over time (Sterman, 2000, p.90). Variables, and therefore reference modes, are drawn from within the five themes identified previously (Perception of Punishment, Disclosure Stance, Vendor Interactions, Motivation for Discovery, Emergence of Markets) and are a direct representation of the structure of the VDDS and relationships within it. The Identified themes also provide a rich narrative upon which to both base dynamic changes of the VDDS and build the structure of the model (Coyle, 1996, p.26; Morecroft, 2015, p.60).

State variables were selected based upon identified themes, with three key criteria influencing the choice; necessity, aggregation and directionality (Albin et al., 2001, p.10). Each state variable characterises factors within each theme as it impacts the VDDS. For example, in the case of vulnerability interaction time, this variable has a direct causal link to the sentiment within the VDDS. Furthermore, a key factor within the VDDS is time. Time is represented as a both an intrinsic aspect of the variables (i.e. the behaviour is mapped to elapsed time) and within variables themselves. Thus time represents how the theme evolves, and is represented by relationships and a number of processes steps as opposed to state variables. Variables such as number of quantity of vulnerabilities, the disclosure route that is taken and quantity of active discoverer are all provided as level of activity, initialisation parameters and associated rates of those activities. A mapping of identified themes and derived state variables is given in Figure 13.

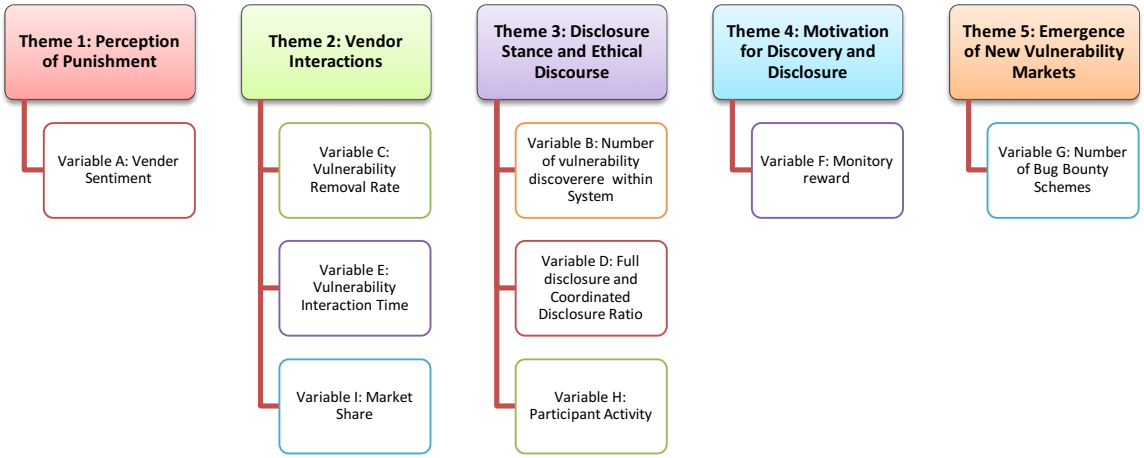


Figure 13 - Systemic Themes and Variable Grouping.

5.2 Reference Mode Identification and Construction

The System Dynamics approach provides several techniques that allows for the construction and exploration of complex models, which express systemic behaviours over time. However, to begin the process reference modes of key variables must be defined. Sterman (2000) states that a reference mode is a

"set of graphs and other descriptive data showing the development of the problem over time" (Sterman, 2000, p.90).

Building upon this Saeed (1998) characterises a reference mode as an abstraction of problematic behaviour that is qualitative, intuitive and shows tendencies that a system exhibits. Typically any sufficiently complex problem expressed as a reference mode will have multiple modalities and will need to be 'sliced' to show these differing behaviours (Saeed, 1992, 1998; Sterman, 2000). This method of slicing the problem space into different partitions, or in our case themes and state variables is useful as it allows for meaningful separation and consolidation phases to be undertaken. Saeed (1998) argues that reference modes, whilst incorporating the historical aspects of the variable under scrutiny should not solely rely upon historical observations; they must also incorporate both the past and abstract concepts (Saeed, 1998).

5.2.1 Initial conditions

A key step in the creation of an empirically based model is the definition of the initial conditions that are used to initiate the model. In the case of the VDDS the conditions and parameters such as number of participants entering the system, elapsed time and participant sentiment are all part of the initial conditions of the system. Furthermore, the parameter space (e.g. minimum and maximum values) of variables within the system is also important. By undertaking statistical investigation in the form of descriptive statistical calculations a more complete understanding can made of the VDDS.

5.3 Descriptive Statistics for Total System Reference Mode

Sterman (2000) states that to gain a base understanding of the system under investigation, an overview of the entire system is required. In the context of this research, this is known as a total vulnerability discovery and disclosure reference mode. To construct the reference mode, a basic technique from within the EDA toolkit, run sequence plot is used. Run sequence plots are defined as a way to summarise a univariate dataset over time (Chambers, 1983, p.95; NIST, 2012).

Initially, the total reference mode provides an orientation as to how the increase in vulnerability growth has behaved historically, and what if any patterns are present. Previously single software application growth curves have been examined via run sequence plots, using cumulative frequency techniques. (Alhazmi, 2006; Alhazmi et al., 2005, 2007, Kim et al., 2007, 2016; Lewis et al., 2015; Woo et al., 2011). Present within these studies are types of both sigmoidal and exponential growth curves that are well-known phenomena within the physical sciences. System Dynamics texts suggest that in some cases common system archetypes and feedback processes are responsible for these types of behaviours (Cash, 2005; Sterman, 2000, pp.264–290).

Construction of the total system reference mode uses the standard repository for vulnerability recording, the National Vulnerability Database (NVD). NVD holds 79249 unique vulnerabilities (as of Dec 4th 2016) with the total number of

entries within the database increasing from the initial recorded vulnerability in Jan 1998 to 79249 vulnerabilities, a period of 18 years. Table 19 provides descriptive statistics for the total vulnerability reference mode, and constructed reference mode based upon this number of vulnerabilities disclosed and recorded publicly across all software globally, within Figure 14. The mean number of vulnerabilities within the total system reference mode is 4364, with the median value of 5063. The standard deviation of the total system is 2189, with population standard deviation calculated using the following formula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Equation 3 – Standard Deviation

There is a general upward trend of vulnerability discovery since data started to be collected in 1998, with a mean increase of 285 vulnerabilities year on year. Furthermore, there have been an above average number of vulnerabilities from 2007 onwards, with one exception 2011 which was only marginally above the mean. There is also a large spread, denoted by the standard deviation figure of 2189. Inspecting the data we can see that 2006, 2007, 2014 and 2015, an abnormally high number of vulnerability disclosures in those years were recorded.

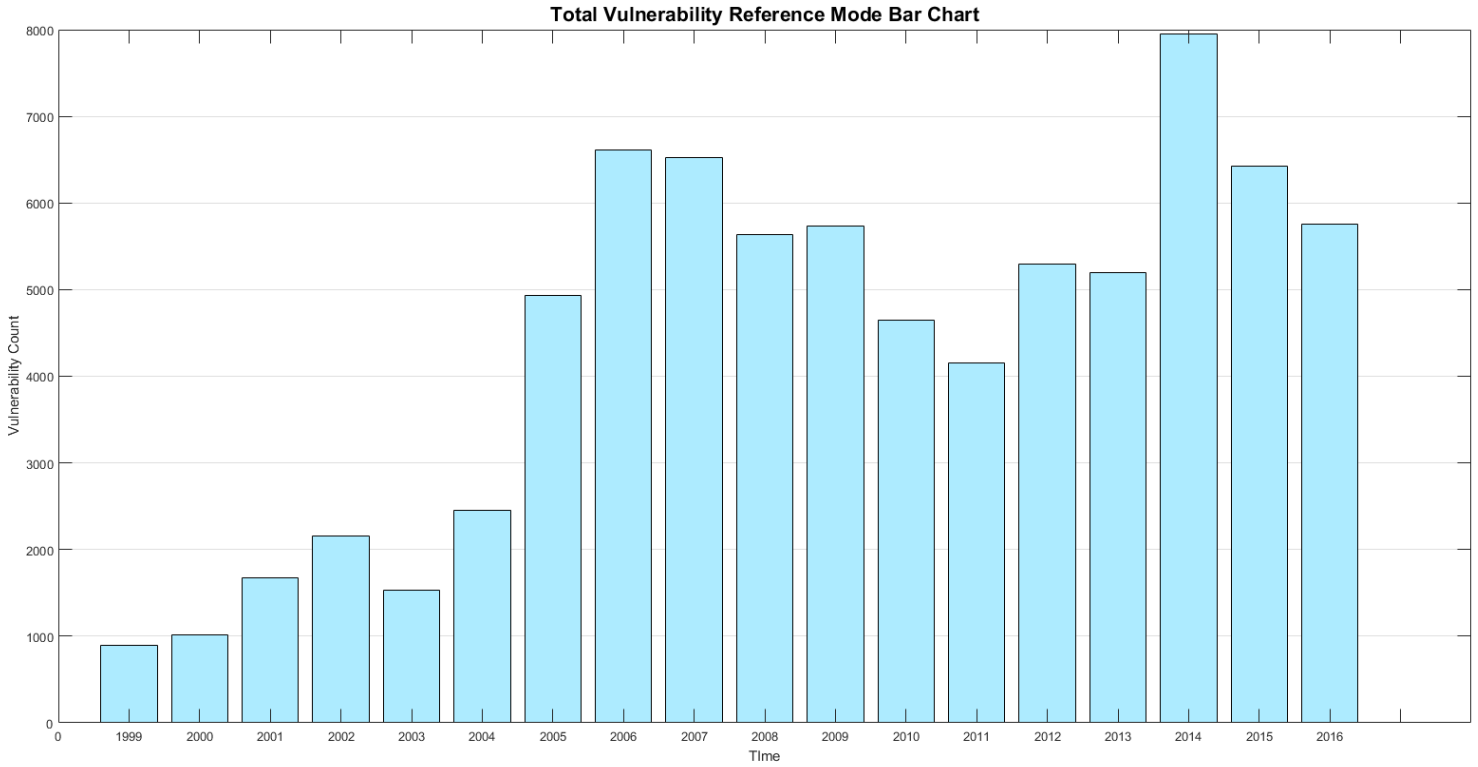


Figure 14 – Vulnerability Trend 1998 – 2017

Number of Disclosed Vulnerabilities 1998 –2016	
Mean (μ) over all years	4364
Median	5063
Standard Deviation (σ)	2189
Total Number of disclosed vulnerabilities (*N*)	79249

Table 19 – Total system reference mode Summary Statistics

To characterise the VDDS, Table 20 below shows that in 2006 an unusually high number of vulnerabilities from Microsoft and Oracle were present, with Microsoft accounting for 3.95% of all vulnerabilities that year and oracle 2.21%. The Microsoft percentage decreased to 3.73% in 2007, with Apple replacing Oracle with 3.15%. A similar pattern is also present for 2014 and 2015 with IBM accounting for 5.68% of total vulnerabilities in 2014, and Apple accounting for 9.3% of total vulnerabilities in 2015. Comparing Microsoft across all 4 years we can see that the proportion of total number of vulnerabilities increased from 3.95% in 2006 to 8.00% in 2015 despite similar totals previous years. Indeed, the top 30 software originators represented in Table 20 below account for 24.96% (2006), 29.68% (2007), 43.38% (2014) and 65.41% (2015) of all disclosed vulnerabilities respectively.

	2006 (Total No. 6659)		2007 (Total No. 6594)		2014 (Total No. 7946)		2015 (Total No. 6588)	
	'microsoft'	**3.95%**	'microsoft'	**3.73%**	'ibm'	**5.68%**	'apple'	**9.30%**
1	'oracle'	**2.21%**	'apple'	**3.15%**	'oracle'	**4.72%**	'microsoft'	**8.00%**
2	'apple'	**2.07%**	'ibm'	**2.00%**	'cisco'	**4.59%**	'cisco'	**7.44%**
3	'mozilla'	**1.77%**	'oracle'	**1.88%**	'microsoft'	**4.40%**	'adobe'	**7.33%**
4	'ibm'	**1.35%**	'sun'	**1.74%**	'apple'	**3.33%**	'oracle'	**7.16%**
5	'linux'	**1.34%**	'php'	**1.71%**	'google'	**1.91%**	'ibm'	**4.74%**
6	'sun'	**1.16%**	'cisco'	**1.70%**	'adobe'	**1.72%**	'google'	**4.22%**
7	'cisco'	**1.04%**	"ipswitch"	**1.21%**	'redhat'	**1.56%**	'mozilla'	**2.81%**
8	'joomla'	**0.77%**	'mozilla'	**1.18%**	'mozilla'	**1.48%**	"abe"	**1.52%**
9	'mybulletinboard'	**0.63%**	'hp'	**0.97%**	'linux'	**1.47%**	'hp'	**1.52%**
10	'novell'	**0.63%**	'linux'	**0.94%**	'hp'	**1.27%**	'emc'	**1.17%**
11	'php'	**0.63%**	'joomla'	**0.85%**	'apache'	**1.10%**	'linux'	**0.97%**
12	'hp'	**0.59%**	'ca'	**0.76%**	'sap'	**1.04%**	'sap'	**0.93%**
13	'drupal'	**0.56%**	'wordpress'	**0.73%**	"abe"	**1.00%**	'apache'	**0.88%**
14	'symantec'	**0.53%**	'apache'	**0.67%**	'emc'	**0.77%**	'redhat'	**0.83%**
15	'adobe'	**0.47%**	'bea'	**0.67%**	'openstack'	**0.69%**	'siemens'	**0.55%**
16	'xerox'	**0.47%**	'symantec'	**0.59%**	'owncloud'	**0.67%**	'mediawiki'	**0.50%**
17	'bea'	**0.42%**	'hitachi'	**0.56%**	'juniper'	**0.59%**	'symantec'	**0.50%**
18	'invision_power_services'	**0.42%**	'drupal'	**0.55%**	'moodle'	**0.59%**	'wireshark'	**0.50%**
19	'phpbb_group'	**0.42%**	'adobe'	**0.53%**	'magzter'	**0.57%**	'juniper'	**0.49%**
20	'freebsd'	**0.38%**	'web-app.org'	**0.47%**	'gnu'	**0.56%**	'openssl'	**0.49%**
21	'mambo'	**0.35%**	'redhat'	**0.44%**	'plone'	**0.50%**	'ffmpeg'	**0.44%**
22	'gnu'	**0.33%**	'novell'	**0.39%**	'qemu'	**0.45%**	'fortinet'	**0.44%**
23	'ca'	**0.27%**	'xoops'	**0.39%**	'drupal'	**0.44%**	'gnu'	**0.43%**
24	'netbsd'	**0.27%**	'vmware'	**0.38%**	'siemens'	**0.41%**	'xen'	**0.43%**
25	'apache'	**0.24%**	'wireshark'	**0.32%**	'debian'	**0.39%**	'gehealthcare'	**0.39%**
26	'woltlab'	**0.24%**	'clam_anti-virus'	**0.30%**	'mcafee'	**0.39%**	'moodle'	**0.38%**
27	'wordpress'	**0.24%**	'trend_micro'	**0.30%**	'xen'	**0.37%**	'mcafee'	**0.36%**
28	'deluxebb'	0.23%	'alstrasoft'	0.29%	'symantec'	0.36%	'openstack'	0.35%
29	ipswitch'	0.23%	'asterisk'	0.26%	'wireshark'	0.36%	'owncloud'	0.34
30	'jelsoft'	0.23%	'gnu'	0.26%	'php'	0.34%	'php'	0.033

Table 20 – Vendor Percentage Variation in 2006, 2007, 2014 and 2015

What is striking about these figures is that it suggests that if vulnerabilities were addressed quickly, or prior to the release of the software, then a large portion of risk that organisations face could be mitigated. This indicates that most vulnerability discoverers may concentrate upon the most vulnerability prone or available software. What motivates, or causes vulnerabilities to be discovered within these software applications is outlined within chapter four. Figure 15 shows a comparative change in the percentage over time for the top seven software originators.

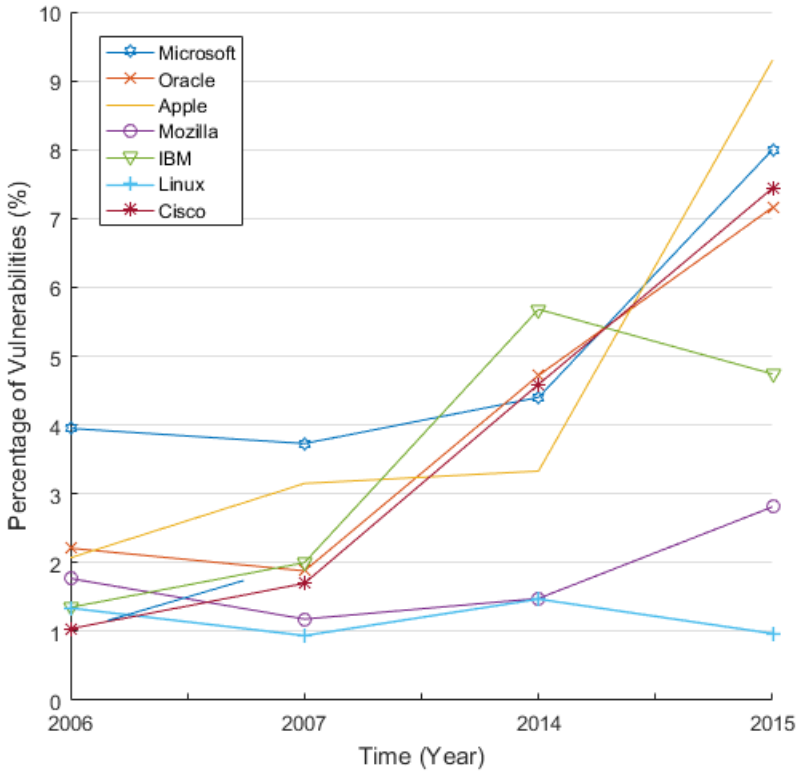


Figure 15 - Percentage Change in Discovered Vulnerabilities 2006 – 2015

5.3.1 Three Key Inferred Futures

As stated by (Saeed, 1998) to be useful, reference mode illustrations must take the concrete historical record alongside potentially hidden abstractions of the constituent parts of the problem, in our case the themes uncovered within the thematic analysis (Saeed, 1998). Accordingly, identified themes are used to infer potential future trends and patterns giving rise to possible modes of

behaviour of a long period. Therefore, three probable futures of the total vulnerability reference mode are presented, all of which take place after March 2017, which is denoted by the vertical black line. A detailed quantification of the range numbers associated with these possible scenarios is given in chapter six, whereas a qualification is given here. The three inferred futures are shown in Figure 16 below.

The first future that is hypothesised to occur is **flat**. This future exists where the number of recorded vulnerability disclosures and therefore number of discovered vulnerabilities found decreases to zero. This assumes several things: a) The perfect debug of all new vulnerabilities upstream within the development process and removal of vulnerabilities within software; b) once a vulnerability is removed from the software ecosystem, it is not reintroduced and that no new vulnerabilities are introduced via new software versions; and c) this future is denoted by the bold red line.

Second future is the continued increase of vulnerability at **steady** rate of discovery and subsequent disclosure events, albeit at an increased rate than the previous rate. This scenario assumes that: a) new vulnerabilities are discovered in existing software, but improved software practices reduce the number of vulnerabilities with new software; b) the vulnerability discovery system continues to add new vulnerability opportunities to sell or receive bounties and; c) the number of participants increases. This future is denoted by the bold magenta line.

The third future is the **exponential** increase in vulnerability discovery and disclosures. The assumption here is that: a) new vulnerabilities are introduced to new software and software practices are not significantly improved from today's levels; b) the number of participants within the system increases significantly; c) opportunities to sell vulnerabilities increases significantly and; d) the desire for vulnerabilities increases. The resultant historical trend shows a clear increase in the cumulative number of vulnerabilities that have been discovered over the past 31 years.

Within each future, there are characteristics which drive the growth of vulnerabilities within software. These characteristics are borne out of the policy decision that are made by software originators and discoverers alike, and contribute to the growth. As such it is suggested that growth in vulnerabilities is influenced by such policies, and therefore can be changed.

5.3.2 VDDS Time Epochs

The behaviour of the VDDS can be seen to split into specific time boxed or epochs. The initial VDDS epoch is a period of uncontrolled and unconstrained growth of vulnerabilities. The time span of this epoch runs from the mid 1990's through to approximately 2009-10, with the defining characteristic of epoch one as the demonization of the vulnerability discoverer. Furthermore, this characterisation included the increased dependence of society upon potentially vulnerable information systems used to conduct commerce and communications. Epoch two, commences around 2010 with the advancement of vulnerability disclosure platforms, and changes in perception towards vulnerability discoverers together with software originator market rewards. The defining characteristic of epoch two is the emergence of new markets for vulnerabilities and associated rewards. Epoch two is currently underway now. Epoch three, is speculative and almost certainly consists of new markets and services, with an acknowledgement that vulnerabilities are central to the risk mitigation process. Epoch three is allied with the inferred futures and may take the form of one of the three possible, flat, steady or exponential dependent upon the conditions, and policies the VDDS adheres to at that time.

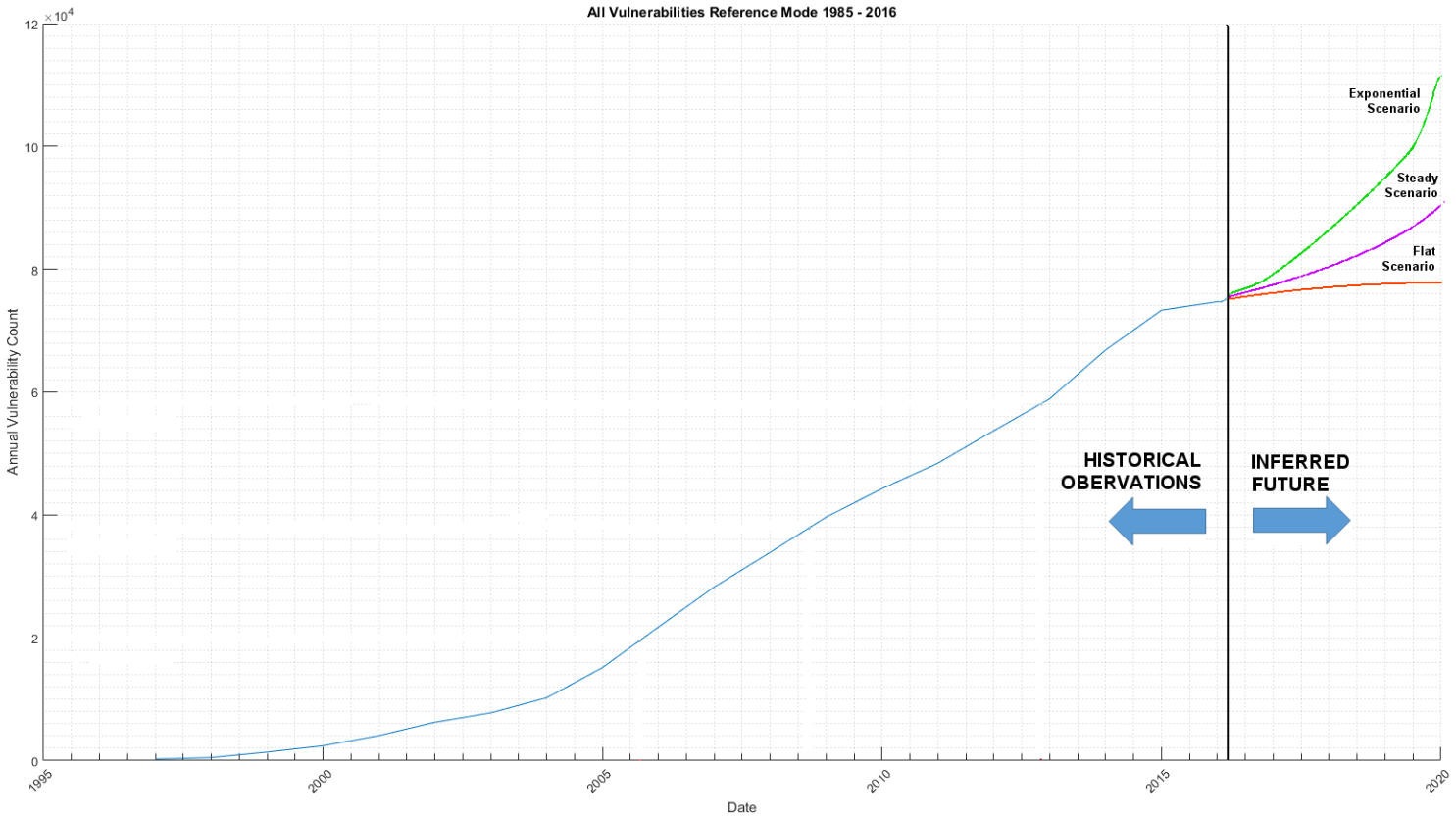


Figure 16 - Total system reference mode Inferred Futures

5.4 State Variable Exploration and Analysis

Regarding the total vulnerability discovery and disclosure reference mode, we can see that there is sigmoidal S-shaped growth exhibited. The VDDS is a system that can be extrapolated to show sets of interactions and variables that represents a real-world phenomenon (Giordano, 2009, p.53). Therefore, extracted from the five identified themes are nine state variables that represent the behaviours of a sub-system within the VDDS. The nine variables range from sentiment of vulnerability discoverer and they feel toward vendors, to the potential reward for discovering and disclosing vulnerability. In keeping with the exploratory stance of this research the variables were explored and characterised in both time and space. This exploration draws on several statistical techniques and shows the descriptive statistics of collected data, such as mean, variance and shape.

5.4.1 State Variable A - Researcher sentiment

As outlined previously there are specific points within the disclosure process where vulnerability discoverers must make a choice between full, direct coordinated or paid-coordinated disclosure. These choices are influenced by both the personal philosophy of the discoverer and previous lived, or more importantly recited, experiences of disclosure. The recited experiences are documented and cited when discoverers solicit advice from the community on which disclosure approach to take. This advice is recorded within social media platforms or forums and provides insight into the attitudes, experiences and outcomes of disclosures. Sentiment is represented by a positive or negative number and is based upon the words and the context they used when engaging in a discussion about a topic (Easley and Kleinberg, 2010). In the context of the VDDS discourse ranges from what the best course of action is, to how vendors treat discoverers once a vulnerability has been uncovered.

A range of techniques and tools exist to analyse text, however online discourse is unique and therefore requires a particular analysis technique (Baccianella et al., 2010; Esuli and Sebastiani, 2006). The chosen analytical method to produce

measures of sentiment that exists within the VDDS is the widely used online technique known as Sensistrength (Thelwall et al., 2010, 2012; Thelwall and Buckley, 2013). Sensistrength is an opinion mining, or sentiment classification technique, in which a large number of training words have been scored with numbers denoting either a positive, or negative sentiment score. Negative sentiment is scored between -1 and -5 with -1 being not negative, and -5 being very negative. Conversely for positive sentiment, +1 is mildly positive, and +5 is extremely positive (Thelwall et al., 2010). Scores are provided at the end of the assessed sentences within the analysed text, and scores are summed to show an overall score. The version of Sensistrength that is used is v2.2, an example of Sensistrength output is shown in Figure 17 below.

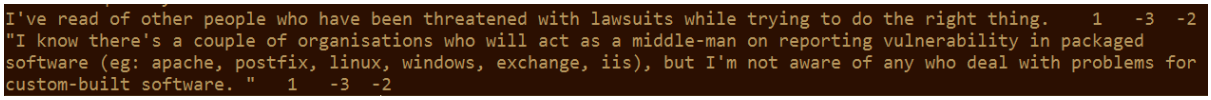


Figure 17 – Sensistrength example output

5.4.1.1 Descriptive Sentiment Analysis Statistics

The data used to generate the descriptive sentiment statistics was extracted from the main Nvivo corpus, and is centred on the social media datasets. Specifically, the social media and full disclosure email archive datasets were used and data were categorised into calendar years, to allow for comparison. However, due to the varying size of datasets a meaningful basis for comparison was impossible to calculate. This variation is due to the large variation of collected sentences within data items discussing disclosure experiences with vendors. Consequently, normalised average scores were calculated and analysis performed via Matlab 2016a processing raw sentiment data.

Year	Lines Analysed	Raw Sentiment Score	Normalised Average Score
2010	2257	-112	-0.04
2011	2645	-476	-0.17
2012	4283	-403	-0.09
2013	4463	-698	-0.15
2014	8639	-1301	-0.15
2015	9292	-113	-0.01
2016	5893	-581	-0.09

Table 21 - Data Population, Raw and Processed Sentiment Scores

Table 21 above reveals that there has been significant fluctuation in sentiment toward with the VDDS, with sentiment being negative in all years. Starting in 2010, the normalised average score starts at -0.04, which relative to the rest of the scores is second highest. Sentiment in the following years of 2011, 2012 and 2013 drops in 2011 recording the lowest score with -0.17, recovering in 2012 to -0.09 and falling again in 2013 with -0.15. Sentiment holds steady in 2014 with a score of -0.15, with a rapid recovery in 2015 to 0.01 and 0.09 in 2016. The sentiment exhibited by vulnerability discoverer is, and continues to be negative. The trend of the data suggests that despite efforts from vendors and 3rd parties to introduce new process and incentives to change the perception of vulnerabilities within software this may not have had the desired affects. As such the sentiment toward vendors is consistently negative, despite a shift into almost positive territory in 2015.

5.4.1.2 Reference Mode and Analysis

The sentiment within the VDDS ranges from -0.17 to -0.01, and includes sentiment exhibited by both vulnerability discoverers and interested third parties. Whilst not directly involved within the mechanics of finding vulnerabilities within software, third parties do have an influencing impact upon the discourse that takes part around the system. At present the strength of this influence upon the decision that discoverers take when a vulnerability is found

is unknown. Alongside the range of values which the data exhibits is the shape, or distribution of the data. The shape of the standardised data is difficult to classify using standard normal distributions; however, it is possible to plot values to gain a visual overview of the behaviour. Therefore, the parameter space is taken to be in the range $\{-0.01 \ldots - 0.17\}$ outlined in Figure 18 below.

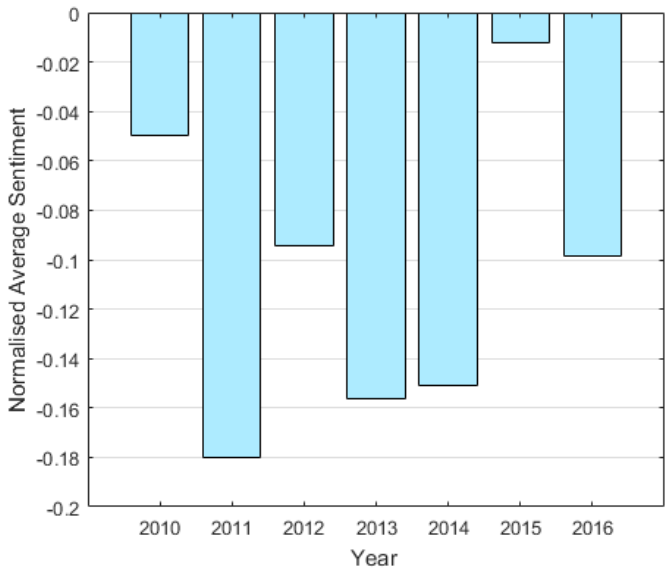


Figure 18 - Reference mode for Sentiment for years between 2010 – 2016 (Normalised Average Scores)

The reference mode shows a constant negative sentiment toward software originators and toward vulnerability discovery in general. As such we can assume that most if not all aspects within the VDDS are negative, or are perceived to be negative. There is a striking exception to this in 2015, where there is almost a neutral level of sentiment -0.01, suggesting that an event occurred to moderate the negative sentiment. At present the cause of this is unknown.

5.4.2 State Variable B - Number of Vulnerability Discoverers within VDDS

As identified in the previous chapters the vulnerability discovery and disclosure system is a sociotechnical system characterised by the activities of people using technology to discovery vulnerabilities. Therefore, we can assume that the number of discoverers that are within the VDDS at any given time is a proxy

measurement of the potential number of vulnerability discoverers within the VDDS that exist. However, there are data source that track the number of submissions whereby vulnerabilities and subsequent exploits are collected and reported. One such aggregation place is a well-known online database known as ExploitDB.com (ExploitDB, 2017). ExploitDB.com is an archive of software exploits created by vulnerability discoverers which accompany vulnerability details typically to prove that the vulnerability exists and can be activated. Alongside the vulnerability and exploit details, the recorded published date of the exploit and crucially author is included within a freely downloadable archive of exploits.

Bringing data of author and submission date together provides a valuable source of data that indicates both the level of activity, as vulnerabilities discoverers may publish more than one exploit proof, and the frequency of those publications.

5.4.2.1 Reference Mode and Analysis

The ExploitDB.com data archive was downloaded and processed on 21st December 2016, and contains 36904 distinct entries. The archive runs between the initial entry and last entry, which are 1st August 1988 and 19th Dec 2016, 13 years and 8 months respectively. The seemingly long and counter intuitive duration of recorded exploits within ExploitDB.com is due to the retrospective addition of historically significant vulnerabilities and exploits. For example details of the first recorded internet worm which exploited a vulnerability within Sendmail has been included (Orman, 2003). However, meaningful publication of exploits was recorded from April 2003, the establishment date of the platform.

Initially data pre-processing was undertaken to clean and to allow statistics to be calculated. To ascertain the number of unique vulnerability researchers which exist within the VDDS duplicates entries were removed from the data, with recorded data of the first published exploit noted. With the removal of duplicate entries (exploit authors) this was reduced to 7843 unique authors. Table 22 below gives a summary of the descriptive statistics for the entry of vulnerability researchers into the VDDS.

Statistic	Values
Total Number of unique vulnerability researchers	7843 (as of December 2016)
Total Number of Exploit Publications	36904
Date Range	Aug 1998 – Dec 2016
Mean exploits Published (Per day)	2 (1.5 rounded up)
Median exploit Published (Per Day)	2
Mode Entries Published (Per Day)	1

Table 22 - Summary Descriptive Statistics: Variable B

To investigate the historical behaviour of the state variable a cumulative frequency graph was plotted. Inspecting Figure 19 we can see that the number of recorded researchers starts slowly then starts to accelerate around January 2002, prior to the setup of ExploitDB.com. The acceleration continues linearly until Feb 2011 where a slowdown begins to occur. The rate of researchers registering and submitting exploits to ExploitDB.com ranges initially from 6.2 per week, rising to 10.3 during the linear phase then reduces to 6.0 per week. Given the large number of users submitting in the linear phase, the data tends to be skewed in a positive manner toward the end of the time with a value of 0.183. Clearly this is only a snapshot into the potential number of vulnerability researchers that exist, and can potentially have a significant bias within it. However, this gives us a good indication of the number of vulnerability researchers that have reported vulnerabilities and exist on the ExploitDB.com platform. It is therefore reasonable to use this measure as a proxy.

The mean number of entries per day was calculated to show that 2 entries per day are added to ExploitDB.com, with the maximum number submitted on one day as 19. The median value is again 2 per day, with the mode, or most common number is a single researcher entering the market per day. Participants enter the VDDS at a mean rate of 2 per week (rounded up from 1.9). This rate is not constant and varies over the dataset with extreme values of 19 and 15 being recorded in Sept and Dec 2004 respectively.

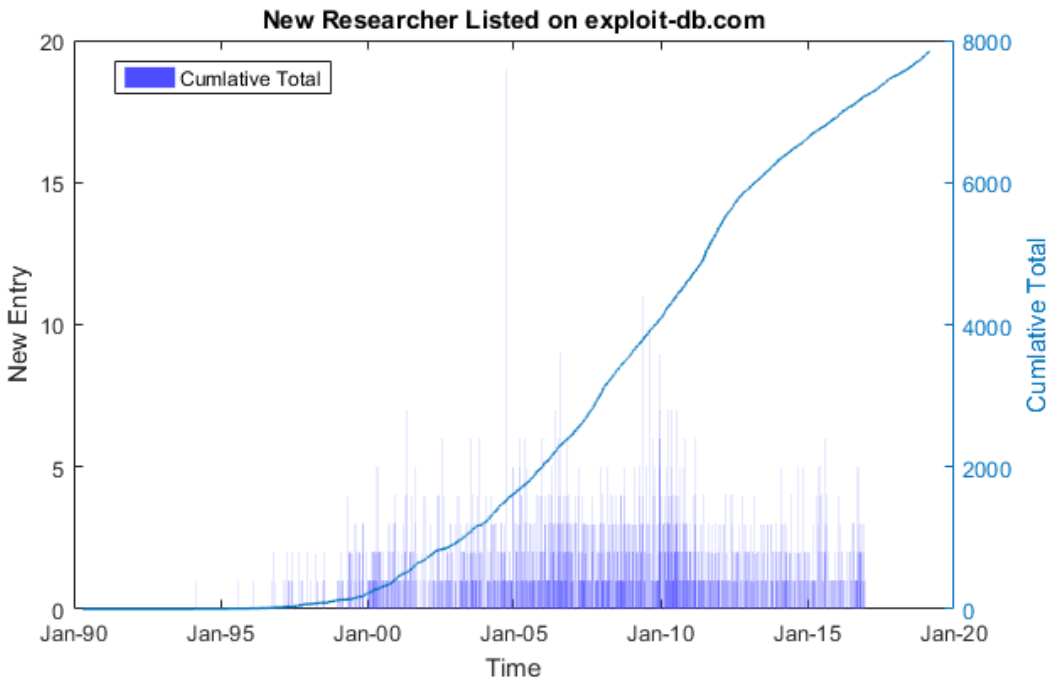


Figure 19 - Cumulative Frequency Curve and Histogram for New Discoverers Entering into the VDDS

Vulnerability researchers enter the VDDS and submit vulnerabilities and exploits to ExploitDB.com at a mean rate of 2 per week. Figure 20 below shows a bar plot with a steady progression and increase of new entrants into the VDDS peaking between Aug (228 entrants) and Nov (226 entrants) 2006 and again in Jan 2010 (283 entrants). Entrants then reduce in number rapidly to 39 at the end of the collected data in Dec 2016. The behaviour of entrants into the VDDS show an increase and then decrease of discoverers over a period of 15 years. There are potentially two processes resulting in the behaviour observed within Figure 20, new entrants onto the ExploitDB.com platform, and existing users of the platform. Activity of the vulnerability is explored further within section 5.4.8 – discoverer activity.

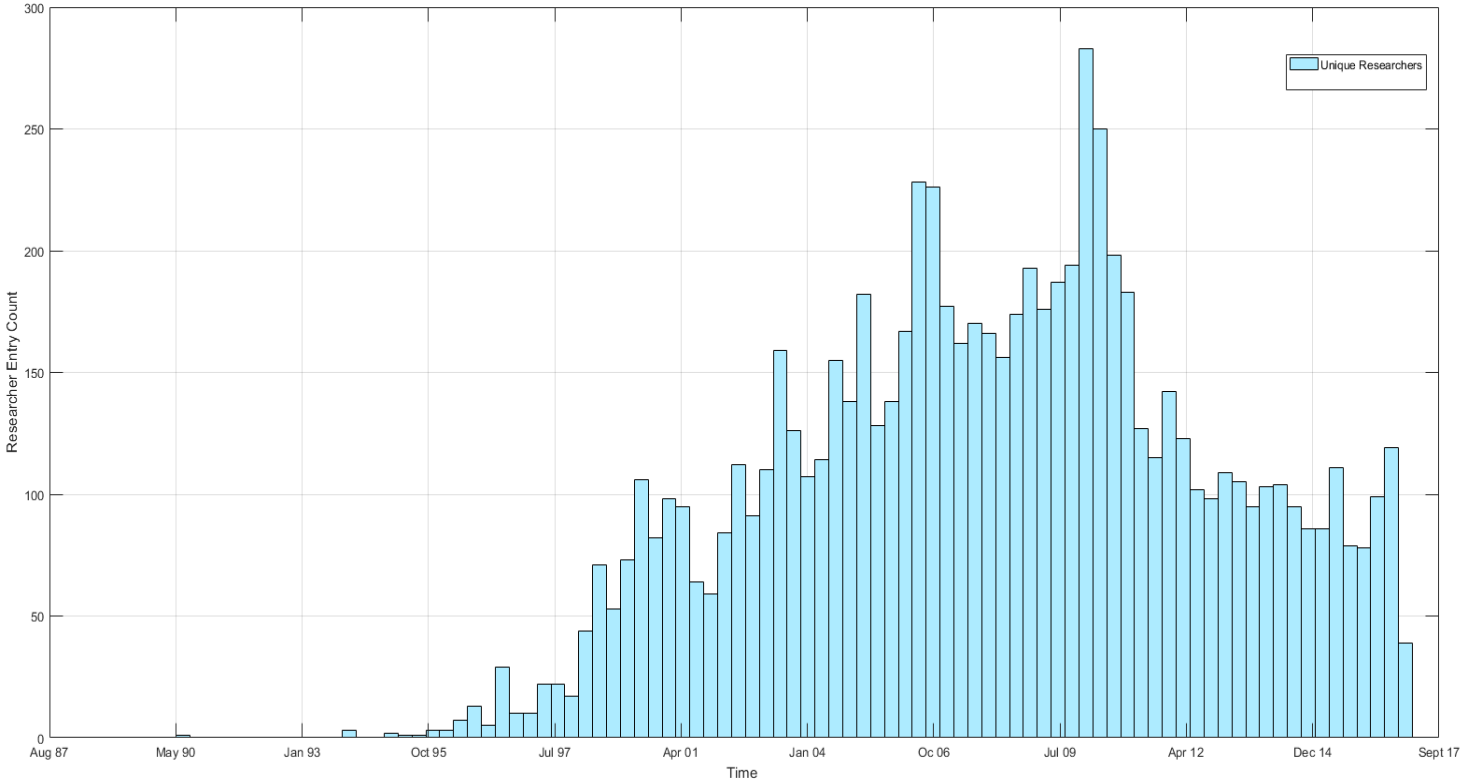


Figure 20 - Reference Mode Plot for New Discoverers Entering the VDDS

5.4.3 State Variable C - Vulnerability Removal Rate

Removal of vulnerabilities from software is analogous to the software debugging process that is used to remove non-security related bugs (Klein, 2011, p.5). The debugging process comes in two distinct yet related forms, perfect and imperfect debugging (Pham, 2000, p.120). As previously discussed many studies state that the number of vulnerabilities that have been discovered and disclosed within software follows a sigmoidal growth pattern (Alhazmi et al., 2005, 2007; Woo et al., 2011). Therefore we can conclude that the number of vulnerabilities that can be discovered within the software declines over the lifetime of the software (Woo et al., 2011). In the seminal work published by Alhazmi and Malaiya (2005) they state that the perfect removal of vulnerabilities from within software systems is the key issue within software development (Alhazmi et al., 2005). Alhazmi and Malaiya (2005) go on to define a measure to assess the removal rate of vulnerabilities from within software as a measurement of vulnerability density. Vulnerability density is measured over time and calculated as the number of vulnerabilities per unit size of code, made up of 3 parts; *Known Vulnerability Density* (V_{KD}), size of the software (*S*), measured in lines of code and reported vulnerabilities within the system (V_K) (Alhazmi et al., 2005). Known vulnerability density is a simply calculated and given by:

$$V_{KD} = \frac{V_K}{S}$$

Equation 4 - Known vulnerability density (Alhazmi et al., 2005)

The equation above assumes that the size of the code with the example, Microsoft Windows NT 4.0, cited in Alhazmi and Malaiya (2005) is static. Initially the number of known (reported) vulnerabilities within the software at time 0 (zero) is 0 (zero) as no vulnerabilities are known. This initial state however does not take into consideration internal quality assurance processes, and removal of known vulnerability prior to release. From time 0 (zero) the number of known vulnerabilities increases, and therefore the known vulnerability density (V_{KD}) for the total software package rises accordingly. Plotting the time of occurrence of

discrete V_{KD} events allows us to build a picture of the increasing, and eventual decreasing vulnerability attack surface of Windows NT 4.0

5.4.3.1 Reference Mode and Analysis

Figure 21 below shows vulnerability discovery as density per month, showing individual density events (i.e. vulnerabilities being discovered per month) over a 13-year period. The density events are assumed to incorporate the removal of vulnerabilities from the software, and in our case the software used to illustrate the concept Microsoft Windows NT 4.0.

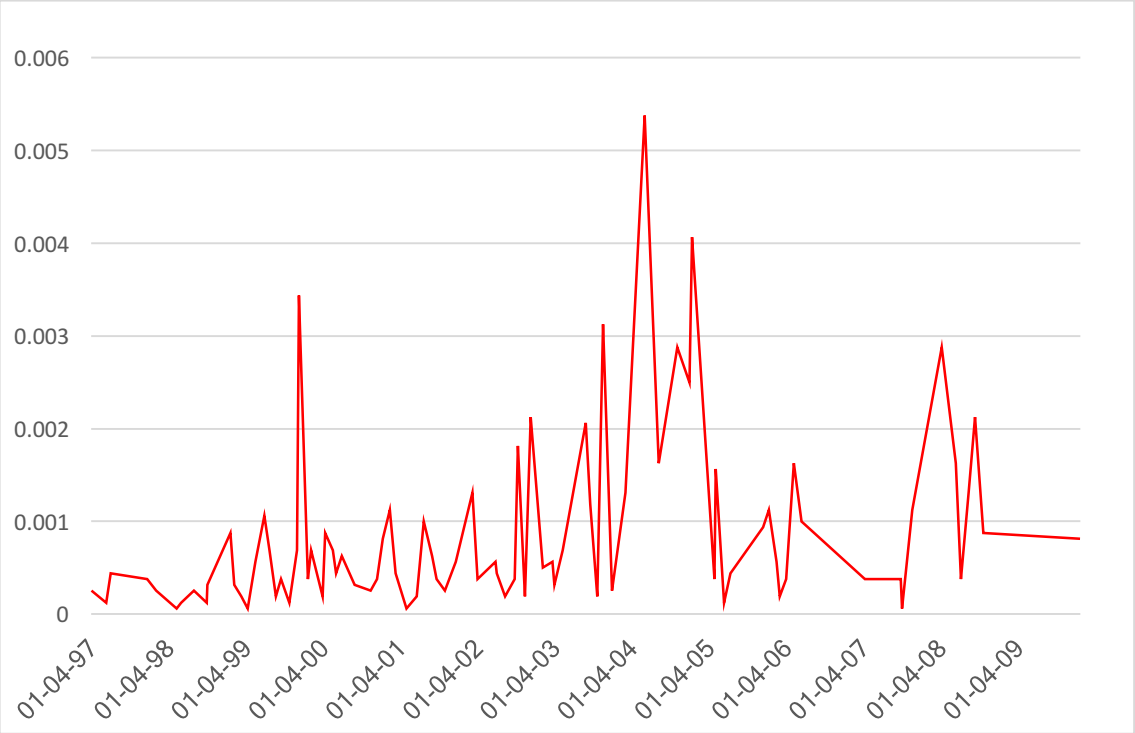


Figure 21 – Observed Vulnerability Discovery Density over Time Windows NT 4.0

Consequently, an estimate of the number of vulnerabilities that are still present within a software system can be calculated. To estimate the quantity of vulnerabilities residing within software we can employ an approach that is used within software and reliability engineering. Known as survival analysis this technique is used to identify the occurrence and timing of events (Allison., 2010, p.1) and is used to understand data that is both censored, and time dependant (Allison., 2010, p.4). We can therefore use survival analysis techniques to both estimate the quantity of vulnerabilities that remain within the software and

estimate when this may occur. The survivor function is defined below where T is the time of an event:

$$S(t) = \Pr\{T > t\} = 1 - F(t)$$

Equation 5 – Survivor Function (Allison., 2010, p.15)

In the case of Windows NT 4.0 the data is right censored from Jan 2010 when recording of vulnerabilities disclosures stopped and Microsoft declared Windows NT 4.0 end of life. Survival curves, or more accurately in our case, vulnerability discovery curves are calculated by observing the time of failure (t), number of failures at time (d), and the estimated number of remaining vulnerabilities (r). The probability of detection $S(t)$ is calculated as the inverse function of the hazard rate (number of vulnerabilities detected). For example, at time t_2, the hazard rate for windows NT 4.0 is defined as:

$$S(t_2) = 1 - \left(\frac{16}{1208}\right) = 0.989$$

Equation 6 – Survivor Calculation for Windows NT 4.0

Showing that at time t_2 the probability of vulnerability detection is 0.989 or there is a 98.9% chance of a vulnerability being detected within Windows NT 4.0. Table 23 below shows the probability of finding vulnerability within Windows NT 4.0 during the course of the software lifespan.

Using Windows NT 4.0 as an example of survival function applied to software vulnerability we can calculate the following information. At the midpoint of the software lifetime (2795 weeks), we can calculate that the probability of vulnerability occurrence is Pr(0.52), showing that there is a 52% chance of discovery of a further vulnerability being detected. Alongside this we can also see that within quartile 1 (0 and 1504 days) there is between 100% and 81% chance of discovering vulnerability. Within Quartile 2 (1504 and 4020 days) there is between 81% an 17% chance of discovery, and finally within quartile 3 (4020 and 5499 days) there is between 17% and 0% chance of vulnerability discovery. All values also attract a 95% confidence bound, see Figure 22.

	Time (weeks)	Number of Detection Vulnerabilities (d)	Number of Estimated Remaining Vulnerabilities (r)	Probability of Vulnerability Detection $S(t)$
1	0	1	1224	1
2	458	16	1208	0.989
3	498	1	1207	0.978
4	526	1	1206	0.967
5	562	4	1202	0.956
…	…			…
89	4678	21	82	0.053
90	4709	34	61	0.043
91	4731	14	27	0.021
92	4768	13	13	0.010
93	5499	21	0	0

Table 23 - Life Table for Windows NT 4.0 (Source: Author)

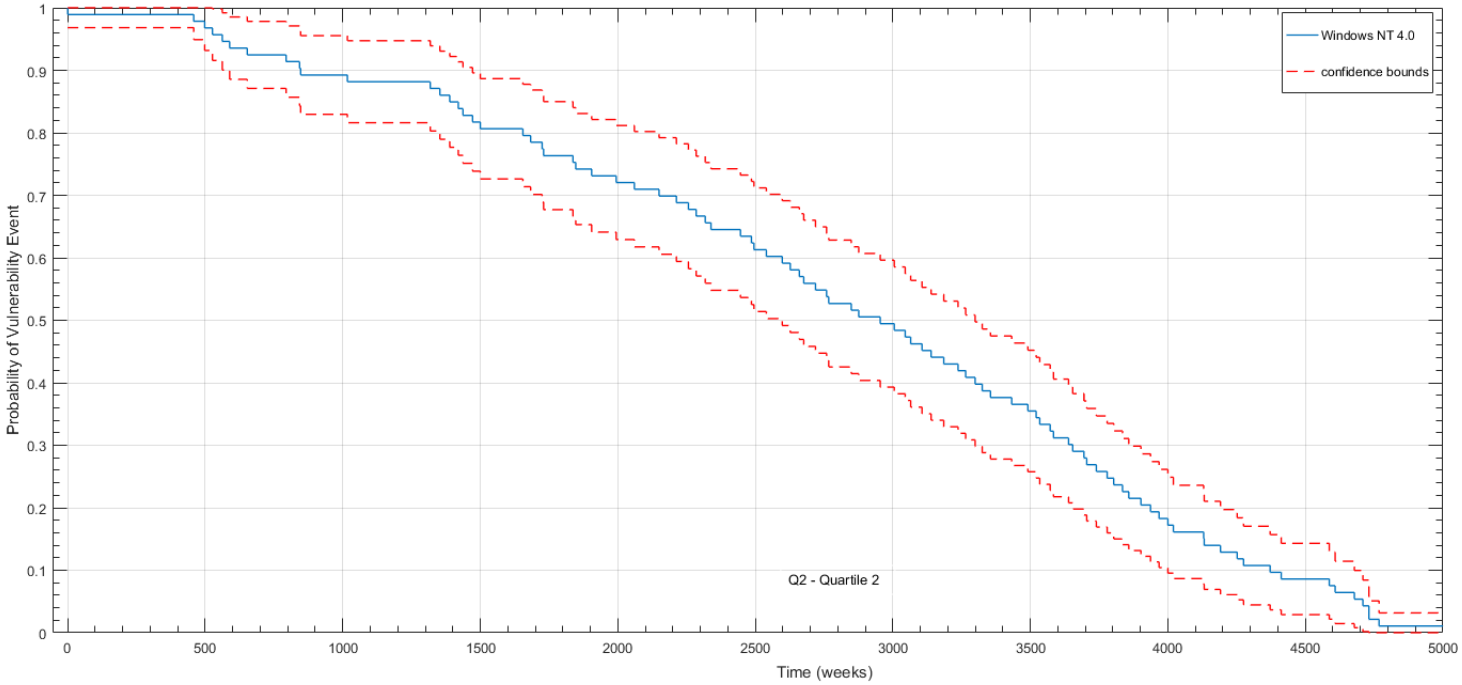


Figure 22 - Reference Mode Windows NT 4.0 Probability of Detection Curve S(t) with 95% confidence bounds

5.4.4 State Variable D - Full disclosure and Coordinated Disclosure Ratio

As stated specific choices are required throughout the discovery and disclosure process, from choosing to find a vulnerability to which disclosure route to take. The route of disclosure is a higher order variable, linked to both the disclosure sentiment and the monies that one may earn if the paid coordinated disclosure route is chosen. Given the stark difference in these choices, an understanding of the current ratio between full disclosure verses coordinated disclosure is an important indicator of how sentiment and rewards are regarded. The measurement of this ratio can also provide a weighting as to which choice, or which is the more probable disclosure route than the other to be chosen that can be included in any subsequent model.

To ascertain the ratio of full verses coordinated disclosures a measurement of the quantities that have been disclosed via one or the other route is required. To calculate the ratios data was collected from the openbugbounty.org repository (OpenBugBounty, 2017) platform between 7th Sept 2015 and 3rd Jan 2017. Openbugbounty.org is a community operated platform for submission of web application based vulnerabilities where vulnerability discoverers can choose to disclose vulnerabilities in one of the previously discussed methods. Openbugbounty.org has made available the number of vulnerabilities that have been submitted to the platform and the number of full vs coordinated disclosures - Table 24 below shows an overview of the collected data.

It is worth noting that a discrepancy exists between the officially recorded vulnerabilities from the NVD and the reported vulnerabilities from Openbugbounty.org. This is due to the fact that vulnerabilities reported on NVD are major issues that have significant impact, whereas Openbugbounty.org vulnerabilities are focussed upon web applications, and platforms. However, the ratio of full vs coordinated disclosure is still a valuable proxy for the state variable.

Date	coordinated	Full	Total	Raw Ratio (Coord:Full)	Percentage of Coordinated	Percentage of Full
07-09-15	1930	29250	31180	2:30	1930/31180 = 0.061 (6%)	29250/31180 = 0.938 (94%)
03-11-15	5223	37182	42405	17:124	12.3%	87.7%
07-02-16	14779	55306	70085	148:553	21.1%	78.9%
06-03-16	13607	59617	73224	136:596	18.6%	81.4%
19-03-16	12123	63079	75202	121:630	16.1%	83.9%
11-05-16	7440	75290	82730	8:75	9.0%	91.0%
18-05-16	9662	75898	85560	48:380	11.3%	88.7%
03-06-16	10094	77919	88013	100:780	11.5%	88.5%
05-06-16	10220	78107	88327	102:780	11.6%	88.4%
20-06-16	10377	79630	90007	103:796	11.5%	88.5%
12-08-16	38121	93103	131224	381:931	29.1%	70.9%

Table 24– Coordinated vs Full Disclosure Ratios From Openbugbounty

N.B.The vulnerabilities totals vary between data points due to vulnerabilities being moved status from vulnerability disclosure choices at the request of the researcher, normally due to inaction of software vendor.

5.4.4.1 Reference Mode and Analysis

The total number of vulnerabilities that were disclosed via the Openbugbounty.org platform was 131224 over a 11 month period, between September 2015 and August 2016. The number of vulnerabilities that were disclosed via the two differing routes is significantly biased toward the full disclosure route. This could be due to a number of factors including the disclosure posture of the membership of Openbugbounty.org, i.e. it is an open platform with no monetary reward operated by the Community of discovers. The discoverers who chose to disclose on the platform do so for kudos therefore have no incentives to coordinate disclosure other than for good ethical reasons.

Figure 23 below shows a comparison between full and coordinated disclosure. We can see the distinct bias toward full disclosure with a 15 to 1 ratio in Sept 2015, however this ratio begins to close toward the end of the data collection period.

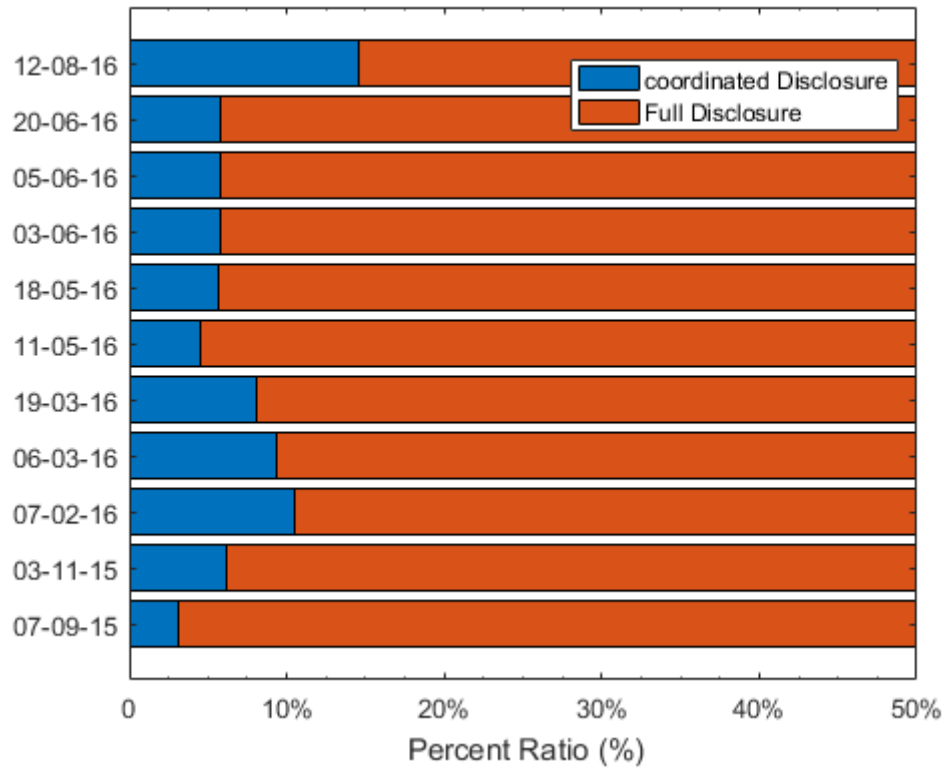


Figure 23 – Ratio of Full Disclosure Vulnerability Vs Coordinated Vulnerability

5.4.4.1.1 Analysis and Data Limitations

Inspecting Figure 23, we can see that the ratio of vulnerability disclosures between full and coordinated is heavily dominated by full disclosures, with percentages of over 75% consistently disclosed via the full disclosure route. This is suggested to be predominantly a feature of both the platform, and the nature of the vulnerabilities that are being disclosed. Openbugbounty.org is a specialist web vulnerability disclosure platform, and as web vulnerabilities are suggested to be ubiquitous and of low value they are disclosed freely open (OpenBugBounty, 2017). However, we can use the ratio as an indicator of the propensity for discoverers to disclose in a full or coordinated way. One significant limitation of the data that has been collected during a small time

period, 11 months. This is due data not being readily available in a public and complete form. However, simple inferences can be drawn from the vulnerability ratio, and extrapolated within the VDDS model.

5.4.5 State Variable E - Vulnerability Disclosure Interaction Time

Vulnerabilities are critical to the risk profile of any organisation. Nevertheless any impact which may result from vulnerabilities is not realised until exploited by a threat actor (Marconato et al., 2012). As such, the time that is taken between the details of the vulnerability being disclosed and a patch or mitigation being created is critical. The time that is taken to fix a vulnerability is linked to the identified meta theme of time, influencing the behaviour of the VDDS as a whole. Coordinated disclosure encompasses both the giving vulnerability information to the vendor freely and the paid version, known as paid for coordinated disclosure. Each approach provides time to software vendors to produce a fix to the issue within the software, and typically a policy outlining any constraints accompanies the disclosure. These polices in general state that if there is inaction on the part of the software originator then the details of the vulnerability will revert to full disclosure and details will be published.

Source	Data Source (Academic Journal, Blog, Commercial Report, Book, Broker Platform)	Mean Time for Patch Availability (days)
(Arora et al., 2010)	Academic Journal	168
(Arora et al., 2005)	Academic Journal	51
(Li et al., 2007)	Academic Journal	52
(Marconato et al., 2012)	Academic Journal	107
(Neff, 2016)	Blog	113
(Beattie et al., 2002)	Academic Journal	64
(Moore et al., 2010, p.94)	Book	44
(Manzuik et al., 2007, p.7)	Book	120

Table 25 – Literature Time Delays Sources

The figures given in Table 25 above show a wide range of interaction lengths, with the weighted mean number of days across all sources calculated as 88.5, with the minimum time reports as 1 and max being 3904 days (Arora et al., 2010). The Figure of 88.5 days provides a measure of the time Microsoft takes to create a fix, and the interaction between software originator and discoverer. It also provides a time period for the vulnerability to transit the vulnerability process from discovered to patch creation and removal of the vulnerability

5.4.5.1 Reference Mode Analysis

In addition to the descriptive statistics gathered from academic papers, empirical data was collected from the HackerOne vulnerability broker platform and the full disclosure email archive (Hackerone, 2016; Seclists.org, 2002). In the case of HackerOne entries indicate the length of time that has elapsed from initial submission of the vulnerability details, to the public and coordinated disclosure of that vulnerability. In the case of the full disclosure email archive, process steps and the date they occurred are also included within the data describing the interaction steps with the software originator.

HackerOne vulnerability disclosure data collected between 6th Nov 2013 and 24th Nov 2015. The range of time it takes the vulnerability disclosure process to complete (the public release of vulnerability details) ranges from 0 day to 619 days. The mean duration (indicated by the red diamond) for a vulnerability to transit the process is 77.4 days, with 0 days being the most frequent occurrence, indicating that a relatively large percentage of 5.04% (33 occurrences) was fixed and publicly disclosed on the same day. When compared to the mean value, the median value of 39 days shows that the data is very positively skewed with a value of +2.49.

Sample size	**654**
Mean	**77.4**
Median	**39**
Mode	**0**
Range	**0 - 619**
Standard Deviation	**102.8**
Skewness	**+2.49**

Table 26 – Descriptive Statistics for HackerOne Duration (Left)

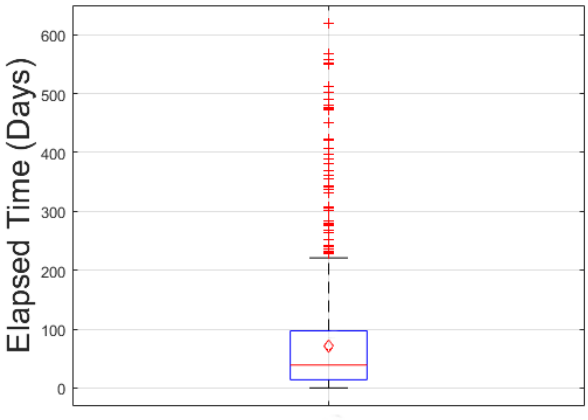


Figure 24 – Boxplot Showing Descriptive Statistics (Right)

Given the heavy positive skewness (+2.49) of the data we can calculate that almost half 47% (308) the vulnerabilities total number (654) of the vulnerability disclosures that are disclosed via the disclosure platform HackerOne are dealt with quickly and within the HackerOne policy standard of 30 days (HackerOne, 2017).

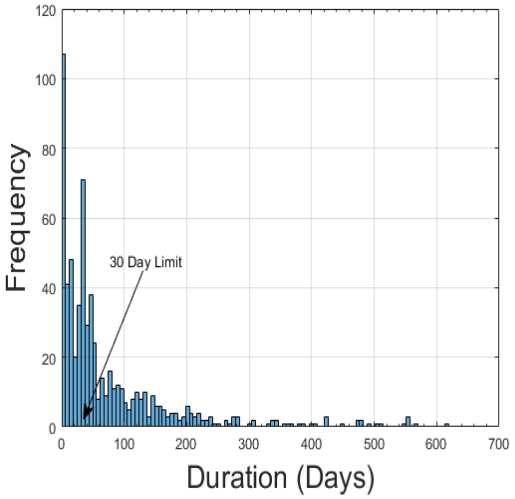
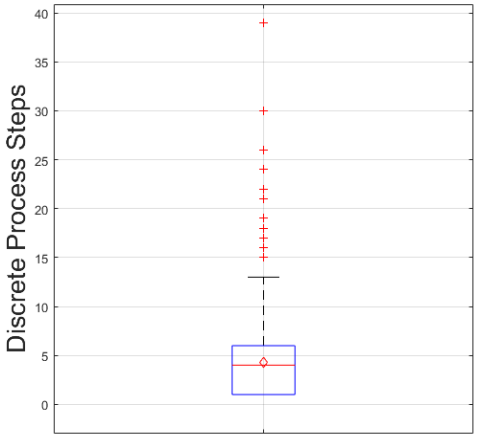


Figure 25 - Skewness of HackerOne Disclosure Time

5.4.5.2 Full Disclosure Email Archive

Additionally, 2745 emails were processed between 18th Feb 2015 – 17th March 2017 processed into 998 entries. The duration of the vulnerability disclosure

process (the public release of vulnerability details) ranges from 0 days to 1612 days. The mean duration (indicated by the red diamond) for a vulnerability to transit the process is 98.2 days, with the median value of 30 days shown in Figures 26 and 27. The duration of 30 days corresponds to an adopted policy from the users of the full disclosure email list for full public disclosure to occur if inaction or unsatisfactory events occur.



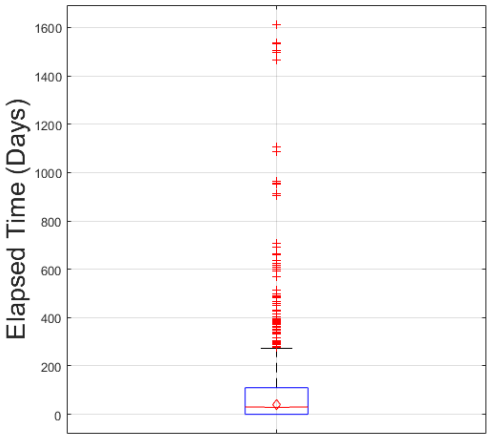


Figure 26 (Left) - Discrete Process Steps (Email Archive) Figure 27 (Right) – Elapsed Time (Email Archive)

The median value of 30 days shows that the data is positively skewed with a value of +4.39, showing that the majority of the data points are clustered around the 30 day point. Alongside this, a number of outliers exist which show a longtail of durations. Given the heavy positive skewness of the data we can calculate that over half of the data 72.1% is located to the left of the mean (98.2 days), and that 38.0% the vulnerabilities disclosed via the full disclosure mail list are dealt within industry standard of 30 days. As shown in Figures 27 and 28 several outliers exist, however the quantity is relatively small in comparison to the rest of the data.

Disclosure Duration		Process Steps	
Sample size	**998**	Sample size	**998**
Mean	**98.2**	Mean	**4**
Median	**30**	Median	**4**
Mode	**0**	Mode	**1**
Range	**0 - 1612**	Range	**1 - 39**
Standard Deviation	**193.6**	Standard Deviation	**3.79**
Skewness	**+4.39**	Skewness	**+2.58**

Table 27 – Descriptive Statistics for Duration

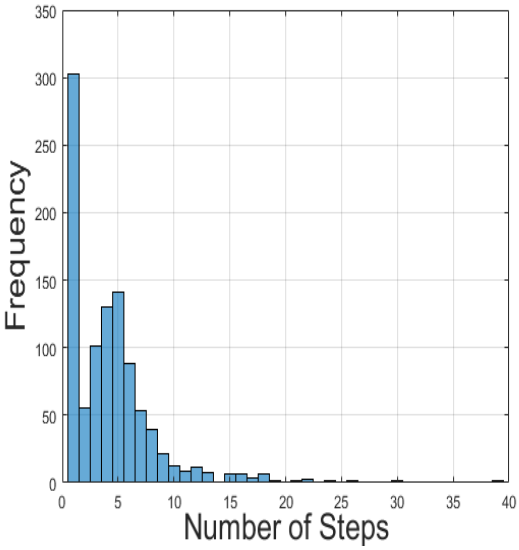


Figure 28 - Number of Disclosure Steps

In addition, a useful metric is the number of interaction steps it takes for the vulnerability disclosure process to complete the transaction - the public release of vulnerability details. The number of steps ranges from 1 to 39, with the mean number of steps (indicated by the red diamond) for a vulnerability to transit the process is 4 steps, with the median value of 4 days being the most frequent occurrence. When compared to the mean the median value of 4 days indicates that the data is positively skewed with a value of +2.58, showing that a typical transition through the disclosure process takes 4 interaction steps. Given the positive skewness of the data we can calculate that over half of the data 67.6% is located to the left of the mean (4 steps), and that vulnerabilities disclosed via the full disclosure mail list are dealt with in 4 interaction steps of less. The large peak shown in Figure 28 indicating one step is the use of the full disclosure route, with one step – the announcement of the vulnerability to the public.

5.4.6 State Variable F - Monitory reward

A recent phenomenon that has impacted the VDDS is the creation and adoption of schemes that allow discoverers to sell and receive recompense in the form of a gift or money for responsibly disclosing vulnerabilities (Bradbury, 2007; BugCrowd Homepage, 2017; Hackerone, 2016; OpenBugBounty, 2017; ZDI, 2017; Zerodium, 2017). The ability to receive recompense for the effort that has been expended is an artefact within the identified themes, and features heavily in the discourse that surrounds the VDDS. At present there are two preeminent open vulnerability platforms that operate alongside corporately run schemes from Oracle, Microsoft and Google (Google, 2017; Microsoft, 2017; Oracle, 2017) known as Bugcrowd and HackerOne. Each of these platforms allows discoverers to submit vulnerabilities into disclosure programs and dependent upon the nature and criticality of the vulnerability, monetary rewards are presented to the discoverer. Unfortunately statistics on monetary reward are not made public by Bugcrowd, therefore only data from HackerOne was collected. Data was downloaded from HackerOne.com on the 17th December 2016 and processed using Matlab 2016a. The average cost for a vulnerability which is recorded via the Bug Bounty programme with for all software and companies registered on HackerOne is $1012.97, with a standard deviation of $2160.10 outlined in Table 28 below.

Reward	Count	Percent
500	328	23.03%
100	196	13.76%
1000	110	7.72%
50	82	5.76%
250	82	5.76%
150	65	4.56%
1500	43	3.02%
200	41	2.88%
300	40	2.81%
3000	33	2.32%
..	..	..
2750	1	0.07%
3137	1	0.07%
3500	1	0.07%
3705	1	0.07%
5040	1	0.07%
5500	1	0.07%
6000	1	0.07%
6500	1	0.07%
9000	1	0.07%
12500	1	0.07%

Summary Statistics	Cost (USD)
Mean	$1012.97
Min / Max	$1 – $20,000
Median	$500
Mode	$500
Standard Deviation	$2166.10

Table 28 - Table Ranked by Most Frequent Rewards Sum

5.4.6.1 Reference Mode and Analysis

The median value of a vulnerability traded on the HackerOne platform is $500 with mode also calculated as $500. The skewness of the data is heavily positive, with a skewness of +5.04, showing that most of the bounties paid are equal to or below $1000 accounting for of 1172 of 1424 entries (82.3%).

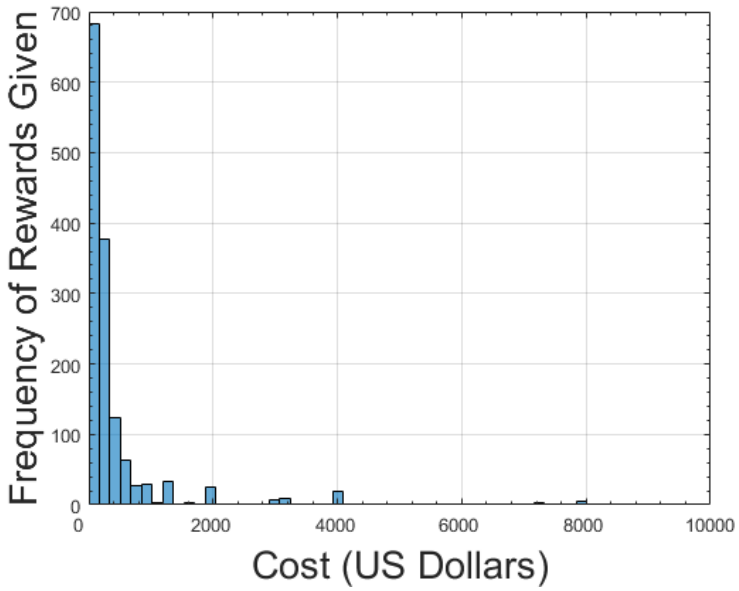


Figure 29 – Frequency of Rewards From Hackerone

5.4.6.2 Microsoft Bounty Hunters: The Honour Roll

In addition to HackerOne data Microsoft publish a partially complete list of bounties that have been paid to vulnerabilities discoverers known as the Honor [sic] roll, and is compiled on a quarterly basis (Microsoft, 2017). Data was accessed from the Microsoft website on the 2nd March 2017. The data contains 61 entries with a price range of $500 to $125,000, median of $7,500 and standard deviation of $27,839.90.

Summary Statistics	Cost (USD)
N = 61	
Mean 2013	$29,838
Mean 2014	$75,000.00
Mean 2015	$20,786
Mean 2016	$15,741.38
Mean 2017	$10,000.00
Mean (All years)	$20109.83
Median (all years)	$7,500
Mode (all years)	$1,500
Standard Deviation	$27,839.90
Min / Max	$500 - $75,000

Table 29 – Descriptive Statistics for Payments

5.4.6.3 The illegitimate Black Market

Alongside the legitimate market where discoverers are paid for disclosure, a black market also exists in parallel to the VDDS (Radianti et al., 2006). The black market is characterised as a place that is used to trade vulnerabilities for illegal purposes and is an unregulated market (Radianti, 2010a). Whilst this research does not consider deeply the mechanisms that exist within the vulnerability black market, we must however consider the influence the market has upon the wider VDDs. Consequently, we cannot completely dismiss the existence of the black market but include any known values that are placed on vulnerabilities within the black market into the construction of the system dynamic model. Many studies have considered the market price for illegally traded vulnerabilities and have provided estimates of those prices (Borgen, 2013; Egelman et al., 2013; Malaiya, 2014; Miller, 2007; Radianti, 2010a; Radianti et al., 2006, 2007a, 2009; Siegel, 2013).

To go further than merely writing about the black-market, one needs to observe the activities within areas of the so called dark web. As such data was gathered from a dark web site known as 'The Real Deal'. The real deal is a vulnerability trading website that provides 0day vulnerabilities for use. The Real Deal is an openly accessible dark web site that requires no credentials and does not require authentication therefore is an open resource and falls within the ethical approval of this research. The Real Deal website was closed in 2016, however archive screenshots were collected on 17th Aug 2015 and show the value of vulnerabilities on sale at that time. Vulnerability prices range from $466 to over $70K for issue that are present in Windows, Linux and Apple software, outlined in table 30 below. Raw data is available in Appendix A.

Vulnerability Type	Advertised Cost (Bitcoin)	Cost (US Dollar)
SQL Injection (OSCommerce)	4.85619985	$1131.49
Microsoft Office 0Day	91.053722836	$21215.51
Adobe Flash 0day	60.70248558	$14143.67
OsCommerce (Sql Injection)	150.000000	$349500.00
Mailbird 0Day (Race Condition)	15.0000000	$3495.00
1Linux 3.13.0-48 (Kernel Panic)	2.00000000	$466.00
Microsoft Internet Explorer 11	35.0000000	$8155.00
Android Webview 0day (Remote code execution)	35.5000000	$8271.50
Wordpress MU (Remote code execution)	4.50000000	$1048.50
Microsoft Word	40.0000000	$9320.00
Firebird	4.85619885	$1131.49
OsCommerce (Sql Injection)	6.07024856	$1414.36
Windows LPE	12.0000000	$2796.00
Microsoft Internet Information Server (remote code execution)	303.51242792	$70718.39
Netis Core Router (Remote code execution)	6.00000000	$1398.00

Table 30 – 'the real deal' black-market costs for vulnerabilities * (BTC to US Dollar was $233 Aug 2015 therefore final conversion is +/- $0.3 due to exchange fluctuations)

5.4.7 State Variable G - Number of Bug Bounty Programmes

Coupled with the value of vulnerability the potential number of markets, of programmes available to vulnerability discoverers is a key part of the VDDS. As stated third party platforms exist to facilitate between vulnerability discoverer and software originator, some like HackerOne are open and transparent, others for example the ZeroDayInitiative do not provide details of their vulnerabilities other than to disclose once a predetermined time has elapsed. As such private initiatives, ran by the companies themselves have been established, and in some cases, monetary rewards. According to Vulnerability Lab, as of 3rd Apr 2017, 495 discrete Bug Bounty programmes exist, each with their specific reward programme (Lab, 2017). Of the 495, 157 offer money for details of a verified vulnerability. Alongside this, low value gifts and public acknowledgement of the discovery is offered.

5.4.7.1 Reference Mode and Analysis

Again, we turn to the HackerOne data for indicating new Bug Bounty programmes entering into the VDDS. Raw data was collected between 7th November 2013 and 31st Mar 2017, with a total of 3424 entries collected and transformed using the Python language to remove duplicates. The transformed raw data resulted in 176 unique programmes in existence on the HackerOne disclosure platform. The platform has recorded a new programme entry into the platform on average every 1.1 weeks. Each programme typically offers a bounty range for differing classification and severity of vulnerabilities. For example, the Uber disclosure policy states that it is interested in:

"Cross-site Scripting (XSS), Cross-site Request Forgery, Server-Side Request Forgery (SSRF), SQL Injection, Server-side Remote Code Execution (RCE), XML External Entity Attacks (XXE), Access Control Issues (Insecure Direct Object Reference issues, etc), Exposed Administrative Panels that don't require login credentials, Directory Traversal Issues, Local File Disclosure (LFD)" (Uber, 2017).

Alongside this bounties of up to $10,000 are paid for critical issues (Uber, 2017). This increase in programmes allows for the increase of coordinated disclosures to happened, illustrated within Figure 30.

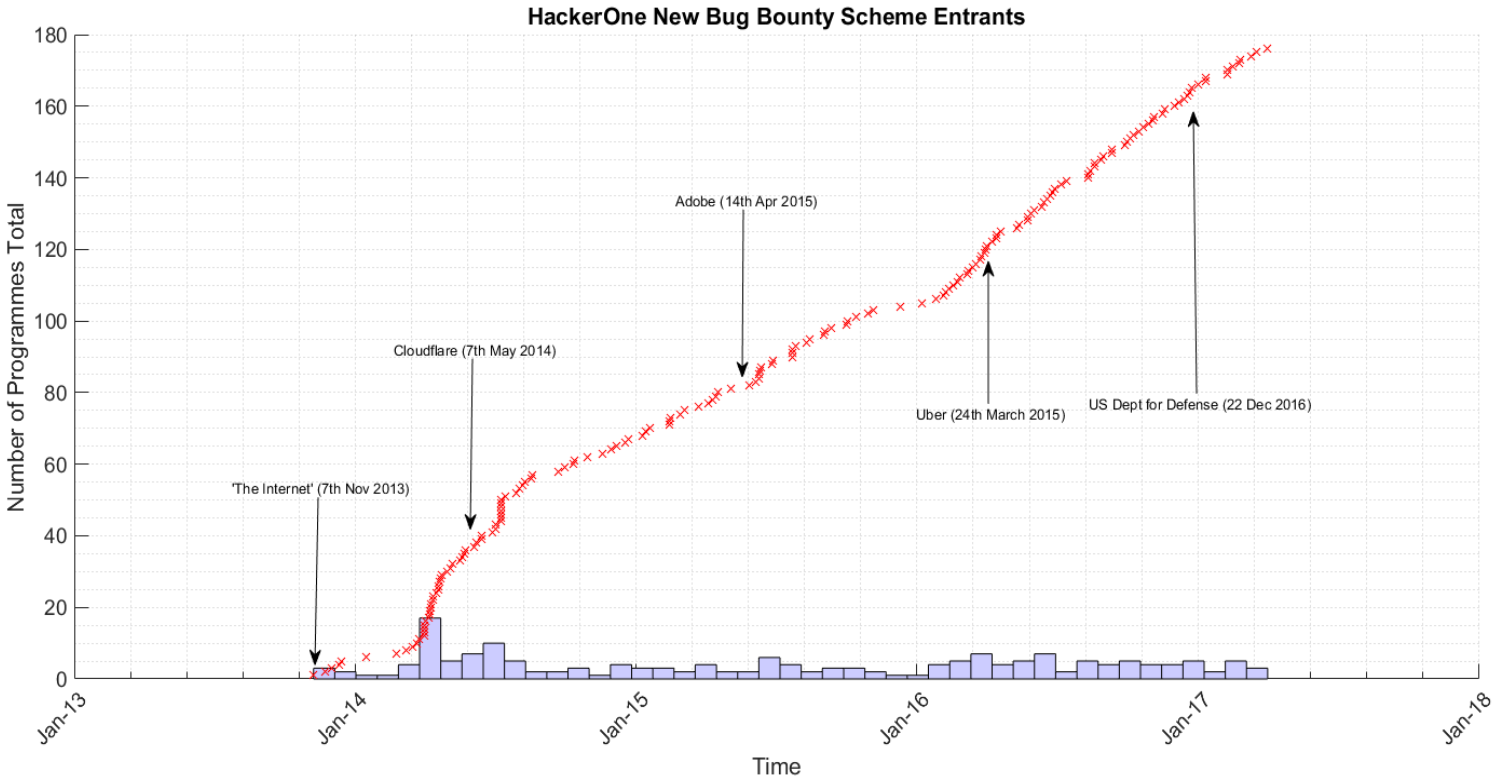


Figure 30 – New Bug Bounty Schemes Timeline

5.4.8 State Variable H - Discoverer Vulnerability Activity

Linked closely to the number of vulnerability discoverers within the VDDS are the discovery activities of the discoverers. To discover a vulnerability, one must expend effort to do so, and have a level of technical competency. The effort that is expended to find a faults and vulnerabilities within software has been extensively studied, with vulnerability discovery effort being considered to be analogous to the testing effort that is needed to test and debug software systems (Alhazmi, 2006; Alhazmi et al., 2005; Holm et al., 2013; Johnson et al., 2016; Pham, 2006; Rescorla, 2005; Sommestad et al., 2012; Wang et al., 2014; Woo et al., 2011). Pham (2006) defines two resources that direct the pace of testing, manpower, including failure and correction identification personnel, and computer time (Pham, 2006, p.482). Consequently, identification and fault correction can therefore be considered to be analogous to vulnerability discovery and vulnerability patching.

To estimate the effort a discoverer expends to uncover a vulnerability we must calculate the frequency of discovery per individual and the time that is taken. A conservative estimate is that a period of 3 months is required to find one vulnerability, whereas other estimates state range $\{1.5 \ldots 3\}$ days dependent upon experience. (Holm et al., 2013; Nom, 2015; Sommestad et al., 2012). An effort model can be considered as a six stage process, with four distinct categories of discoverer, expert, intermediate, beginner and novice, each attracting a number of days to develop a vulnerability and exploit (McQueen et al., 2006). The mean time to discovery for the categories of discoverer range $\{0 \ldots 122\}$ days for expert, range $\{0 \ldots 165\}$ days for intermediate, and in excess of 200 days for beginner and novice (McQueen et al., 2006).

Study / Evidence	Effort Expended (Days)
(Sommestad et al., 2012)	{1.5 … 3}
(McQueen et al., 2006) **(Expert)**	{0 … 122}
(McQueen et al., 2006) **(Intermediate)**	{0 … 165}
(Nom, 2015)	120

Table 31– Summary of Effort Ranges

Given the wide range of recorded durations to discover a vulnerability the assumption is made that only expert and intermediate discoverers will be considered from the McQueen et al., 2006 study.

5.4.8.1 Reference Mode and Analysis

As we are looking to derive the shape and behaviour of key variables from within the VDDS, two key aspects, the rate vulnerabilities are disclosed by researchers, and inter arrival rate are important. The importance of these is highlighted as System Dynamics models aim to show the flows of information or resources around the system, and accurately representing these within the model is key. By understanding the rate which vulnerabilities enter into and exit the VDDS allows the flow of vulnerabilities and disclosure choices to be made. Typically, the process of arrival is described as a queueing system, and specific notation known as Kendall notation is used (Ibe, 2011, p.64). As such we will adopt the same nomenclature, however this is for descriptive purposes only. Arrival rate is defined as per the number of occurrences per unit time, (Ibe, 2011, p.31) and inter-arrival time as the time between occurrences. The unit of time is disclosures per calendar month, 30 days.

Gaining a reliable figure on the number of vulnerability researchers that are within the system and active is difficult, but not insurmountable. As stated achieving a reliable figure on the number of vulnerability researchers that are active within the system is difficult, but not insurmountable. Again, we return to ExploitDB.com dataset which records the submission name and date of the discovered vulnerability. Utilising ExploitDB.com provides a reasonable indication of the number of vulnerability researchers that are active that have

reported vulnerabilities on the ExploitDB.com platform. The activity of vulnerability researchers is a key indicator relating to the number of vulnerabilities that are eventually disclosed. As such the shape of the data is characterised as a series of disclosures, indicating the flow of vulnerabilities within the VDDS. Statistics were calculated showing the mean number of days participants took to discover a vulnerability, which was derived looking at published reports within on the vulnerability database ExploitDB.com. The calculated mean is 256 days, with the median value of 1 day, outlined in Table 32 below. Therefore, the most common type of submitter only discloses one vulnerability to the ExploitDB.com platform in their lifetime, with this category accounting for 61.6% of all submitted reports. Inspecting the data, we can see that there is a heavy positive skew of +4.09, suggesting that the median value of 204 days is a more representative figure for the number of days a researcher is actively discovering vulnerabilities when we normalised and remove the single discoverer records skewing the data.

	Vulnerability Discoverer Name[1]	First submission	Last Submission	Days active	Frequency	Current Probable State
1	**Metasploit**	16-06-08	21-11-16	3080	1450	**Active**
2	**Google_Security_Research**	04-04-13	20-12-16	1356	416	**Active**
3	Luigi_Auriemma	17-12-02	29-06-12	3482	416	Inactive
4	**High_Tech_Bridge_SA**	13-04-10	29-04-16	2208	409	**Active**
5	**anonymous**	01-08-88	13-04-15	9751	359	**Active**
6	LiquidWorm	22-07-08	16-12-16	3069	353	**Active**
7	rgod	21-05-05	11-12-13	3126	333	Inactive
8	indoushka	23-12-09	08-05-14	1597	294	Inactive
9	r0t	23-03-03	14-03-10	2548	257	Inactive
10	ZoRLu	17-02-08	24-09-14	2411	221	Inactive
11	ajann	26-05-06	15-01-09	965	204	Inactive
12	Lostmon	20-05-03	28-03-12	3235	188	Inactive
13	shinnai	11-12-06	23-11-16	3635	176	**Active**
14	Moudi	07-01-09	10-09-10	611	170	Inactive
15	laurent_gaffie	15-09-06	09-11-16	3708	161	Inactive
16	GoLd_M	07-01-07	10-08-12	2042	152	Inactive
17	Kacper	12-05-06	15-06-09	1130	149	Inactive
18	Stack	17-01-08	22-01-10	736	147	Inactive
19	S@BUN	22-01-08	25-11-09	2411	143	Inactive
20	cr4wl3r	03-08-09	24-12-13	1604	130	Inactive

Table 32 – Vulnerability Disclosure Activity Timeline

[1] Groups of researchers exist within the dataset, for example Metasploit, High_Tech_Bridge_SA and Google_Security_Research. These groups are highlighted

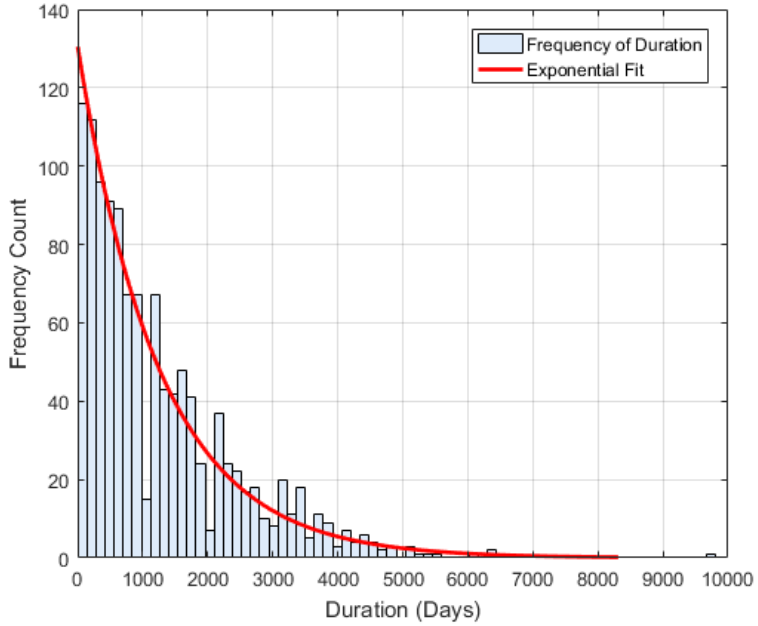


Figure 31 - Amended Frequency of Activity Duration of VDDS Participants (Left)

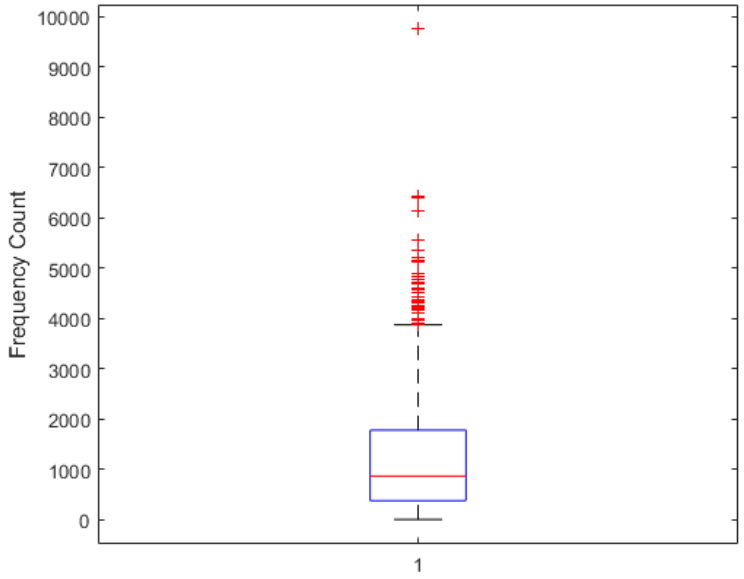


Figure 32 - Amended Frequency of Activity Boxplot (Right)

Using the amended data mean lifetime of a vulnerability discoverer is 1,258 days, with a median of 866 days. Using the frequency of discovery, we can calculate the most prolific vulnerability discoverers find between 416 and 130 vulnerabilities during the active lifecycle. Table 33 below outlines the top 20 vulnerability durations with relative rank, frequency counts, active days and percentage accounted.

	All Data			Single Discover Removed		
Rank	**Days Active on platform**	**Frequency Count**	**Percentage (%)**	**Days Active**	**Frequency Count**	**Percentage (%)**
1	1	4830	61.61	2	50	1.66
2	2	50	0.63	3	40	1.33
3	3	40	0.51	11	38	1.26
4	11	38	0.48	4	34	1.13
5	4	34	0.43	21	34	1.13
6	21	34	0.43	5	30	0.99
7	5	30	0.38	6	30	0.99
8	6	30	0.38	7	25	0.83
9	7	25	0.31	111	23	0.76
10	111	23	0.29	13	21	0.69
11	13	21	0.26	14	20	0.66
12	14	20	0.25	12	19	0.63
13	12	19	0.24	9	18	0.59
14	9	18	0.22	31	18	0.59
15	31	18	0.22	8	17	0.56
16	8	17	0.21	23	15	0.49
17	23	15	0.19	15	14	0.46
18	15	14	0.17	41	14	0.46
19	41	14	0.17	51	14	0.46
20	51	14	0.17	113	14	0.46

Table 33 - Top 20 vulnerability discoverers with relative rank, frequency counts, active days and percentage accounted

5.4.9 State Variable I - Market Share

Alongside the desirability of the vulnerability, the potential attack surface that is subject to exploitation is important (Alhazmi 2005). In the context of the VDDS, this can be expressed as market share of the software, or installed units. As the scope of this research considers operating system vulnerabilities, the collection of operating system specific datasets was carried out, namely Microsoft Windows, Apple OS X and Linux. More specifically, data were collected from three main sources, W3Schools, StatCounter and Wikimedia Foundation and compared to assess the accuracy of the data for the three collected categories of operating system. Data were collected between dates outlined in Table 34 below, and represents the percentage of market the software is used globally.

Dataset	Date Range	Start (%)	End (%)
W3CSchools (Windows)	Jan – 2008 - Feb 2017	82.8	77.50
W3CSchools (Linux)		3.60	5.70
W3CSchools (Apple OSX)		4.40	10.50
Wikimedia Foundation (Microsoft Windows)	Apr – 2009 – Feb 2017	89.50	39.40
Wikimedia Foundation (Linux)		1.49	24.89
Wikimedia Foundation (Apple OSX)		6.05	5.07
Statscounter (Microsoft Windows)	Jan 2009 – Apr 2017	94.42	84.01
Statscounter (Linux)		0.64	1.65
Statscounter (Apple OS X)		3.68	11.32

Table 34 – Market Share Descriptive Statistics

5.4.9.1 Reference Mode and Analysis

Market share data represents the three most popular software products that are used globally. As such, they represent a significant attack surface of the software that is deployed, and per Alhazmi et al., (2005) the most desirable for software discoverers (Alhazmi et al., 2005). Given this assumption, we use market share as proxy of desire to discover, the larger the market share the large the attack surface and desirability. All datasets show a decline in the market share of Microsoft Windows software, from 82.2% to 77.5% for W3Cschools, 89.5% to 39.4% for

Wikimedia foundation and 94.4% to 84.01% for Statscounter over an 8 year time period. Whilst W3CSchools and Statscounter broadly agree (+/- 5%) the Wikimedia data set significantly differs from both W3Cschools and Statscounter. At present the reason for this deviation is unknown. Conversely, both Linux and Apple OSX show an increase over the same time period for all datasets, increasing between 2% and 20% for all datasets over the same time period, 8 years, shown in Figures 33, 34 and 35. Marketshare within the VDDS is a driver of the perceived importance of the vulnerability, and probable attack surface that can be exploited. This therefore has a potential baring on the price that the vulnerability is assessed to attract – the larger the market share, the more valuable the vulnerability.

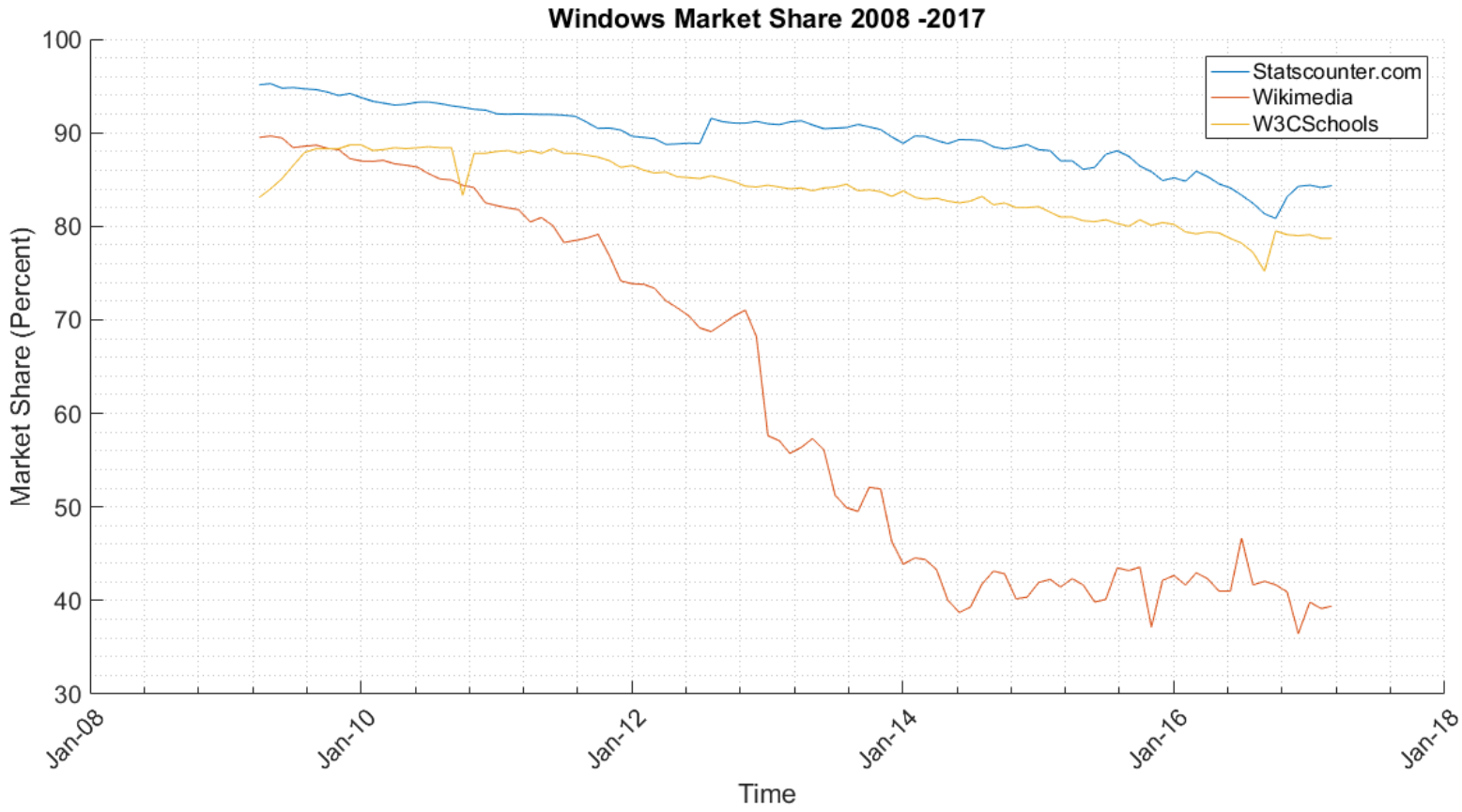


Figure 33 – Windows Market Share 2008 - 2017

Figure 34 – Apple OS X Market Share 2008 - 2017

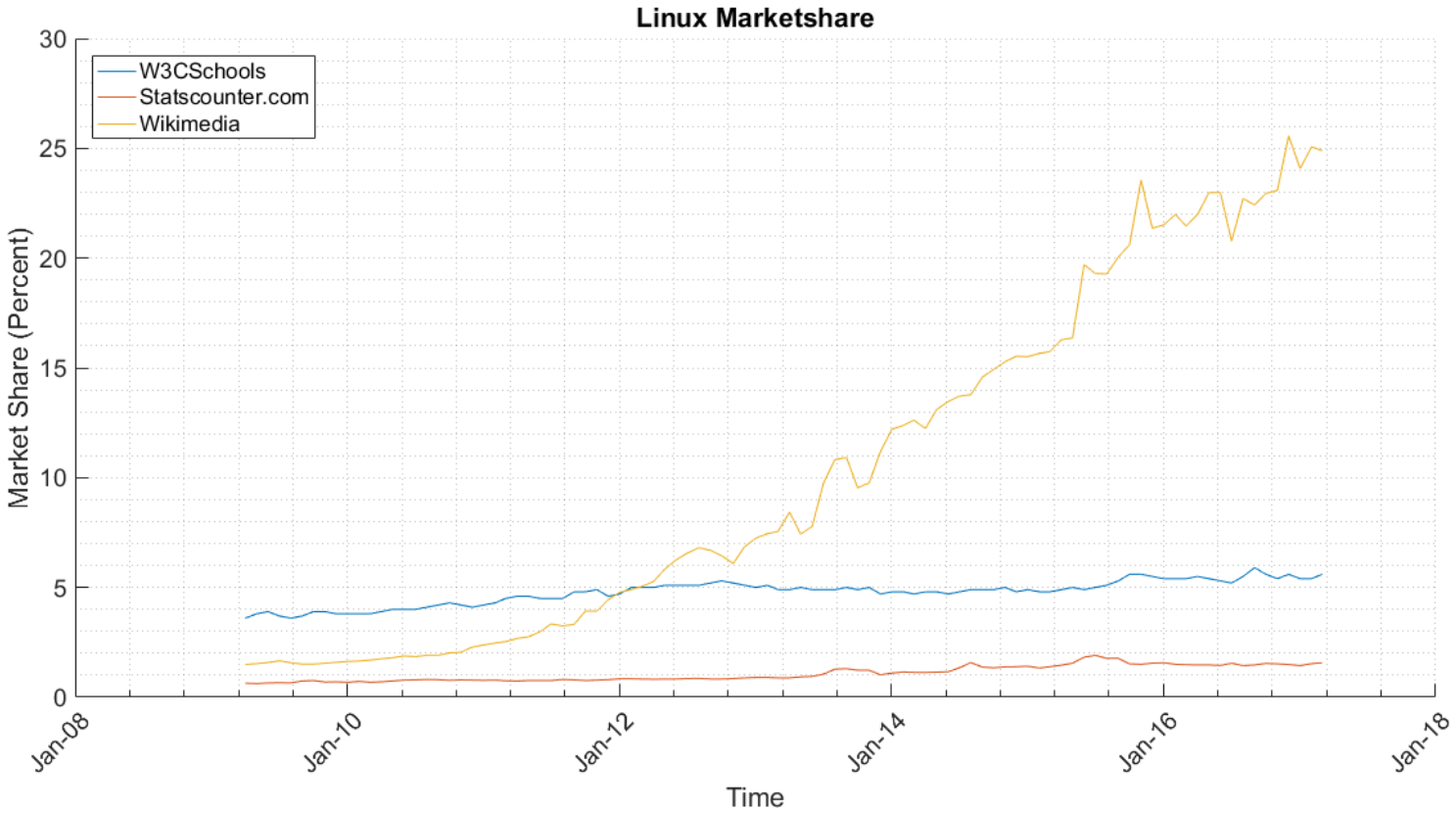


Figure 35 - Linux Market Share 2008 – 2017

5.5 Delays and Time within the VDDS

Delays within the VDDS as stated in wider systems theory are a critical source of dynamic behaviour within a complex system (Coyle, 1977, p.34; Sterman, 2000, p.409; Wolstenholme, 1996, p.15). It is evident that within the VDDS there are several areas which are both shaped by delays, influence the natural order of events, or fundamentally change how the system behaves. Four areas are discovery lag, disclosure submission, and confirmation of vulnerability details between vulnerability discoverer and software originator and the creation of a fix. Moreover, given the importance of the delays within the VDDS an accurate measurement of the magnitude of delays within the VDDS is crucial in understanding their influence.

Delays within System Dynamics are characterised as either material, where a physical or logical exchange takes place (i.e. money) or informational where perceptions or beliefs are adjusted (i.e. sentiment) (Sterman, 2000, pp.411–412). Consequentially, identified delays have been drawn from the thematic analysis, and paired with variables A through I in Table 35 below. Sterman (2000, p.412) states that to characterise the delay structure and behaviour, two principle questions must be answered; what is the average delay? What is the distribution of the average time delay?

	Characteristic	Delay Type	Related Variable	Weighted Mean Delay
Delay 1	Vulnerability Discoverer Skills and Discovery	Informational	Variable H	10.7 Days
Delay 2	Coordinated Disclosure Communications (Platform)	Material	Variable E (Section 1)	77.4 Days
Delay 3	Coordinated Disclosure Communications (Direct to Originator)	Material	Variable E (Section 2)	98.2Days
Delay 4	Vulnerability Fix Creation	Material	Variable E & H	Incorporate within Delay 2 & 3
Delay 5	Vulnerability Fix Dissemination and Communication	Informational	Variable E	Assumed to be 5 Days
Delay 6	Awareness of Vulnerability Rewards	Informational	Variables F & G	10 Days (Estimated)
Delay 7	Software Quality	Informational	N/a	Unknown

Table 35 – Delays and Variable Mapping within the VDDS

Delays surrounding the initial discovery of a vulnerability is extremely variable and ranges from 1.5 days to 165 days and is highly subjective in nature (McQueen et al., 2006; Nom, 2015; Sommestad et al., 2012). Given the uncertainty and potential inaccuracies of the data we must therefore estimate the average time it takes a vulnerability discoverer to uncover a vulnerability using the empirical data collected and outlined in sections 5.4.5 – 5.4.5.1 above, specifically Tables 25 and 26. The calculated mean days to discover a vulnerability in a piece of software is 10.7 days per vulnerability per piece of software. The choice upon which discoverers make their decision to disclosure via the full disclosure, or the coordinated route is influenced by both incentives, and the sentiment toward the software originator. As such the time delay associated with this decision is hard to quantify. Therefore, an estimated value of 5 days has been provided, derived from qualitative observations of the VDDS. The more pernicious delay that exists within the VDDS is the time lag between the disclosure of the vulnerability to either the software originator

directly, or via a third-party vulnerability disclosure platform. Specifically, delays 2, 3, 4 and 5 are subject to prolonged interaction steps and delay, with the mean elapsed time of 77.4 days and 98.2 days respectively.

Alongside the delay associated with reporting and confirming the vulnerabilities existence, the production of a software fix that corrects the vulnerability is a further delay. This production is normally undertaken in parallel to the disclosure process, as the aim is to disclose the details of the vulnerability publicly, but cause no harm to end users. Subsequently, the production of a fix to the uncovered issue can be considered contributory to the delay which is experienced. Dissemination of the details as to how the vulnerability can be mitigated is largely an automated process with the downloading and patching of software undertaken without user intervention. However, Resorla (2003) suggests that after 15 days 60% of affected software is patched, however significant improvements have occurred since the publication of this paper (Rescorla, 2003).

Therefore, an estimated value of 5 days is provided. The receipt of rewards from third party disclosure platforms is imputed from available data as a single day as this is the final step in the interaction between software originator and vulnerability discover. The final set of delays 6 & 7 are ‘meta delays’, and influence all aspects of the VDDS insofar as the awareness of an alternative disclosure stance and the sentiment change over time. Both of these values are estimated due to the availability of data to base an assumption on and the probable errors that are latent within the collection of the data. As such the sentiment change that is stated is chunked into annual changes, therefore the most appropriate value to ascribe to the change in sentiment is 365 days. Alongside this the rate that a vulnerability discoverer may become aware of a new market of paid coordinated disclosure path is given as 10 days.

5.5.1 Delay Time Distributions

Having addressed the first of Sterman’s questions, we now turn to the aspect of the distribution of values around the mean delay time (Sterman, 2000, p.411). Delays measure the time that is taken for information or material to be

processed and transition through the system processes. This is measured by the input time (I), transit (T) and output time (O). Each delay within the VDDS operates differently, with varying impacts upon the output of the delay. Simple non-constant delays exist when material or information is processed in a serial manner. This is important as the order upon then enter into the delay, known as first in, first out (FIFO) dictates the processing speed, and hence delay time (Sterman, 2000, p.416). However, delays within the VDDS are not processes in serial, and exhibit a range of values which can be encountered by entities within the system. Therefore, delays within the VDDS are subject to 1st, 2nd and order delays. These delays manifest as cascading, or mixed delays whereby the transit of information or material is dependent upon the proceeding delay and so forth. It is worth noting that delays 1 through 7 will undoubtedly have a probability distribution that surrounds the mean delay time, however evidence to support an accurate construction of a probability distribution. Where accurate measurement is not possible, Gaussian distributions have been used, and values estimated.

5.5.1.1 Delay 1 - Discovery of Vulnerability

The time that is required to discover vulnerability varies widely (See Tables 25 and 26 above). Consequentially, a more precise measurement indicating the potential elapsed time the discovery process takes is needed. To gain an accurate set of values, the ExploitDB.com dataset was used, more specifically the time intervals between discovery events for the top 5 individuals throughout their active lifespan. Data was processed showing the elapsed time between the 1st, 2nd and nth discovered vulnerability and a weighted mean calculated. Censoring was applied to remove zero intervals from all data to show meaningful comparison between discoverers.

$$f_t(t) = \begin{cases} \mu \exp(-\mu t) & for\ t \geq 0 \\ 0 \end{cases}$$

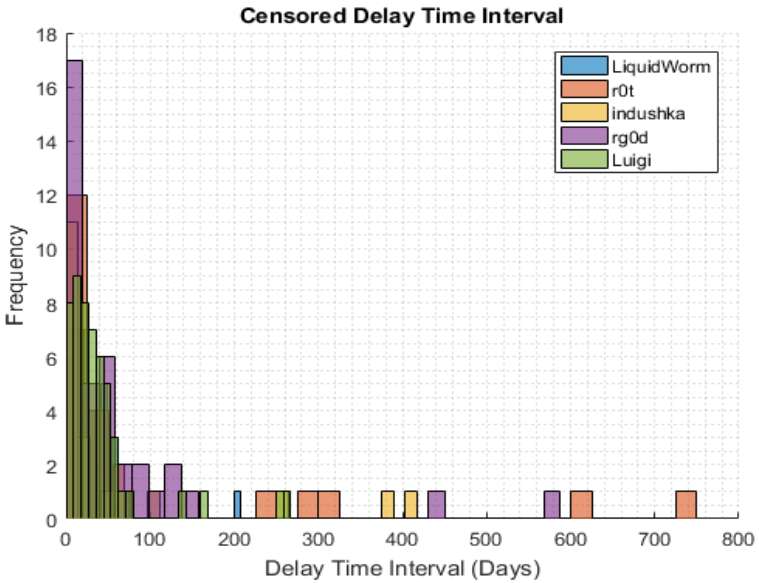
Equation 7 – Exponential Equation

Vulnerability Discoverer Name[2]	Frequency	Mean Censored Delay Interval (days)	Mean Uncensored Delay Interval (days)
Luigi_Auriemma	416	41.4	8.3
LiquidWorm	353	29.3	8.7
rgod	333	64.6	9.4
indoushka	294	69.3	5.4
r0t	257	113.2	9.9
Weighted Mean		63.5	8.3

Table 36 – Top Five Vulnerability Discoverers Delay Interval

Both Censored and uncensored data was used, to calculate time intervals for the top 5 discoverers resulting in 8.3 and 63.5 days respectively. The distribution for both uncensored and censored data follows the exponential distribution for all vulnerability discoverers. The exponential distribution is a continuous probability distribution with the following density function (Ayyub and McCuen, 1997, p.107) shown in Table 36 above.

[2] Groups of researchers exist within the dataset, for example Metasploit, High_Tech_Bridge_SA and Google_Security_Research. These groups are highlighted



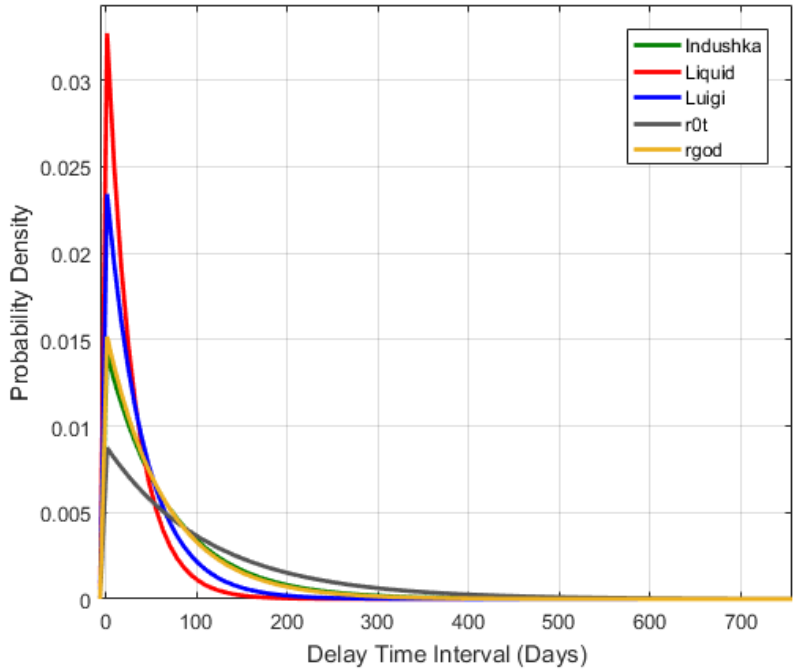


Figure 36 – Delay time intervals for top five vulnerability discoverers (Left)

Figure 37 - Fitted Exponential Probability Distributions for top 5 vulnerability Discoverer (Right)

	Vulnerability Discoverer Name[3]	First submission	Last Submission	Days active	Frequency	Current Probable State
1	**Metasploit**	16-06-08	21-11-16	3080	1450	**Active**
2	**Google_Security_Research**	04-04-13	20-12-16	1356	416	**Active**
3	Luigi_Auriemma	17-12-02	29-06-12	3482	416	Inactive
4	**High_Tech_Bridge_SA**	13-04-10	29-04-16	2208	409	**Active**
5	**anonymous**	01-08-88	13-04-15	9751	359	**Active**
6	LiquidWorm	22-07-08	16-12-16	3069	353	**Active**
7	rgod	21-05-05	11-12-13	3126	333	Inactive
8	indoushka	23-12-09	08-05-14	1597	294	Inactive
9	r0t	23-03-03	14-03-10	2548	257	Inactive
10	ZoRLu	17-02-08	24-09-14	2411	221	Inactive
11	ajann	26-05-06	15-01-09	965	204	Inactive
12	Lostmon	20-05-03	28-03-12	3235	188	Inactive
13	shinnai	11-12-06	23-11-16	3635	176	**Active**
14	Moudi	07-01-09	10-09-10	611	170	Inactive
15	laurent_gaffie	15-09-06	09-11-16	3708	161	Inactive
16	GoLd_M	07-01-07	10-08-12	2042	152	Inactive
17	Kacper	12-05-06	15-06-09	1130	149	Inactive
18	Stack	17-01-08	22-01-10	736	147	Inactive
19	S@BUN	22-01-08	25-11-09	2411	143	Inactive
20	cr4wl3r	03-08-09	24-12-13	1604	130	Inactive

Table 37 - Top Twenty Active and Inactive Vulnerability Discoverers compiled from ExploitDB

5.5.1.2 Delays 3, 4, 5, 6 & 7

Delays that include the interaction between the software originator and the vulnerability discoverer are outlined in section 6.3.2.1 within chapter 6. Nevertheless, it is worth noting that the probability distribution that fits the data is again exponential. The exponential probability distributed was fitted to both the HackerOne and Full Disclosure email archive.

[3] Groups of researchers exist within the dataset, for example Metasploit, High_Tech_Bridge_SA and Google_Security_Research. These groups are highlighted

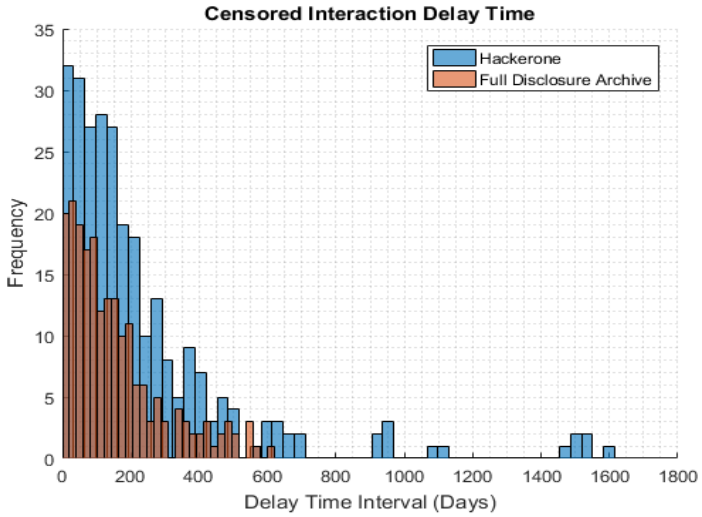


Figure 38 - Interaction Delay Time Plot (Left)

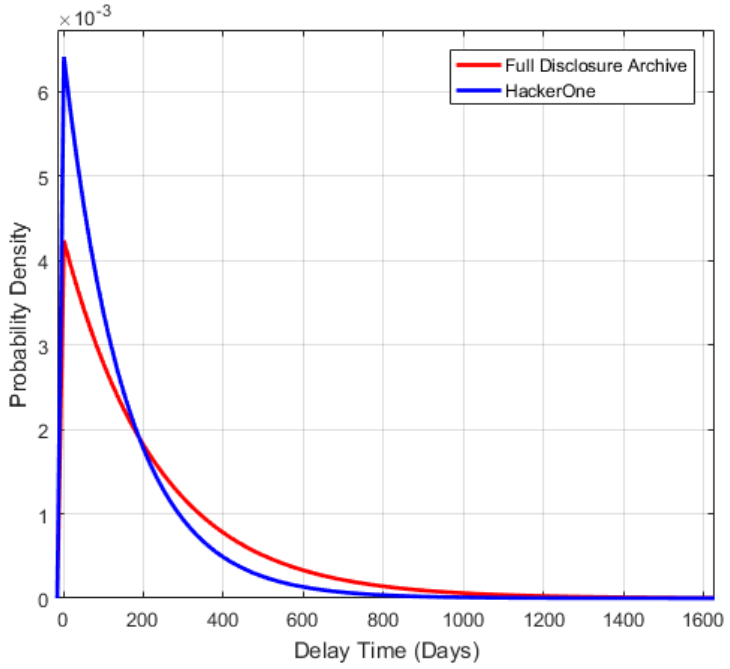


Figure 39 - Fitted Exponential Probability Plot (Right)

Figures 38 and 39 above show the exponential probability distribution that is exhibited with the weighted average delay being 154.4 days for HackerOne and 234.6 days for the full disclosure email archive, when used in a direct way.

5.6 Chapter Summary

To conclude this chapter has described the methods used in the investigation of several key variables that have been derived from qualitative thematic analysis. The identified key variables where characterised numerically, along with descriptive statistics, outlining the behaviour of the variable over time.

Several key findings have arisen from the analysis performed in this chapter. Specifically centred around time delays, disclosure steps and discovery activities. It has been shown that the duration it takes the vulnerability disclosure process to complete (the public release of vulnerability details) ranges from 0 day to 619 days. The mean for a vulnerability to transit the disclosure process is 77.4 days, with 0 days being the most frequent occurrence with the over half of vulnerability disclosures that are disclosed via the disclosure platform HackerOne are dealt with quickly, within 30 days. The average number of disclosure steps that discoverers have to negotiate is four, yet this can increase to 14 if over a long-time period.

The next chapter describes the construction of the VDDS model, drawing on the findings from both the thematic analysis and exploratory data analysis presented in chapters four and five. The exploration of the themes, state variable and delays within the VDDS has provided a solid foundation for the construction of a System Dynamics based model. This model will allow the examination of potential strategies for vulnerability based risk reduction policies.

6 Modelling the Vulnerability Discovery and Disclosure System

Thus far this thesis has outlined the building blocks upon which the VDDS is constructed. This chapter outlines the synthesis of all these aspects to construct a model of the VDDS. Having discussed the nature of the VDDS, this chapter builds upon the qualitative Thematic Analysis and numerical exploration of the historical behaviour of the system. The analysis that has been undertaken has established the relationships within the VDDS, representative state variables, historical behaviours and initial conditions.

The VDDS is made up of entities, interactions and delays, each characterised by a key state variable which has been characterised and quantified. Identified datasets were used to construct historical reference modes to provide insight of how these variables behave over time. However, to move further and begin to understand the systemic behaviours of how the VDDS operates, and crucially how state variables influence each other, we must construct a model of the VDDS. Systems Dynamics has been chosen to build and express both the variables and relationships between those variables in a way that allows for exploration of the interactions between them.

To model the VDDS accurately we must first adopt a specific nomenclature to designate the aspects of the VDDS correctly. The terminology used by Voinov (2008) has been adopted so that entities, relationships and interactions are defined. Initially we start with defining an *element* which is considered to be a building block of a system, and is considered to have both *properties* and *features* (Voinov, 2008, p.7). Features are a distinctive property of system (Peter interacts with Paul), whereas a property is an attribute (e.g. colour) of a feature or Interaction is defined again by Voinov by describing the type of relationship that may exist between elements; specifically flows of *material* and *information* (Voinov, 2008, p.9).

6.1 VDDS Problem Statements

To model any system or process one must adopt a process to investigate the phenomena (Sterman, 2000, p.85). Consequently, several process steps have been propsed by authors outlining the steps to be followed in constructing a System Dynamics model. The most comprehensive is outlined by Sterman (2000) which brings together process steps from both Coyle (1996, p. 11) and Wolstenholme (1996, p. 26-19):

- **Step 1: Problem Articulation** (Theme Selection; Key Variables; Time Horizon; Problem Definition)
- **Step 2: Formulation of Dynamic Hypothesis** (Initial Hypothesis Generation; Endogenous Focus; Mapping)
- **Step 3: Formulation of a Simulation Model** (Specification; Estimation; Tests)
- **Step 4: Testing** (Comparison to Reference Modes; Robustness Under Extreme Conditions; Sensitivity)
- **Step 5: Policy Design and Evaluation** (Scenario Specification; Policy Design; What if...; interaction of Policies)

Adapted from (Sterman, 2000, p.86)

Following the process steps outlined by Sterman (2000), we can see that chapters 4 and chapter 5 are dedicated to the initial step, problem articulation. Together, both chapters provide a foundation for articulating and defining the problem space, identifying themes and structures within the system and classifying key variables. To assist in the initial phase of constructing the dynamic hypothesis the formation of a high level problem statement is suggested by Coyle (1996) as a way to critically evaluate the system and problematic behaviour (Coyle, 1996, p.27). Therefore, a parsed narrative based on the thematic analysis outlining the VDDS is given below within Table 38:

The Vulnerability Discovery Problem Statement.
A software originator ***produces software***. The software which is produced is released after a period of time and sold (or in the case of Linux, supplied for free) to both individuals and large organisations. The time to produce software is based upon the requirements and features that are set by the customers of the software, and is considered a complex undertaking involving many thousands of man hours. The software originator has ***never been able to produce vulnerability free software***, however the ***desired state of any software system is to be vulnerability free.*** As such, quality assurance of the software takes place both prior to release of the software and afterwards. The pre-release activities undertaken consisting of the use of a 'safe' programming language, utilising a development methodology and black/white box testing. Alongside these quality assurance activities, several secondary undertakings are also started namely marketing and sales, previous software release retirement activities and support/training. ***Post release the majority of software testing moves outside of the software originators control, and is where vulnerability discoverers begin to test for issues.***
Post release a interrelated set of factors influence the discovery of vulnerabilities with the primary entity within the post release discovery process being the vulnerability discoverer. The rate which the ***vulnerability discoverer identifies vulnerabilities within software is dependent upon the skills, software quality, previous experiences and tools at their disposal***. There is no way to predict with any certainty when the next vulnerability will occur.
Once identified the vulnerability discoverer may ***choose to disclose the vulnerability to the software originator for altruistic reasons (known as coordinated disclosure)***. Other courses of action may result in selling the vulnerability on the open or underground vulnerability markets, (Paid for Coordinated Disclosure). ***Paid for disclosures attract monetary recompense in the form of currency.*** There are potentially significant delays that occur when disclosure happens. Dependent upon which approach is taken will influence potential future actions. The vulnerability discoverer can choose to be ethical or unethical, which is considered to be a rational choice. The rational choice is based upbringing, altruistic leanings and so on. The choice is also ***influenced by the narrative and discourse that surrounds the VDDS*****, which may contain both positive and negative influences.**
The number of vulnerabilities that continue to be discovered and disclosed continues to rise and be exploited by threat actors.

Table 38- VDDS Problem Statement

The problem statement shows a structure that is dynamic in nature, and has multiple interactions and dependencies. Furthermore, there are a number of identified processes and time delays that interact both from a resource

transformation and information transmission perspective. For example, if we focus upon one aspect of the problem statement, the relationship between the vulnerability discoverer and software originator we can see that this relationship has been, and continues to be problematic. Interactions between these two elements of the system are inconsistent, time consuming and at a high level an aspect of mistrust exists between them (I_didnt_do_that, 2011). Furthermore, the discourse that exists around this interaction has been consistently negative, influencing discoverers decision making as to how vulnerabilities are treated.

Selecting the most appropriate time horizon within the modelling process requires careful consideration of both how long the problem has been manifest, and meaningful enough to describe the problem itself (Sterman, 2000, p.90). Furthermore, the availability of data that allows exploration of the VDDS and the variables that make up the VDDS are of equal importance. Hence the time horizon that has been selected to model and simulate the VDDS is the time period between 1st Jan 1990 and present day (in the case of this research data collection stopped on 16th Dec 2016). The basis for the selection is threefold. Firstly, to gain the fullest understanding of the VDDS all possible and appropriate data must be included within the simulation and model. Secondly, a system as complex as the VDDS may exhibit behaviours that originate from simple interactions or delays from timeframes outside of the simulation windows in data is truncated. Thirdly, all features, such as the introduction of new markets for vulnerability disclosure, ethical considerations and the emergence of the internet occurred in this time. Where data is not available to quantify aspects of the VDDS values have been imputed or, in the case of delays qualitative observations have been deconstructed to provide a reasonable approximation.

6.2 Dynamic Hypothesis Formulation

To begin with a dynamic hypothesis was constructed from the thematic and exploratory analysis to form a contextual and environmental evidence base to work from (Sterman, 2000, p.95). A dynamic hypothesis is considered a ‘working’ hypothesis that initially guides and reduces a complex problem to

allow the formation of a theory of how the system operates (Morecroft, 2015, p.111). It is a widely held view that a number of fundamental modes of structure account for a large proportion of observed behaviours, and therefore can be explained using System Dynamic techniques (Forrester, 1958; Morecroft, 2015, p.111; Sterman, 2000, pp.118–132). Previous studies have indicated complex behaviour containing differing interactions between system elements characterised by feedback loops, delays and transformation rates may cause unexpected behaviours. These interactions are characterised by defined structures, which create patterns of behaviour and recur throughout nature and complex systems – commonly known as system archetypes (Senge, 1990, p.94).

To construct our dynamic hypothesis, we must revisit the earlier chapter – thematic analysis – and the identified themes. To recap the identified themes were Perception of Punishment. Vendor Interactions, Disclosure Stance and Ethical Discourse; Motivation for Discovery and Disclosure Emergence of New Vulnerability Markets. These themes represent the ground truths that have emerged from empirical observation which indicate the structure of the VDDS. Coupling the phenomena under investigation (vulnerability growth) with identified variables and behaviours provides the necessary components to assemble an initial model.

Considering the problem statement and thematic analysis together a number of key entities have been identified within the VDDS:

- Software Discoverer;
- Software Originator (Vendor);
- Software Vulnerability Brokers (Paid);
- Software Vulnerability Disclosure Platform;
- Cyber Criminals;
- End User;
- National Governments;
- Standards Bodies;
- Law Enforcement;

6.2.1 Description of Entities of the VDDS

Primary elements are those which are directly involved in the production, or discovery of software vulnerabilities and directly affect the behaviour of the VDDS. The two primary elements within the VDDS are Vulnerability Discoverer and Software Originator. Both elements are fundamental to the VDDS as without them the system would simply not exist. As such these elements are characterised as central, and interactions with these elements produces the most behavioural change within the VDDS.

Vulnerability Discoverer (VD) searches for vulnerabilities within software. It is an active entity which ascertains the existence of the vulnerability by drawing upon skills, resources and tools at its disposal. The vulnerability discoverer can be ethically characterised as 'black', where the intention is to provide the discovery of a new vulnerability to a cyber-criminal (CC) on the black market. The other is 'white' where the intention is to provide the discovery to the software vendor (SV) for fixing and dissemination. A third course of action also exists where the discovery event is reported to a vulnerability broker (VB) or vulnerability platform (VP) where both the CC and SV actors may bid or pay for the vulnerability.
Entity Interactions: Cyber Criminal (CC), Software Vendor (SV), Vulnerability Broker (VB), vulnerability Disclosure Platform.(VDP)

Software Originator (SO) creates, markets and supports the software which has the discovered vulnerability within it. It is an active entity which creates software for end users, and relies upon the skill and resources of its employees to produce vulnerability free software. The software originator may interact with Vulnerability Discoverers (VD), National Governments (NG), Standards Bodies (SB), Vulnerability Platforms (VP) and Vulnerability Brokers (VB) as each may influence the production of good quality code. The software originator will also provide patches and updates to the software once a vulnerability has been discovered. Software Originators are considered to be the externally facing aspect of the software originator. The software vendor is responsible for functions that are not undertaken by the software originator, for example sales and marketing or liaison with vulnerability brokers.
Entity Interactions: Vulnerability Discoverers (VD), National Governments (NG), Standards Bodies (SB), Vulnerability Platforms and Vulnerability Brokers (VB)

Secondary entities are defined as those who have direct contact with software systems, but do not have a direct impact or influence on the production of vulnerabilities within the system. Secondary actors may influence the rate of discovery via policy or actions upon the primary actors.

Cyber Criminal (CC), creates and deploys the software vulnerability within an attack vector (e.g malware). The attack vector is normally targeted at end users of the software, providing a level of risk. The cyber-criminal may interact with both the Vulnerability Discoverer and Vulnerability Broker to obtain vulnerabilities for their use. Both National Government and Law Enforcement will interact with the Cyber-Criminal primarily via intelligence gathering or enforcement activities.
Entity Interactions: Vulnerability Discoverer (VD), Vulnerability Broker (VB), National Government (NG), Law enforcement (LE)

Vulnerability Disclosure Platform (VDP); interact with the Software Originator (SO) and vulnerability discoverer (VD) to provide a service that provides post release security assurance, and potential rewards to discoverers. The service that is provided is a typical broker service, however more transparent, and provide a level of control to the vulnerability discoverers as they adhere to full disclosure principles.
Entity Interactions Software Originators (SO), Vulnerability Discoverers (VD) Vulnerability Brokers (VB)

Vulnerability Brokers (VB), is an intermediary between the Vulnerability Discoverer and the software vendor or Cyber-Criminal. The broker purchases vulnerabilities and provides a conduit for the marketplace that exists for the buyers of vulnerabilities.
Entity Interactions: Software Vendor (VB), Cyber Criminal (CC), Vulnerability Discoverer (VD), Software Originator (SO)

Tertiary actors can be described as the actors who take part in the system but their effects are negligible, or there is significant delay in their policy decisions being felt upon primary or secondary actors. Additionally, the boundary between secondary and tertiary actors can be porous and may change rapidly.

VDDS Participants
Entity Interactions VDDS participants are classed as entities that add to the discourse that surrounds the VDDS, yet do not discover or disclose a vulnerability, yet impact upon the sentiment that exists within the VDDS. VDDS participants influence the software discoverer as they offer opinion or experience in the discoverer and disclosure processes.

End User (EU), receives and operates the software that has been produced. The end user is directly affected by the vulnerabilities that are present within the software as they may be used to subvert security controls. Each end user will have a relationship with the software originator, either directly or via a patching dissemination mechanism. The burden of risk is present in this entity as they are utilising potentially vulnerable software. End users may also be involved with standards bodies if they are sufficiently concerned around the production and maintenance of secure code. The end user also controls the market share of the software originators product.
Entity Interactions: Software Vendor, Standards Bodies.

National Governments (NG), provide governance, policy and regulation of the vulnerability environment. NG also provide resources that may influence all actors involved in the vulnerability discovery process. NGs may interact with standards bodies, law enforcement, software originators, end users via governance and policy transactions Security Information Provider (SIP), provides information on their current state of 'active' vulnerabilities that are being reported in the wild. The information provider may also interact with Vulnerability brokers to allow time for security vendors to provide code fixes to vulnerable software.
Entity Interactions: Cyber Criminal (CC), Software Originators (SO), Standards Bodies (SB)

Standards Bodies (SB), are responsible for setting acceptable levels of software code, and producing guideline and methodologies for achieving the standard. Normally at a national level, standard bodies can directly influence the quality of code that is produced, and is normally made up of many actors, primarily Software Originators, National Governments and End Users.
Entity Interactions: Software Originator (SO), National Government)NG), End Users (EU)

Law Enforcement (LE), are responsible for enforcing the law, tracking and disrupting criminal activities. Each national government has a law enforcement capability with varying efficiencies. Cyber criminals are transnational and as may cross multiple sovereign boundaries very quickly. Therefore the disruption activities of law enforcement may not have the desired effects or impacts.
Entity Interactions: Cyber Criminal (CC), National Government (NG)

Each entity typically has a relationship with one or more of the other elements of the VDDS. For example, the software vulnerability discoverer interacts and depends upon the software originator to produce a piece of software, and for that software to have vulnerabilities within it (Eduard Kovacs, 2011). Likewise, vulnerability brokers and vulnerability disclosure platforms depend upon vulnerability discoverers to find vulnerabilities within software so that they can sell to the highest bidder. Within the VDDS, as with most systems, a hierarchy of sub-system, system and super system exists, with differing elements operating within each, or across system boundaries. Again, adopting accepted nomenclature, we assign levels of primacy to the elements that operate within the system, specifically primary, secondary and tertiary. The levels that are ascribed to the elements are evident as they are central to the behaviour of the VDDS, see Table 40.

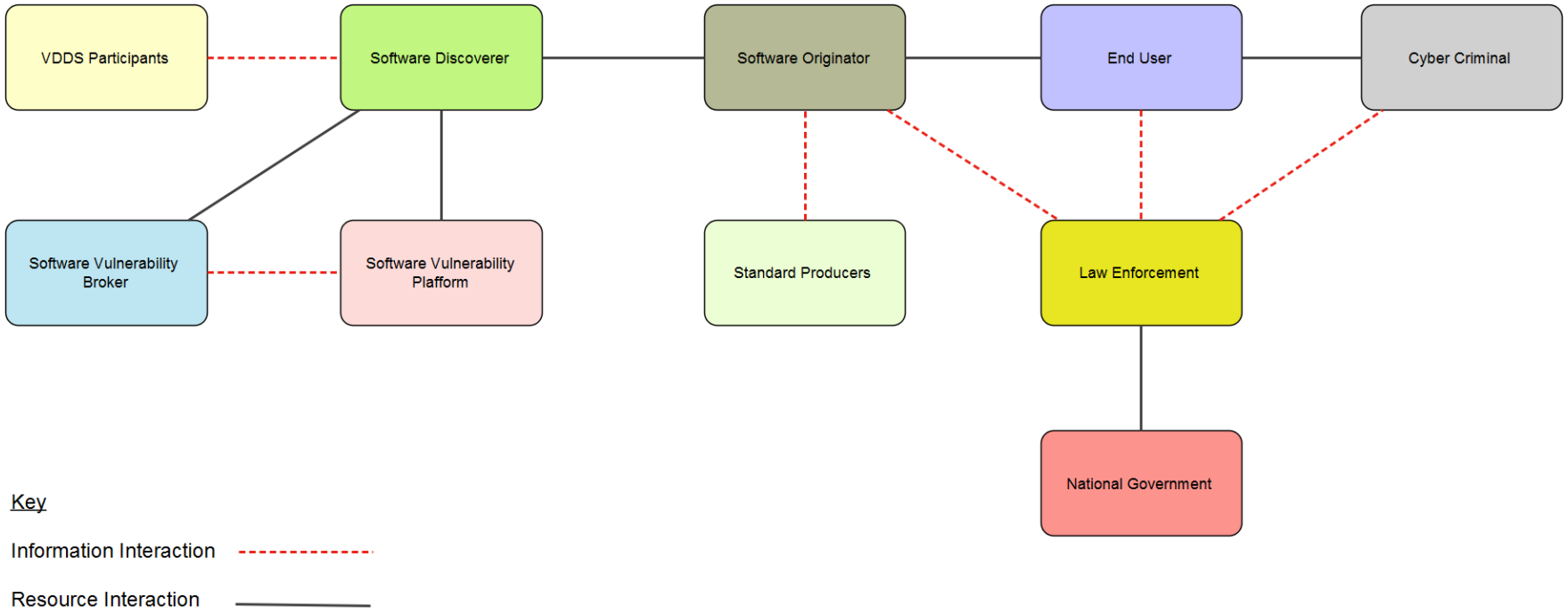


Figure 40– Entity Interactions Showing Resource and Information Transfers

6.2.2 Linking Raw Data to Model Variables and Structure

As with any research that constructs a model from empirical observation, the ability to understand the lineage between raw data and the variables representing the data is critical. Establishing a link to how the variable is rooted within the data also provides confidence that the model represents the phenomena under investigation.

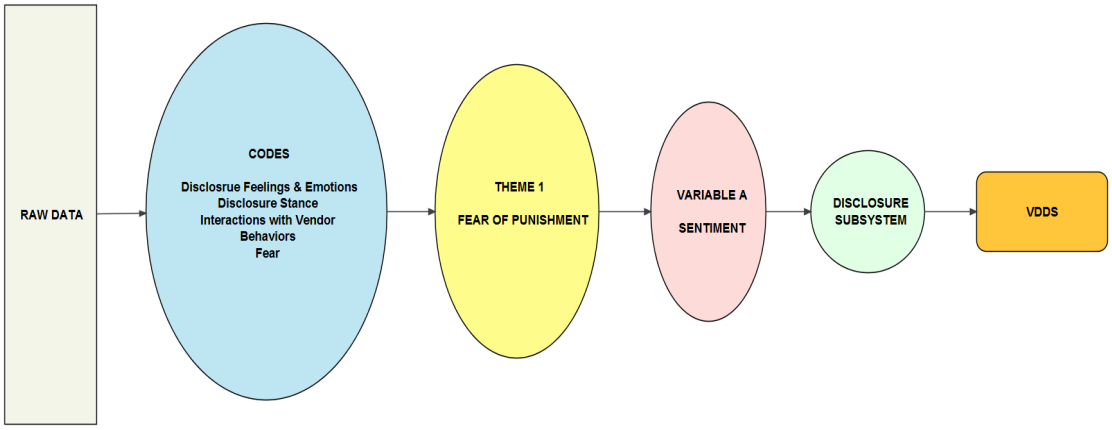


Figure 41 – Relationships Linking Raw data to the VDDS Model

For example, the set of codes that represent interaction with the vendor were derived from the raw data, and clustered around feelings, emotions, interactions with software vendors and punishment. Once synthesised Theme one – Fear of Punishment was created representing all the coded items from within the data corpus. Once all themes were generated, variables to represent the theme were codified and explored in the case of this example Variable A – Sentiment. Finally, the structure and interactions of entities within the theme and variable was constructed using causal and stock and flow diagramming techniques to form the disclosure subsystem within the main VDDS model. Table 39 below shows the linkage between the raw data and the VDDS in addition to identified structural delays.

Linkage →					
Related Themes	**Variable Name**	**Delay Reference**	**Structure Loop IDs**	**Primary Subsystem**	
Theme 1: Perception of Punishment	Variable A: Researcher sentiment	Delay 2, Communications (Coord) Delay 4, Fix Creation Delay 5, Fix Dissemination	Loop A Disclosure (Coord) Loop H – Full Disclosure	Disclosure Subsystem	
Theme 5 Emergence of new vulnerability Markets	Variable B: Number of Vulnerability Discoverers within System	Delay 6, Awareness of Rewards	Loop G Discovery Loop A Disclosure (Coord) Loop D – Rewards Loop C - Discoverers	Active Discoverers Subsystem	**V** **D** **D**
Theme 2: Vendor Interactions	Variable C: Vulnerability Removal Rate	Delay 4, Fix Creation	Loop G Discovery	Discovery Subsystem	**S**
Theme 1: Perception of Punishment	Variable D: Full disclosure and Coordinated Disclosure Ratio	Delay 2, Communications (Coord) Delay 3, Communications (Direct) Delay 4, Fix Creation Delay 5, Fix Dissemination	Loop A Disclosure (Coord) Loop B Disclosure (Direct)	Disclosure Subsystem	
Theme 2: Vendor Interactions	Variable E: Time to fix vulnerability from	Delay 2, Communications (Coord) Delay 3, Communications (Direct)	Loop A Disclosure (Coord) Loop B Disclosure (Direct)	Disclosure Subsystem	

	coordinated disclosure	Delay 4, Fix Creation Delay 5, Fix Dissemination	Loop H Full Disclosure		
Theme 4: Motivation for Discovery	Variable F: Monitory reward	Delay 6, Awareness of Rewards	Loop D – Rewards Loop F Participants	Active Discoverers Subsystem	V D D S
Theme 5: Emergence of new vulnerability Markets	Variable G: Number of Bug Bounty Schemes	Delay 6, Awareness of Rewards	Loop D – Rewards	Active Discoverers Subsystem	
Theme 3: Ethics and Disclosure Stance	Variable H: Participant Activity	Delay 7, Software Quality Delay 1, Vulnerability Discovery Skills	Loop G Discovery Loop A Disclosure (Coord) Loop B Disclosure (Direct)	Disclosure Subsystem	
Theme 5: Emergence of new vulnerability Markets	Variable I: Market Share of Software	Delay 7, Software Quality	Loop E – Desire to Fix	Active Discoverers Subsystem	

Table 39 - Linkage between Codes and VDDS Subsystems

6.2.3 Model Boundary & VDDS Structure

To model any system one must choose the appropriate scope of the system, (I.e. what you wish to model) and bound the system accordingly. As the VDDS has been characterised as a wicked problem on a global scale, choosing a meaningful scope is challenging. Therefore, the scope of the VDDS will not be bounded by geography (as the VDDS is global in nature) or time (the VDDS time horizon encompasses the entire time of the existence of the VDDS), but will be bounded by primacy of elements and fidelity. In the context of this research we take primacy as to mean the importance and impact of the element on the system, and fidelity as the amount of data and insight we have on selected elements. These two aspects of the VDDS have been chosen due to the concept known as *bounded rationality*; we have limited information about the system in totality, and cannot process the whole of the data generated by it. Sterman (2000) suggests that to summarise the scope of a system, a model boundary chart is constructed to limit the scope of the model, and clarify which variables are endogenous, exogenous and excluded from the model (Sterman, 2000, p.97).

Endogenous	Exogenous	Excluded
Variable A: Researcher sentiment	Technology Trends	Technology Trends
Variable B: Number of Vulnerability Discoverers within System	Number of General Cyber Attacks	General IT literate Population
Variable C: Vulnerability Removal Rate	Government Policies	Software Testing Methodologies
Variable D: Full disclosure and Coordinated Disclosure Ratio	Law Enforcement	Gross Domestic Product
Variable E: Time to fix vulnerability from coordinated disclosure	New Business Opportunities	Software Vendor Competition
Variable F: Monitory reward	Cyber Criminal Activity	General IT literacy Population
Variable G: Number of Bug Bounty Schemes		Globalisation Retrenchment
Variable H: Participant Activity		
Variable I: Market Share		

Table 40 – Exogenous, Excluded and Endogenous Variables

Each category of variable that is assessed within the system plays a part in influencing the behaviour of it. The different aspects of both endogenous and exogenous is the ability to affect change upon them. Clearly government or law enforcement policy is important and can impact the VDDS significantly over time, though time to change policy is slow. Similarly, cybercriminals and vulnerability brokers also have impact, yet the ability to effect direct impact upon them is limited unless you are tasked remit to deal with them (i.e. criminal arrest). Therefore, the boundary of the system is sketched in Figure 42 (page 206) with endogenous variables highlighted in blue, within the model boundary and exogenous highlighted in yellow.

6.2.4 Sub Diagrams & Basic Mechanisms

Sub-diagrams in the context of System Dynamics provide an overall architecture of the model, showing flows, structure and coupling between each element or sub system (Sterman, 2000, p.99). Sub-diagrams have been used extensively to aid in the development of the dynamic hypothesis, providing a high level overview, and bounding the problem space (Sterman et al., 1997, p.38).

Using the sub-diagram and evidence from Chapters 4 and 5 one of the key interactions is between the software vendor, vulnerability platform and vulnerability discoverer. This tripartite interaction corresponds to the accepted ground truth within the literature about the software vulnerability system. At a high level the interactions consist of:

- creation or reintroduction of a software vulnerability within the software that has been created;
- discovery of vulnerability within the software under scrutiny;
- trade of the software vulnerability;
- disclosure of the vulnerability to the general public and patching;
- profiting from vulnerability knowledge.

The interaction between the software vendor, vulnerability platform and the vulnerability discoverer is crucial (detailed in Chapter 5). Once the vulnerability has remained dormant within the software for an undetermined amount of time, the software vulnerability is discovered by a vulnerability discoverer. The vulnerability discoverer is now required to make a choice, a) report directly to the software vendor or, b) sell the software vulnerability to a vulnerability platform. The notable difference between these two choices is that selling to a vulnerability platform introduces the notion of monetary gain.

The interaction between the software vendor and the vulnerability discoverer is generally direct, and without the use of intermediaries. For a number of years this was the de facto way that vulnerabilities were disclosed to, and addressed by, the vendor and to the community. Historically the relationship was based

upon mutual trust, and considered to be the proper approach, so that if a software vulnerability was discovered it was reported altruistically and fixed. However, the relationship between these two elements of the system has deteriorated over time and a shift from responsible disclosure to full, or brokered disclosure occurred. Vulnerability platforms are commercial enterprises which trade vulnerabilities within the VDDS extracting value from the vulnerability by selling detection characteristics to their customers thus providing a degree of protection from publicly unknown vulnerabilities. Alongside these services, platforms may also provide interfaces between the software vendor and discoverer. Two further endogenous variables exist that influence the VDDS, those of third party talkers and end users.

Third-party talkers are elements of the VDDS that influence the software vulnerability discoverer, software originator and the end user and impact upon the choices those elements make. Whilst not a distinct group of actors or processes such as a software originator or vulnerability platform, the discourse produced by the talkers is disseminated via social media or specific knowledge exchange platforms such as closed online forums. The discourse that surrounds the decision-making processes, specifically the disclosure stance with respect to vulnerability discoverers, is generated by actors discussing, quoting, amplifying and potentially distorting aspects of the VDDS. In almost all cases talkers who discuss the VDDS are not directly involved within the primary process of discovery or disclosure of vulnerabilities, but discuss the process as a third party. Alongside the third-party talkers are the final element of the endogenous VDDS, end users. End users are ultimately the consumers of the software created by software originators and the victims of potentially exploited vulnerabilities uncovered by discoverers and used by exogenous elements such as cyber criminals. Consequentially, the end user holds significant power in the ability to influence the VDDS, yet rarely does so.

The exogenous set of elements and interactions are between the software vulnerability broker, cyber-criminal, national government and law enforcement. These sets of interactions are characterised by the illegal accumulation of

wealth or assets in the case of Cybercriminals. Cyber criminals may task vulnerability discoverers to ‘look’ for vulnerabilities within well used software, which in turn maybe used within malware. Conversely the cybercriminal may approach a black or illegal software vulnerability broker for a vulnerability to use. Each method of obtaining a vulnerability involves payment. The interactions between these entities is both transient and sporadic. Black market brokers act as an interface to the criminal underworld where purchased vulnerabilities maybe used within malicious software or used to gain unauthorised access to information systems. The vulnerability discoverer is the key entity within the triad as without it vulnerabilities within software would not be discovered, other than unintentionally or by mistake. Vulnerability discoverers can be either individuals or groups. Each of these will share information and tools to enhance their capability to find software vulnerabilities. The final tertiary set of interactions, which are excluded from the VDDS are general societal or economic factors and impact. Each of these interactions with the VDDS are subtle and wide ranging. Specifically, general literacy and software vendor competition processes are hard to quantify and difficult to predict, for example the potential retrenchment of globalisation and nationalism in early 2017 (Green, 2016).

There is a third option that is worth noting, yet out of scope of this research, vulnerabilities that are discovered and used by nation states. The impact of the actions and commissioning of contractors to discover issues within software is unknown, yet becoming a significant issue (Ablon et al., 2017; Goodin, 2017a). This is particularly acute when we consider the events that led up to the WannaCry malware outbreak of April 2017, which was allegedly based upon a US government developed vulnerability (Goodin, 2017b).

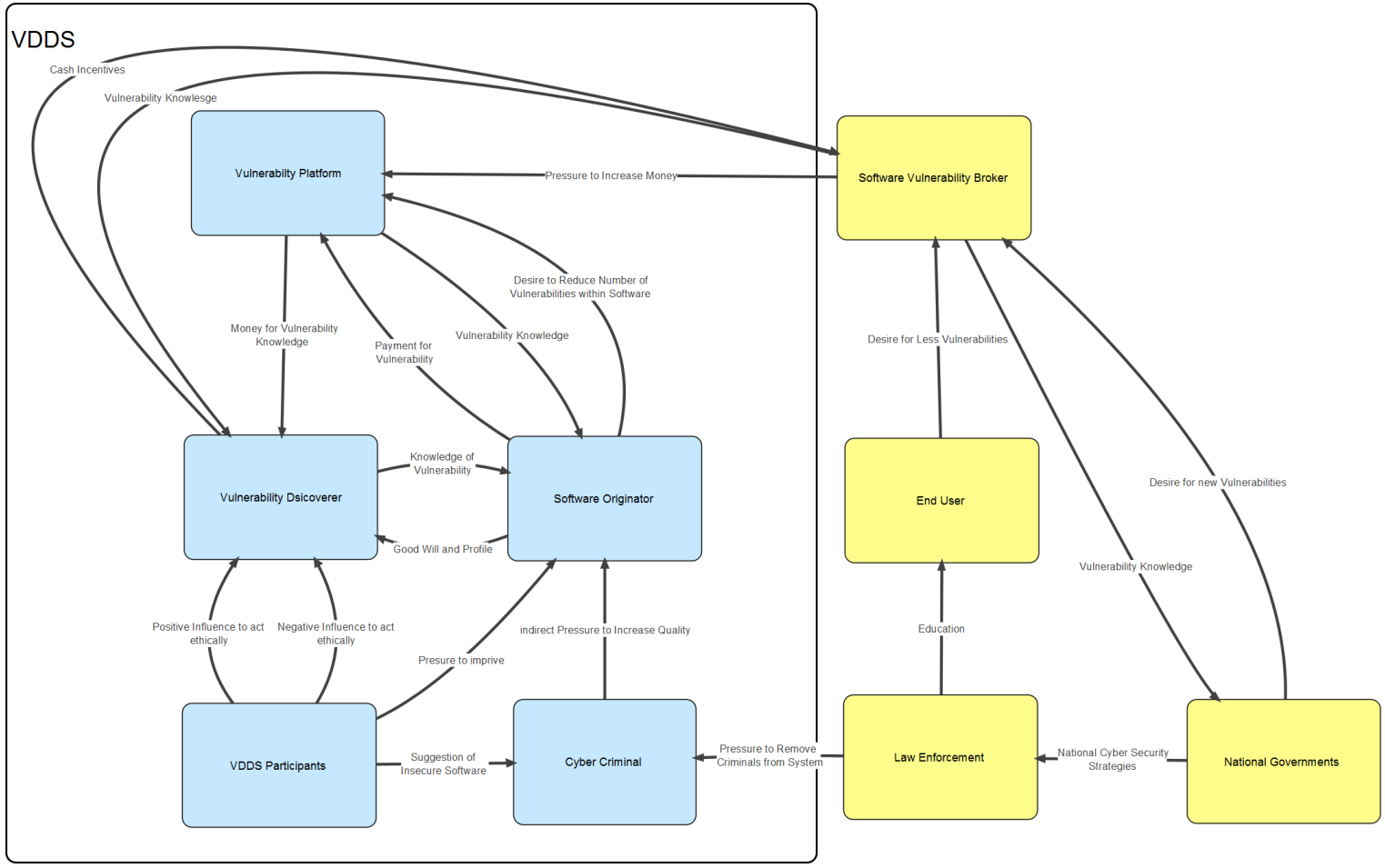


Figure 42 - VDDS Model Boundary and Exogenous Influencing Factors, with model scope within black rectangle

6.3 Structural Mapping of the VDDS

Model boundary charts, subsystem diagrams and identification of historical variable and reference mode behaviours provide an insight as to how the VDDS is bounded, however they do not express in detail the interactions that exist (Sterman, 2000, p.102; Wolstenholme, 1996, p.17). To adequately formulate a dynamic hypothesis, we must utilise both influence diagramming (or sometimes known causal loop diagramming) and stock and flow maps. An influence diagram, according Coyle show the:

"…influences at work in the system, the interplay of which is the cause of the dynamic behaviour". (Coyle, 1996, p.18)

Influence diagrams allow the structure of the VDDS to be constructed and map the interactions of all elements within the system boundary. Influence diagrams follow a specific notation, and this research will follow the conventions outlined by Sterman, Coyle and Wolstenhome (Coyle, 1996, p.22; Sterman, 2000, p.139; Wolstenholme, 1996, pp.17–20). Influence diagrams are used to capture the hypotheses about dynamic behaviour and feedback loops that maybe apparent, and consist of variables connected via arrows indicating the direction of causal influence. Additionally, each causal link is accompanied by a polarity showing how the dependant variable changes when the previous variable changes. See Sterman (2000, p. 138-139) for an expansive overview of the use of polarity and influence diagramming. The adopted notation for constructing the influence diagrams for this research is outlined in Table 41.

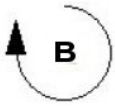
Negative Causal Correlation *If the cause increases, then the effect decreases*	
Positive Causal Correlation *If the cause increases, then the effect increases*	
Positive Informational Link *Where physical flows do not occur*	
Reinforcing Loop	
Balancing Loop	

Table 41 – Influence Diagramming Notation

6.3.1 Total VDDS Causal Loop Diagram

System Dynamics uses two distinct yet related techniques for modelling complex systems such as the VDDS; Causal Loop Diagramming (CLD) and Stock and Flow Diagrams (SFD). Each technique has a set of diagrams and nomenclature that is used to express flows and accumulations of quantities, for example the quantity of vulnerabilities that maybe present within a piece of software. By using CLD diagramming these variables are presented as words, and flows between them connected via directional arrows.

Drawing upon the overall system VDDS sub diagrams and detailed evidence collected in the thematic and EDA phases a series of CLD's to describe the system at a high level was constructed. The CLD's provide insight into the overall structures that require investigation, focusing upon the causal links or loops that are formed within the system. More specifically influence diagrams

look to establish the existence of information and resource loops that interact to create potentially dynamic behaviour within the system. The approach that has been selected to build an influence diagram is the list extension method outlined by Coyle (1999, p. 31-33), as the approach lends itself well to the nature of collected evidence. The list extension method provides a systematic approach to build logically consistent and robust influence diagrams, with the exploration of variables of interest at the heart of the method.

The initial variable, and arguably most important feature of the VDDS is the number of vulnerabilities that have been publicly disclosed, or more specifically the deficit between the desired state, i.e. the number of vulnerabilities (zero), and the recorded number. The variable `Vulnerability Discrepancy` can be considered a direct analogue of the number of vulnerabilities that are recorded within the national vulnerability database. The first list extension yields two variables `Quantity of Publicly disclosed variables` and `Desired Number of Disclosed Vulnerabilities.` These variables have an immediate and direct effect on `Vulnerability Discrepancy` and represent the discordance between the actual amount of vulnerabilities and desire to produce vulnerability free software. The causal link between `Desired Number of Disclosed Vulnerabilities` and `vulnerability Discrepancy` is positive as once a vulnerability is discovered urgency is needed to disclose the vulnerability due to malicious exploitation. This may seem counterintuitive as software originators may wish to keep vulnerabilities confidential however, once the vulnerability has been located a patch must be created to mitigate the issue. The second list extension shows `quantity of available undiscovered vulnerabilities within software` and `quantity of discovered vulnerabilities`. Both variables represent the total quantity of vulnerabilities within software both discovered and undiscovered. There is a time delay between discovered and undiscovered vulnerabilities which is the time taken to uncover vulnerabilities, or discovery time. The third and fourth list extensions include the variables `quantity of active discoverers, Monitory Rewards` and `Quality of software.` Both `quantity of active discoverers` and `monitory rewards` form the driving force of the system. The number of active discoverers directly

influence the number of vulnerabilities that are discovered as the more discoverers that exist, the more the possibility of discovery. The level of proportionality will be explored later within the subsequent simulation chapter. Quality of software also influences the number of vulnerabilities that are available for discovery. Finally, the rewards that are used to entice both vulnerability discovery, specifically money influences the discoverers desire to discover vulnerabilities for profit. Each vulnerability element is considered to be a product of deeper subsystems that operate at a more granular level. Therefore, each element of the high-level influence diagram will be decomposed into further influence diagrams.

6.3.2 VDDS System Archetypes, Delays and Loop Polarity

Within the VDDS there are several loops that are immediately obvious, and the corresponding flows around them. Loops within influence diagrams are characterised as either positive (reinforcing) or negative (balancing) and act upon the variables by decreasing or increasing the value (Sterman, 2000, p.142). The initial loop, labelled Loop A, is identified as a negative balancing loop between vulnerability discrepancy, quantity of publicly disclosed vulnerabilities and quantity of discovered vulnerabilities. The loop illustrates the interaction that has been characterised as the coordinated (unpaid) disclosure movement, where vulnerabilities are shared with software originators for the common good. The behaviour brought under control here is the growth of vulnerabilities within software. Loop B is a positive reinforcing loop between quantity of vulnerability discoverers and quantity of discovered vulnerabilities. This loop is the driving force of the VDDS when coupled with variable controlling the number of undiscovered vulnerabilities. The behaviours here correspond to disclosure of vulnerabilities, feeding into the discrepancy between desired, and actual recorded vulnerabilities. The third loop, (Loop C) is also a positive reinforcing loop increasing the number of vulnerability discoverers, thus increasing the number of discovered vulnerabilities. The fourth and final loop that is present (Loop D) is included within the high-level influence diagram as it plays a role in the number of discoverers that are within the system, and

potentially which disclosure route will be taken. The influence of this variable is represented as the aggregate sentiment of how the software originator is represented within the community.

Two distinct system archetypes exist within the VDDS, an exponential growth archetype and logistic growth (Senge, 1990, pp.379–384). Both archetypes operate different levels within the system and differ in primacy or dominance over time. The logistic growth archetype consists of a balancing causal loop, as the number of software vulnerabilities within the software is finite, and uncovering the vulnerabilities within software provide limits to the exponential growth of the discovered vulnerabilities.

The `quantity of undiscovered vulnerabilities within software` variable is the theoretical maximum amount of vulnerabilities that are within the software. The measure is difficult to calculate; however reasonable approximations can be made by estimating complexity of the software prior to the discovery of vulnerabilities and by calculating the number of vulnerabilities at the end of the software lifecycle. This measure provides an asymptotic maximum to the total amount of possible vulnerabilities that can be found within the software. The better the quality the software is, the lower the number of vulnerabilities that will be introduced, and with a lower level of quality the higher number of vulnerabilities. The pre-release software quality variable is a measurement of the practice and procedures that are used to create the software. The quality of the software is measured via the proxy variable of vulnerability density which allows us to measure the quality of the system, which is made up of a number of vulnerabilities and source lines of code. A further exponential archetype exists alongside and interacts directly with Loop B, which increases the number of vulnerability discoverers within the system, and is dependent upon the availability of monitory rewards for finding vulnerabilities. Within this variable is the concept of researcher interest governing non-monetary motivation. The interest is directly linked to the number of vulnerabilities discovered, as the researcher is the sole entity who causes a discovery event to occur. A final archetype is represented as a delayed and broad interaction that is reinforcing

and increases the number of vulnerability researchers that are present within the system. Finally loop E feedbacks the value of the potential rewards that exist by disclosing via a broker or vulnerability disclosure platform.

6.3.2.1 Casual Loop Delays

Within the VDDS several delays have been identified ranging from interaction delays between vulnerability discoverers and software originators through to production of software fixes. Delays within the VDDS are both resource transit, and informational in nature, each occurring within differing parts of the VDDS.

Delay 1 - Vulnerability Discoverer Skills

Vulnerability discovery is a labour intensive and highly technical activity, which in some cases may take several years to become proficient in uncovering issues. Consequentially, there is delay in gaining the requisite skills and experience required to enter the VDDS. As outlined in section 5.5 previously delays impacts upon the number of vulnerability discoverers available to find vulnerabilities within software.

Delay 2 – Coordinated Disclosure Communications (Via Disclosure Platform)

One of the major delays within the vulnerability disclosure process is the communication between software originator and vulnerability discoverer. The delay is a composite delay, cascaded with Delays 3,4 & 5. This delay cascade is the most significant within the VDDS and in some cases, may consist of delays of up to 3 years. The cascading nature of the delays is also a consequence of the risk upon which the software originator must mitigate. This differs from Delay 3 as a third party exists between the software originator and vulnerability discoverer. Details can be found in section 5.5.

Delay 3 – Coordinated Disclosure Communications (Direct to Originator)

Running potentially in parallel with Delay 2 is the Delay in coordinated disclosure delay direct to the software originator. This delay is comparable to Delay 2, however does not have a third-party mediating communication or

rewards. In addition, delays are normally encountered internally to the software originator. Details can be found in section 5.5.

Delay 4 - Vulnerability Fix Creation

The creation of a fix to the issue that has been uncovered is highly variable and there is no standard time delay. It is assumed that the processes that cause delay are due to the complexity of producing a fix to the vulnerability and the serial nature of the processes that take place within the software originator. Details can be found in section 5.5.

Delay 5 - Vulnerability Fix Dissemination and Communication

Once a fix to the disclosed vulnerability details are produced it then needs to be both communicated that a fix exists and disseminated to vulnerable software users. The communication of fix details is in most cases immediate, however the application of fixes to software is dependent upon local complexities and policy Details can be found in section 5.5.

Delay 6 - Awareness of Vulnerability Rewards

The usage of vulnerability disclosure platforms has increased over the past 5 years. As such the awareness of potential rewards via disclosing to the platform has also increased. The delay surrounding announcing new programmes or schemes is considered insignificant.

Delay 7 - Software Quality

Once created the quality of software is considered difficult to improve without significant resources and time. As such the usage of standards, improved development practices and skilled engineers to improve the quality of the software can in almost all cases take years rather than months or weeks. Details can be found in section 5.4.3. Using nomenclature outlined in Sterman (2000), delays are denoted by rectangles, influences with bold arrows and polarity of the influence at the end of the arrow.

The total VDDS is shown in Figure 43 below and outlines the major delays, feedback loops and most importantly the interactions between variables. (Sterman, 2000, p.442).

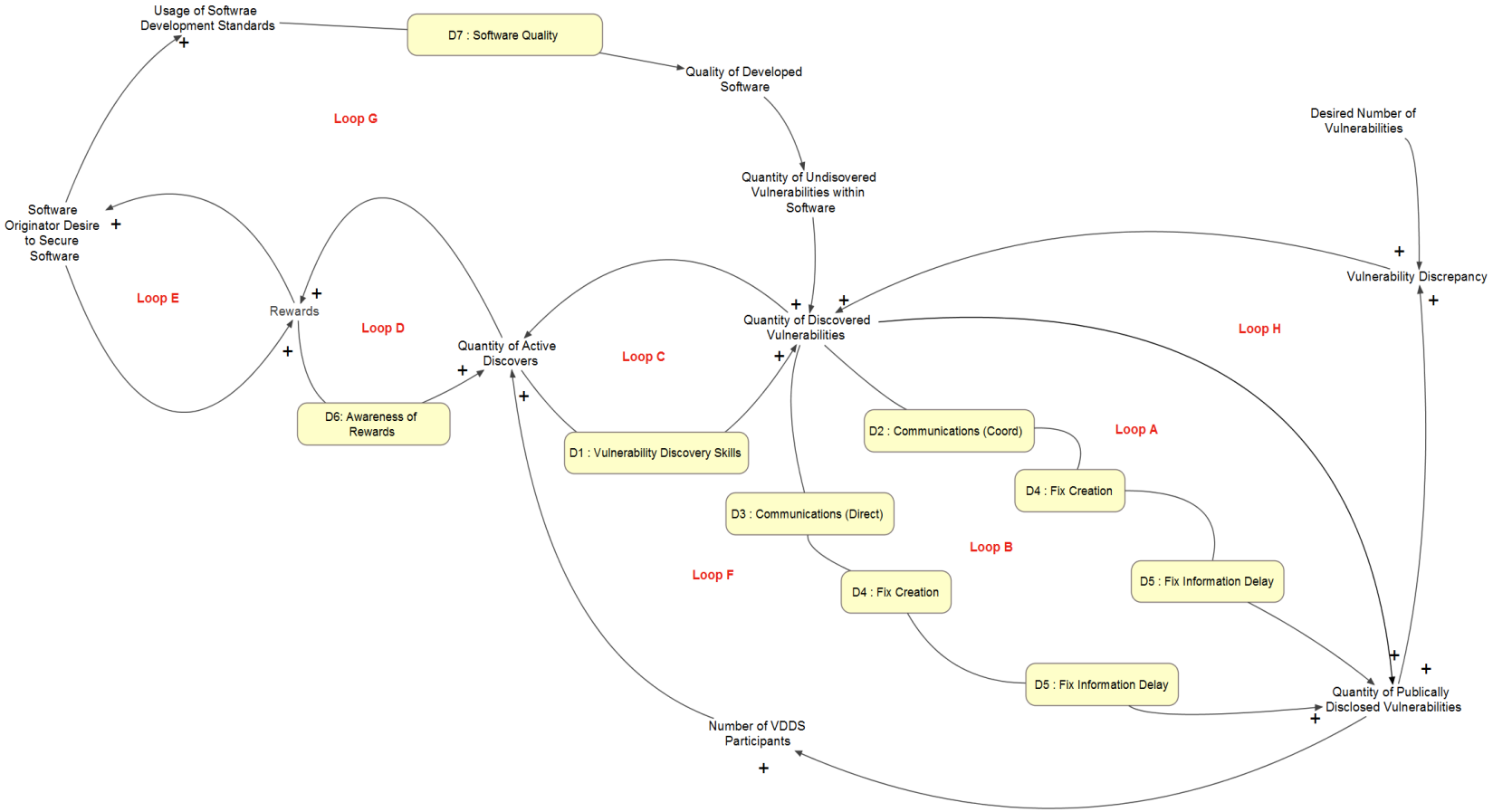


Figure 43 - High-level Causal Loop Diagram for Total VDDS

6.4 Total VDDS System Causal Structure

The VDDS is made up of interactions, loops and delays that have an impact upon local variables that feed the flow of vulnerabilities within their subsystem. However, the VDDS is not a system that can be considered in a reductionist manner, consequentially we must observe the whole VDDS model as a complete interacting system. To understand the causality of the variables and stock within the model we must perform a rigorous analysis upon the model structure and investigate the propagation of influence upon behaviours. To do this a technique known as causal tracing was used to trace the reason of any change to that variable or stock, by construction of a tree diagram (Voinov, 2008, p.53).

6.4.1 Discovery Subsystem Causality

The causal structure of the discovery subsystem is centred on the rate at which vulnerabilities are discovered. Inputs to this variable are highlighted in yellow, and consist of Discovered vulnerabilities, VDDS participants, discoverer fraction and undiscovered vulnerabilities. What stands out in the hierarchy is the set of variables that influence the rate of discovery and the interaction between them. For example, discovery rate feeds back upon itself, and forms part of the causal substructure, thus forming an integral part of the balancing and reinforcing feedback loops that are present.

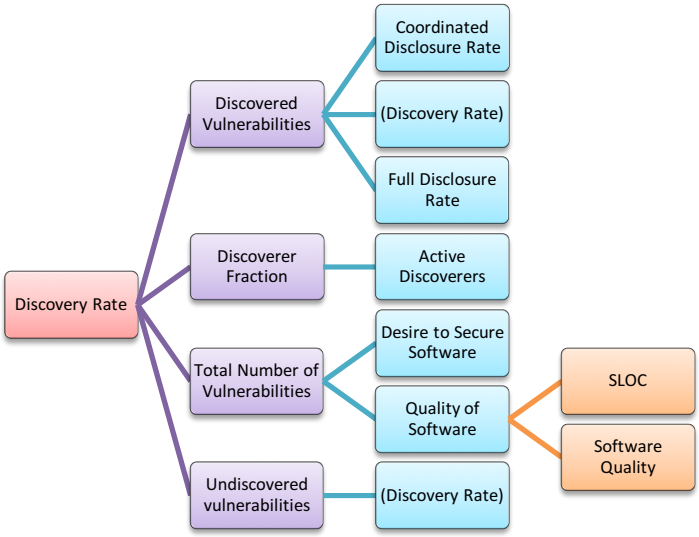


Figure 44 – Discovery Rate Causal Tracing Diagram

6.4.2 Disclosure Subsystem - Full Disclosure Flow Causality

The full disclosure flow within the disclosure subsystem provides the flow to the end state of public disclosure, in parallel to the coordinated disclosure flow. Again, the primary variable, full disclosure rate forms part of the causal substructure of the flow, influencing both undiscovered vulnerabilities and discovered vulnerability variables.

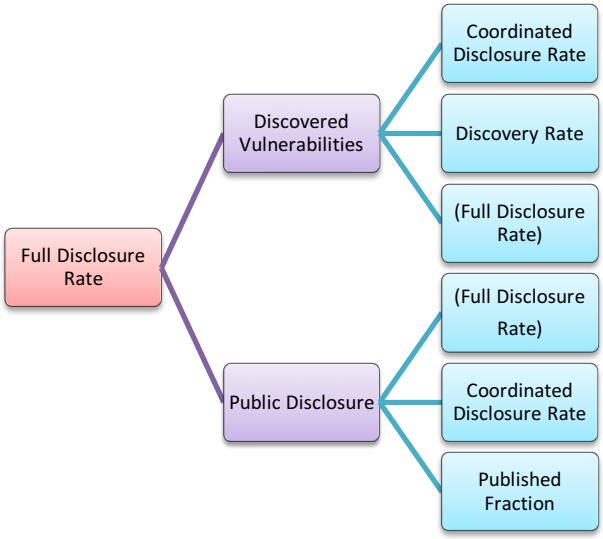


Figure 45 – Full Disclosure Rate Causal Tracing Diagram

6.4.3 Disclosure Subsystem: Coordinated Disclosure Flow Causality

Running in parallel to the full disclosure flow the coordinated disclosure flow provides a process route for vulnerability disclosure to occur. Again, the primary variable, coordinated disclosure rate forms part of the causal substructure of the flow, influencing both undiscovered vulnerabilities and discovered vulnerability variables. This is particularly important as the number of disclosures is mediated by the sentiment of the VDDS toward software originators.

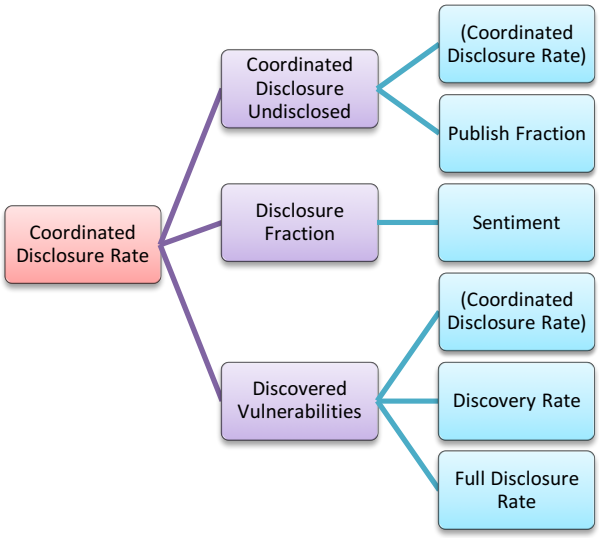


Figure 46 – Coordinated Disclosure Rate Causal Tracing Diagram

6.4.4 Discoverer Subsystem Causality

The final subsystem is arguably the primary engine within VDDS alongside the discovery subsystem. As such the variables that directly influence the flow are highlighted in yellow, with secondary and tertiary variables highlighted in blue and green. Again, the primary variable, conversion rate forms part of the causal substructure of the flow, influencing both potential discoverers and variables. This is particularly important as the number of disclosures is mediated by the sentiment of the VDDS toward software originators, increase the potential number of vulnerabilities.

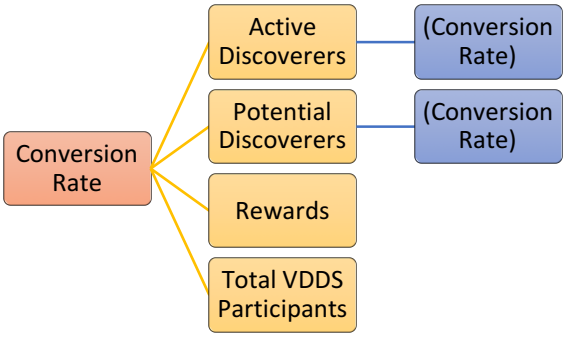


Figure 47 – Discover Conversion Rate Causal Tracing Diagram

6.5 Subsystem Influence Diagrams

Within each extension there are lower level subsystems which exist and generate output, which feeds into the higher-level variables, and ultimately the discrepancy between the desired state of the system (i.e. no vulnerabilities present) and the actual state of the system (number of publicly reported vulnerabilities). Therefore, an exploration of the subsystems that are present is required. Obviously, there is a level of detail that is required to ensure that the exploration is meaningful, yet not too reductionist as to distract from the overall goal of exploring the systemic behaviour.

6.5.1 Discoverers Subsystem

Rewards within the VDDS for discovery are a relatively new phenomenon with the first of the modern Bug Bounty schemes being founded in 2007 (Bradbury, 2007). Since the inception of the first scheme in 2007, the concept of legitimately being paid for the discovery of vulnerabilities has pervaded the discourse that surrounds the VDDS. Initially, large corporate entities such as Apple, Google and Oracle operated internal bounty schemes specifically tailored to their software offerings. This trend has culminated in the establishment of software vulnerability trading platforms that exist to broker vulnerabilities uncovered by discoverers and the originators of the software systems. In this model originators pay the broker to run a programme that advertises software for test, and payments are given for discovery. The rewards

that are given to discoverers are in part derived from the criticality of the vulnerability and impact of the vulnerability and range (Hackerone, 2017). The range of bounties that are given via the platform range from 'swag' (t-shirts, mugs etc), to $30,000 US dollars. Outside the platform company specific bounties, for example the famous pwn20wn competition hosted at the security conference CanSecWest can offer bounties in excess of $50,000 (Portnoy, 2010).

The reward system is a simple set of interactions between platform, software originator and vulnerability discoverer. The software originator sets the value parameters of the discovered vulnerability based on the variables `severity level`, `impact level` and factors that are internal to the software originator. Once set, software is advertised and if a vulnerability is uncovered this is then reported to the broker platform. If proven, then a bounty is awarded. This interaction is via the platform and may take between 1 day and 12 weeks (in some cases much longer). The broker interactions are also tailored to get the most proficient discovers to focus their attention on software originators who pay the most lucrative bounties for discovered vulnerabilities, therefore there is a propensity to increase the bounty over time. The price of the vulnerability is set by the software originator. The price is influenced by the variable number of active discoverers, and feeds back to the price of the vulnerability. The more discoverers there are, the lower the price, and with less discoverers the higher the price. This is an example of the supply/demand theory within basic economics, and has been applied to information security more widely (Anderson, 2001a; Anderson and Moore, 2006; Sloman and Smith, 2006, p.15). The variable `vulnerability price` is key to the subsystem as it sets the variables and influences the `quantity of active discovers` within the VDDS. The loop that is formed between the price that is set per vulnerability and the quantity of active discoverers is a negative balancing loop as an increase in discoverers lowers the price available as there is a larger pool of discoverers.

Alongside the legitimate trade in vulnerabilities between discoverers, brokers and software originators is the illegal trade of vulnerabilities between cyber

criminals and allegedly nation states (Ablon et al., 2017). The trade of vulnerabilities has been studied extensively, however is out of scope of the VDDS due to the opaque nature of the trade, and the scope of the ethical approval of this research (Fidler, 2014; Miller, 2007; Radianti, 2010a). However, to completely ignore the illegal trade in vulnerabilities would render the VDDS models that have been constructed incomplete. Hence the illegal trade of vulnerabilities has been factored into the VDDS model in the simple form of an exogenous information influence via the number of vulnerability discoverers whom are active within the VDDS and the discourse that surround the VDDS.

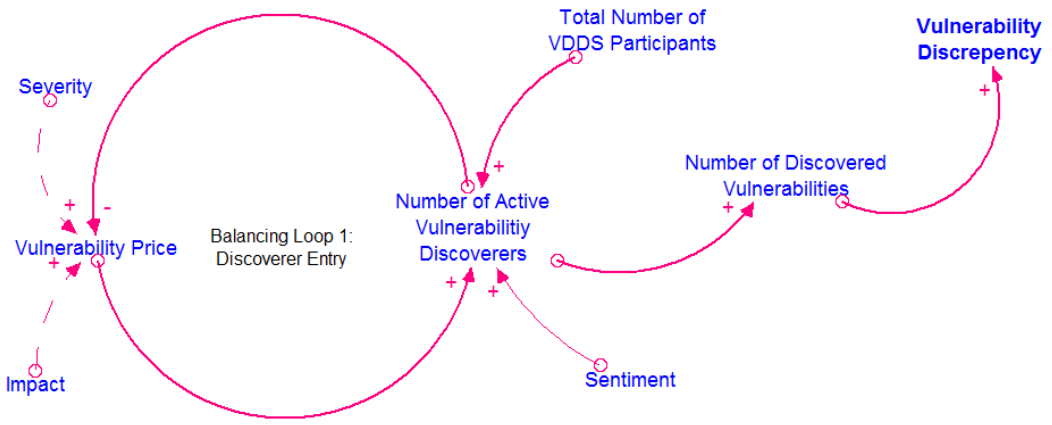


Figure 48 – Vulnerability Discoverers Subsystem Causal Loop Diagram

6.5.2 Disclosure Subsystem

The disclosure subsystem within the VDDS is separated into 2 flows, full disclosure and coordinated disclosure. To recap, full disclosure is the complete and immediate public disclosure of all details of a vulnerability without consultation or remediation. Coordinated disclosure is the gradual release of information about vulnerabilities either direct to the software originator or via a third party. Full disclosures typically have only one interaction, which is with the community that inhabits or surrounds the VDDS. Conversely the coordinated vulnerability disclosure stream is split into 4 parts; unpaid direct, paid direct;

unpaid indirect and paid indirect. Each disclosure path is chosen by the vulnerability discoverer and is based upon the personal economic and philosophical drivers, and influenced by influences, for example the discourse surrounding vulnerability disclosure and sentiment toward software originators. The rate at which the vulnerabilities are disclosed is based upon the number of undiscovered vulnerabilities within software and the incentives on offer.

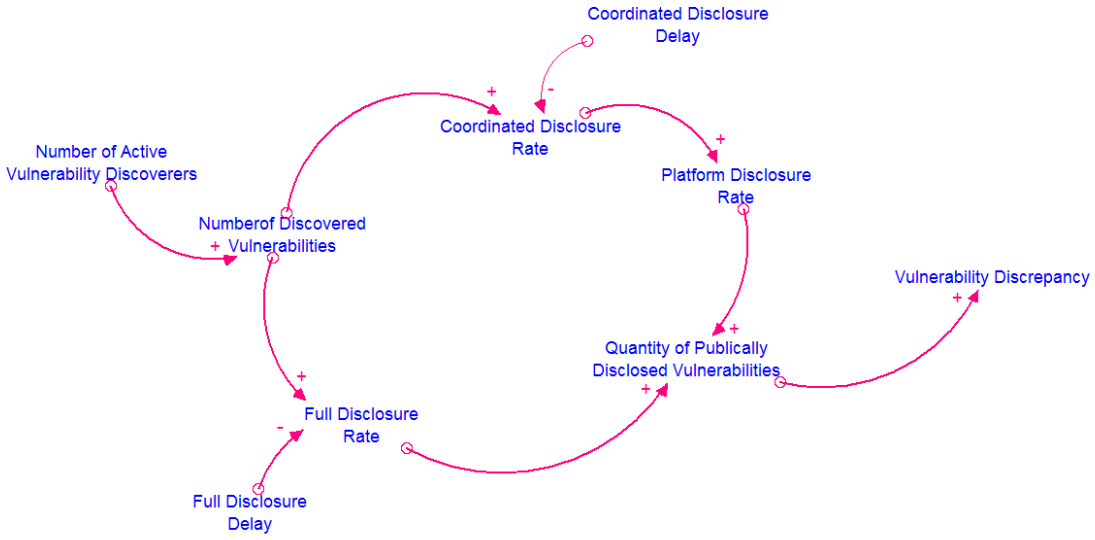


Figure 49 – Vulnerability Disclosure Subsystem

Incentives are influenced by the sentiment toward the software originator. If the sentiment is positive, then there is a higher chance of an *unpaid direct* disclosure occurring, with the *paid indirect* occurring if sentiment is negative. Alongside the process of disclosure are the interactions on the price upon a vulnerability discoverer may be receive one disclosed via a third party directly. The higher the price, the more chance the vulnerability discoverer will choose the paid routes to disclosure.

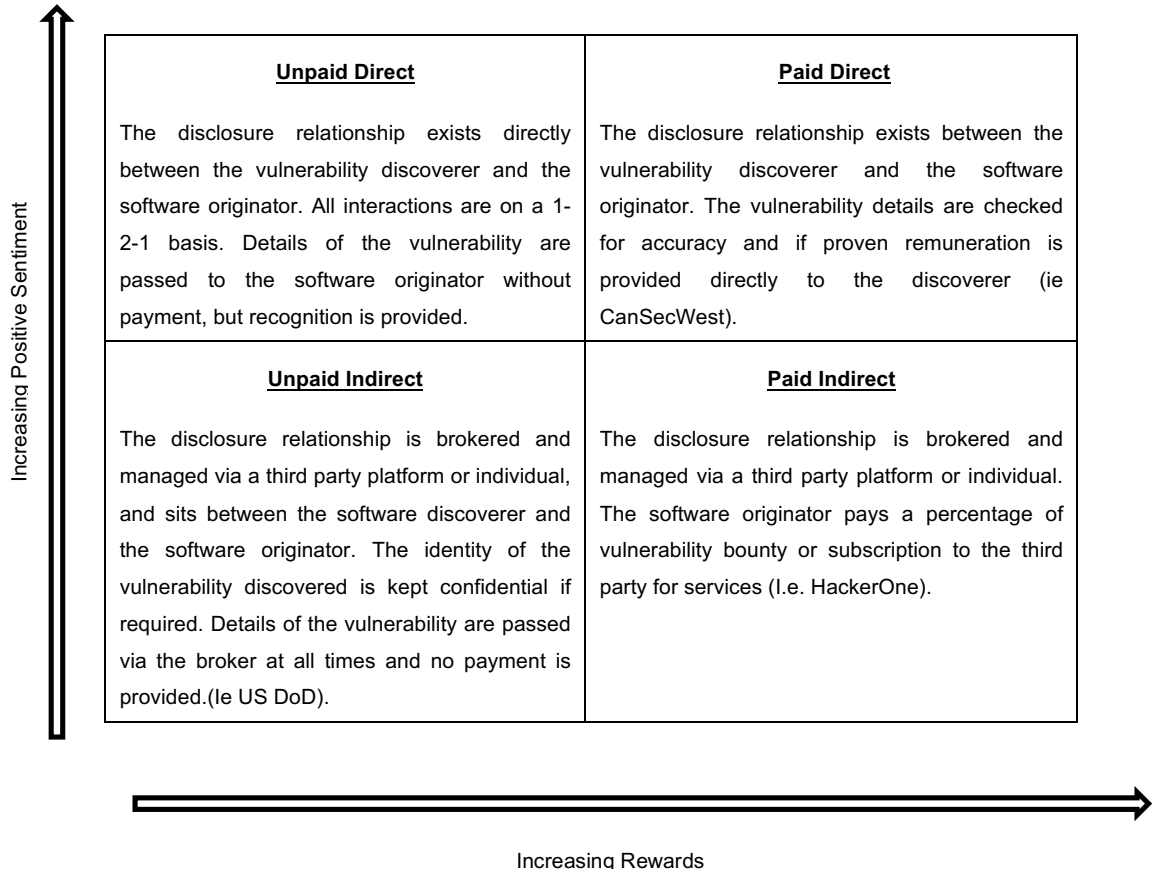


Figure 50 – Four Modes of Coordinated Vulnerability Disclosure

6.5.3 Software Quality Subsystem

The final subsystem that influences the VDDS is the quality of the software that exists within the system. The software originator produces software with vulnerabilities within it, and these are in turn discovered and disclosed over time. However, the rational desire within software originators is to release software without software vulnerabilities. Therefore, the pressure to reduce the number of vulnerabilities is a constant influence on all aspects of the VDDS. This desire shapes both the disclosure process, sets the price of the vulnerability, the availability vulnerabilities to find and impacts on the number of vulnerability discoverers within the VDDS. Consequentially, the quality of the software that is released is of paramount importance to the VDDS, as if the quality is high then there are less vulnerabilities to be found, and lower quantities of discoverers, disclosures and monetary rewards. This subsystem consists or a balancing loop with a delay, as the software quality processes take

both time both from a design and testing perspective. The quality of the software within VDDS is dependent upon the standards and engineers that create the software within the originator. The software engineers typically follow standard approaches when developing software, and use standard tools and development environments when developing. Alongside this, the competency of the software engineer is also a factor in the quality of the software that us produced. The interactions within the software quality subsystem are linear and do not contain any obvious loops or system archetypes, however there is a potentially significant delay in training or obtaining sufficiently competent software engineers to develop code. Moreover, the ability to create and adopt software standards also contains a time delay.

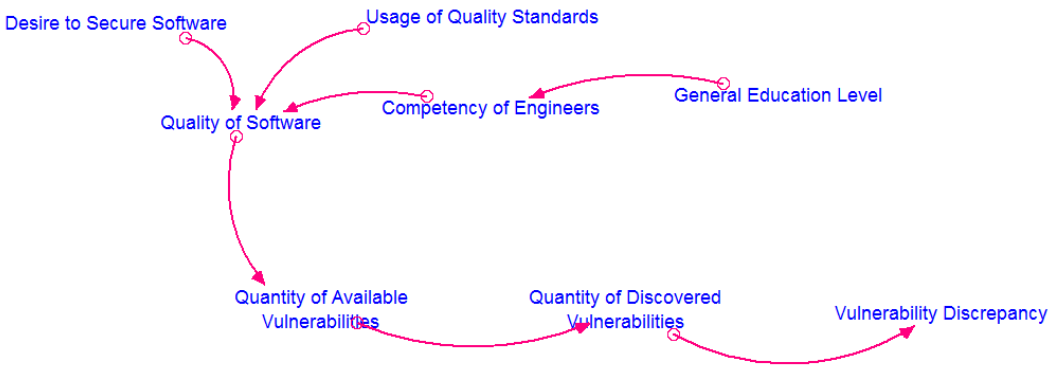


Figure 51 – Software Quality Subsystem

6.6 Adding Rates to the VDDS - Stock and Flow Diagrams

The construction of Stock and Flow Diagrams (SFD) is the penultimate phase of investigation of modelling the VDDS. SFDs characterise the state of the system, information flows around the in each time-period. Furthermore, SFDs allows the creation of rate equations to govern the movements of resources and information around the system. The initial construction of the SFD builds upon the series of CLDs and extends identified feedback loops, and identified variables. The fundamental system archetype that is central to the VDDS is the

interaction between two feedback loops, a balancing loop increasing the quantity of vulnerabilities within software, and a reinforcing loop removing the vulnerabilities. The interaction of these two feedback loops is classed as a second order non-linear archetype, whereby the balancing loop initially dominates and few vulnerabilities are found (Sterman, 2000, p.285). Once the balancing loop has iterated several times, the dominance begins to give way to the reinforcing loop, eroding the number of vulnerabilities that are to be found. This erosion slows the rate of discovery, and consequentially the rate of growth plateaus, with an inflection point marking the shift in dominance from balancing to reinforcing loop.

6.6.1 Discovery Stock and Flow Diagram

Observing the total VDDS reference mode, we can see there is the potential for a fundamental pattern of behaviour exhibited by the VDDS. Alongside this behaviour, are two well-known archetypes, as shown by the CLD – in particular 'fixes that fail' and 'limits to growth' (Senge, 1990, p.379 and 388). Both archetypes exhibit behaviour that is like the total system VDDS reference mode, however, as they interact with each other they can be considered a summation of both. The limits to growth archetype corresponds to the discoverers the sub-system and disclosure subsystem. Fixed that fail corresponds to the quality subsystem and curiously, disclosure.

Hence the investigation of the exhibited archetypes using SFDs is a critical component to the characterising the behaviour of the VDDS. The archetypes consist of two related stocks, `discovered vulnerabilities` and `undiscovered vulnerabilities`. The rate at which vulnerabilities are discovered is directly controlled by the `total number of vulnerabilities within software` and the `quantity of active vulnerability discoverers`. Both variables directly control the rate upon which vulnerability discoverers find issues within software. Present within the SFD are two secondary variables `Rewards` and `Sentiment`. Both variables influence the quantity of active discoverers by increasing the efficiency, or rate which vulnerabilities are discovered per unit time. Together these variables have been labelled as a concept notionally termed within this

investigation as *motivation* as these drive the discoverer to uncover vulnerabilities.

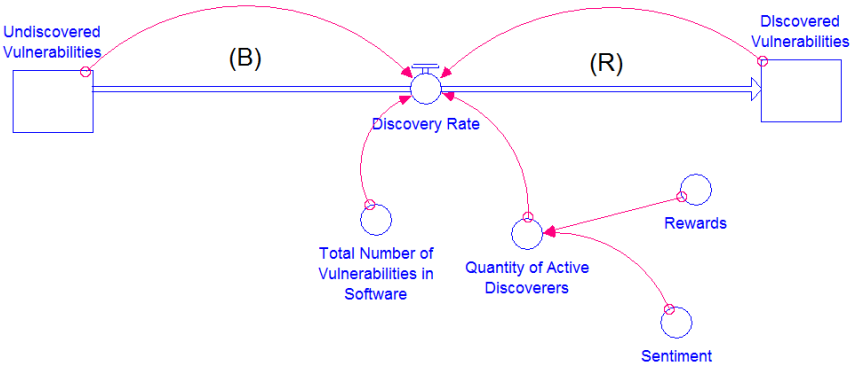


Figure 52 – Basic VDDS Archetype

Each motivating factor changes over time as the sentiment or rewards for a high severity vulnerability, drives the motivation discoverers have in uncovering issues. Finally, the rate of vulnerability is the conversion factor of turning undiscovered vulnerabilities into discovered vulnerabilities, but not disclosed. Alongside these variables, resources flow back to the rate of discovery insofar as the number of vulnerabilities that have been discovered reduces the available number that are available for discovery. The reduction is the product of the balancing loop which is dominant towards the end of the discovery time.

Within this subsystem there are several assumptions that have been made. Specifically, the number of vulnerabilities that are present within the software remains static and once discovered are removed from the software and not reintroduced. This is modelled on the software reliability concept known as perfect repair (Lin, 2011). Moreover, all vulnerabilities are treated equally, with attributes such as severity being discounted. Finally, the archetype does not include identified delays and all vulnerabilities are processes in serial. These simplifying assumptions allow for the exploration of the simple vulnerability discovery process, and more importantly demonstrate that a simple set of balancing and reinforcing loops can create a complex nonlinear phenomenon.

6.6.1.1 Discovery Subsystem Equations

Equation 8 shows a set of simple equations for the conversion of Undiscovered Vulnerabilities (U) into Discovered Vulnerabilities (D) with the flow between both stock is governed by the Discovery Rate (DR) equation. Vulnerabilities flow from Undiscovered to Discovered when a discoverer finds a vulnerability, increasing the stock level. This then depletes the number of available vulnerabilities that are to be found. Vulnerabilities are found at a specific rate, known as discovery rate, measured as found vulnerabilities per month, or (1 divided by month). Thus, the undiscovered vulnerabilities generate U_{dr} discoveries per month. Additionally, the probability of finding a vulnerability is governed by the number of total number of vulnerabilities within the software (S) and number of discovered vulnerabilities (D) giving D divided by S (D/S). The discovery rate (DR) is therefore the total number of discoveries (U_{dr}) multiplied by the undiscovered vulnerabilities (U) and quantity of discovers (Q) multiplied by Discovered vulnerabilities over the total number of vulnerabilities in the software (D/S).

$$D(t) = D(t - dt) + DR\ dt$$

$$DR = (U\ Q)\frac{D}{S}$$

$$U(t) = U(t - dt) + (-DR)\ dt$$

$$Q = DR(1 - s)$$

Equation 8 – Discovery Subsystem Stock and Flow Equations

6.7 Whole System Stock and Flow Diagram

Having defined the fundamental building block of vulnerability discovery using both causal loop and stock and flow diagrams, I will now move to integrate the remaining subsystems, discoverer, quality and disclosure into the model. The causal loop diagrams detailed in sections 6.5.1 – 6.5.3 show the interactions between variables, and balancing or reinforcing loops. Nevertheless, to explore the model completely stock and flow diagrams with accompanying rate equations will now be presented.

6.7.1 Discoverer Subsystem Stock and Flow Diagram

The discoverers sub system is a 2nd order nonlinear system representing the number of vulnerability discoverers that are active within the VDDS. The subsystem acts as a catalyst in the discovery of vulnerabilities within the subsystem as a vulnerability cannot be discovered without an active discoverer. Given this importance, the rate at which potential discoverers are converted to active discoverers is equally important from the general VDDS population. The rate upon which active discoverers are generated is based upon two variables, `Total number of participants` within the VDDS and `Rewards` available to the discoverer. These variables are located within balancing and reinforcing loops oscillating between dominance and submissive archetypes. The initial phase of the conversion is dominated by a reinforcing loop that drives the conversion of general VDDS participants to active discoverers. This conversion takes place slowly at first, then accelerates toward complete conversion of all potential discoverers. In this phase awareness of the vulnerabilities within the software is low, the rewards that are available is again low therefore there is a low number of discoverers available. As time progresses the potential rewards that are on offer drive the rate of conversion forward, and the greater the potential rewards that are on offer. The actual rewards that have been awarded are noted in section 5.4.6 and give rise to new potential discoverers entering the VDDS over time as the motivation increases.

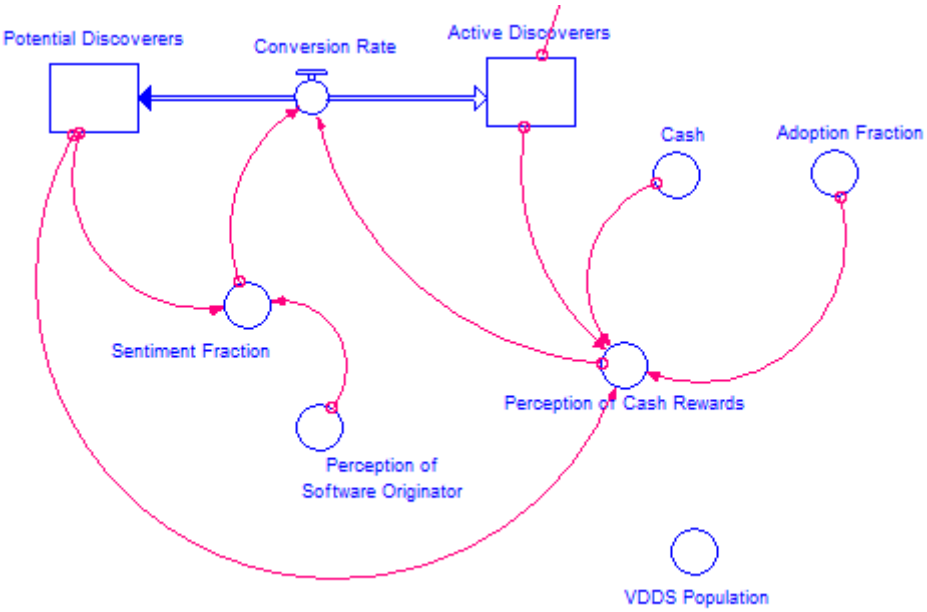


Figure 53 – Discoverers Subsystem Stock and Flow Diagram

6.7.1.1 Discoverers Subsystem Equations

In a similar manner to the discovery building subsystem outlined in section 6.7.1 the discoverer subsystem is driven by the number of available discoverers. Equations 9 – 11 show a simple set of equations for the conversion of VDDS Population (TP) to Active Discoverers (AD). The flow from VDDS participants to Discoverers is governed by the Conversion Rate (CR) equation, with VDDS participants flowing from the potential discoverers stock (PD) to the Active Discoverers (AD) stock. Additionally, the conversion rate is also mediated by the Cash on offer (CA), resulting in a fraction increasing the Perception of Cash Rewards (PR) and the Sentiment fraction (SF). The conversion rate is the sum of conversions resulting from cash rewards, and implicit word of mouth adoption (adoption fraction constant). The conversion rate is also based on the sentiment toward the software originator and perception which are again constant within the model. Hence conversion rate is simply:

$$CR = SF + PR$$

Equation 9 – Conversion Rate

$$SF = PD\ SF$$

Equation 10 – Sentiment Factor

$$PR = \frac{CA\ AF\ PD\ AD}{TP}$$

Equation 11 – Perception of Cash Factor

Both stocks feed into the conversion rate. As indicated previously the dominance of the balancing feedback loop at the initial phase of the conversion process is in part dependent upon the low levels of rewards and knowledge of vulnerabilities. Given these factors a large pool of potential discoverers that can be converted into active discoverers, search for vulnerabilities within the software. This conversion continues to increase till the quantity of available converts approaches a level whereby the dominance moves to the reinforcing loop, and continues to deplete the availability of potential discoverers at a slow rate. In turn the increase in active discoverers impacts upon the rate of discovery.

6.7.2 Software Quality Subsystem Stock and Flow Diagram

Numerous studies have stated that software quality metrics can be used to quantify the number of software vulnerabilities within software (Concas et al., 2007; Garrison, 1993; Mohagheghi et al., 2004; Nguyen et al., 2010; Shin et al., 2011) Moreover, studies to undercover the correlation of Complexity, Cohesion and Coupling (CCC) metrics (commonly known as the Chidamber-Kemerer (CK) object orientated metric suite) with the number of vulnerabilities within software have also been undertaken (Chidamber et al., 1994; Chowdhury et al., 2011; Lewis et al., 2015; Moshtari et al., 2013; Shin et al., 2011). Given this relationship the number of vulnerabilities that are within software and available for discovery is related to the quality of the software and the desire for the

software originator to produce vulnerability free software (Alhazmi et al., 2005). Here the assumption is made that the software originator has made a rationale choice that there is a complete desire to reduce the quantity of vulnerabilities to zero. Whilst this is practically impossible to achieve, the desire to achieve this is not. As outlined in section 5.4.3.1 the vulnerability density of software is given by equation 12, calculated by using the number of reported vulnerabilities and the size of the software. Using vulnerability density as a proxy, quality of the software ranges from low quality to high quality, represented as the vulnerability density of the software [0-1] calculated via million source lines of code (MSLOC) and number of vulnerabilities publicly disclosed.

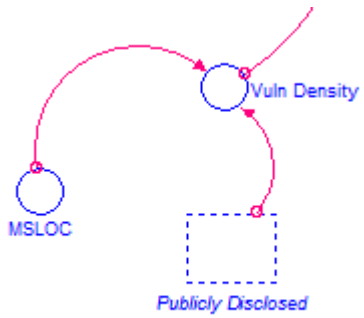


Figure 54 – Software Quality Subsystem Stock and Flow Diagram

6.7.3 Software Quality Subsystem Equations

Figure 55 below shows two flows representing the quality of software with vulnerabilities residing within it. Vulnerability Density (VD) of the software varies all of the time, with a low density towards the beginning of the life of the software, and increasing towards the end of the lifecycle. Hence the density is governed by the size of the software, measured in million lines of source code (MSLOC) divided by publicly disclosure vulnerabilities (PD), resulting in PD/MSLOC. The desire to security the software is wrapped within vulnerability density.

$$VD = \frac{PD}{MSLOC}$$

Equation 12 - Software Quality

Delays within the system exist between the number of discovered vulnerabilities and the quality of the system, and therefore the number of vulnerabilities within the system. The delay is caused by the desire of the software originator to fix issues in a controlled manner, and the rate at which vulnerabilities are discovered, thus impacting the perceived quality of the software.

6.7.4 Disclosure Subsystem Stock and Flow Diagram

The disclosure subsystem reflects the choice which a vulnerability discoverer should make when deciding upon a course of action once a vulnerability is discovered. The disclosure subsystem within the VDDS is separated into 2 streams, full disclosure and coordinated disclosure. To recap, full disclosure is the complete and immediate public disclosure of all details of a vulnerability without consultation or remediation. This is a 3rd order nonlinear system that consists of `Discovered Vulnerabilities`, `Disclosure Platform` and `Publicly Disclosed` stocks. The initial stock that receives vulnerabilities into the subsystem is the `Discovered vulnerabilities` stock. The stock is directly connected to the rate of both full disclosure and disclosure platform stocks via the flow or vulnerabilities. Both the full and Platform disclosure flows are mediated by the delay which is encountered. Vulnerabilities flow around the disclosure subsystem as discrete units, processed one at a time, however there can be multiple instances of the process occurring in parallel.

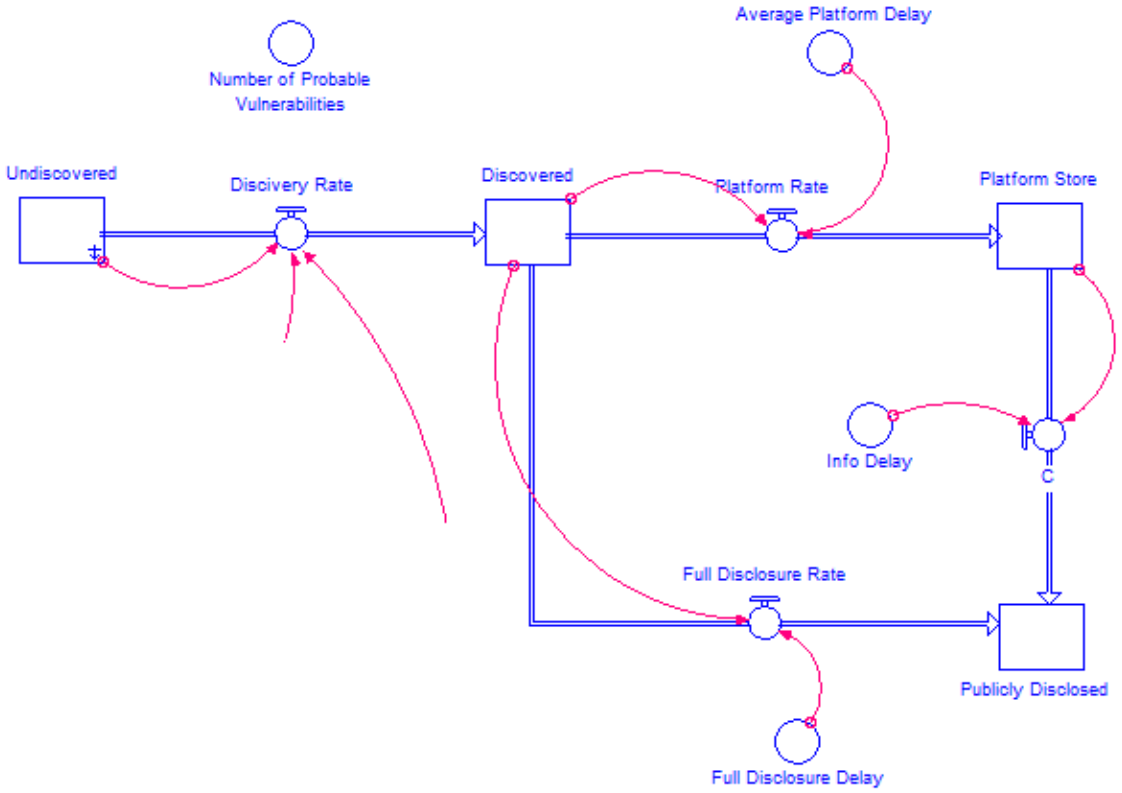


Figure 55 – Disclosure Subsystem Stock and Flow Diagram

6.7.4.1 Full Disclosure Flow

The full disclosure flow is connected directly to the publicly disclosed vulnerabilities stock, which is the final stage of the disclosure process. The disclosure rate is ruled by the fractional rate of disclosure which consists of the sentiment towards the software originator. Minimal delays exist within this flow as the choice to disclose by the discoverer can be considered almost instantaneous once the decision has been made. Given the instantaneous nature of the flow a potentially large amount of vulnerabilities can transit directly from discovered to publicly disclosed quickly. The rate that vulnerabilities flowing from the discovered stock consists of a disclosure fraction, and again consists of two feedback loops consisting of a reinforcing loop and balancing loop.

6.7.4.2 Full Disclosure Flow Equations

Equation 13 shows a very simple equation for the full disclosure rate that is encountered when a vulnerability is disclosed via the full disclosure route. The

flow is governed by the full disclosure delay rate (FDD) equation, with vulnerabilities flowing from the discovered (D) stock directly to the publicly disclosed stock, measured in vulnerabilities per month. The discovers per month is divided by the per unit time (FDD), thus D / FDD.

$$FD = \frac{D}{FDD}$$

Equation 13 – Full Disclosure Rate

6.7.4.3 Platform Disclosure Flow

Coordinated disclosure is the alternative flow that takes place in the disclosure subsystem. The main flow of vulnerabilities exists once a vulnerability discoverer has decided to coordinate the disclosure of a vulnerability either via a vulnerability disclosure platform or directly to the software originator. Both choices have been expressed in the SFD as a single flow that is mediated by the coordinate disclosure rate and the coordinated publish rate. Both rates differ insofar as they exist at opposite ends of the coordinated disclosure process, with published rate existing at the end of the process, penultimate to the public disclosure stock. The rate which vulnerabilities flow is governed by the delays between the discovered state and the coordinated delay stock, which acts as a 'buffer' and is typically located within the vulnerabilities disclosure platform or within the software originator processes.

6.7.4.4 Platform Subsystem Equations

The disclosure subsystem contains three stocks, Discovered Vulnerabilities (DV), Disclosure Platform Store (PS) and Publicly Disclosed (PD). Like the discovery subsystem, flows from the discovered stock are mediated via rate equations. The platform flow consists of two rate equations. Platform Disclosure Rate (PDR) consists of number of vulnerabilities disclosed within the D stock (D), divided by the average platform delay time (PDT) and importantly. Similarly, the Coordinated Disclosure Rate (C) is mediated by Disclosure Platform Store (PS) and Information Delay Time (IDT). This subsystem assumes that a vulnerability cannot be rediscovered once publicly disclosed. Coordinated

Disclosure Rate (PDR) and Platform Disclosure Rate (C), which are commonly known as removal rates shown in equations 14 & 15.

$$PDR = \frac{D}{PDT}$$

Equation 14 – Platform Disclosure Rate

$$C = \frac{PS}{CDT}$$

Equation 15 - Coordinated Disclosure Rate.

Figure 56 shows the full VDDS system with all variables, relationships and flows in place. We can clearly see the different subsystems, and flows around the VDDS.

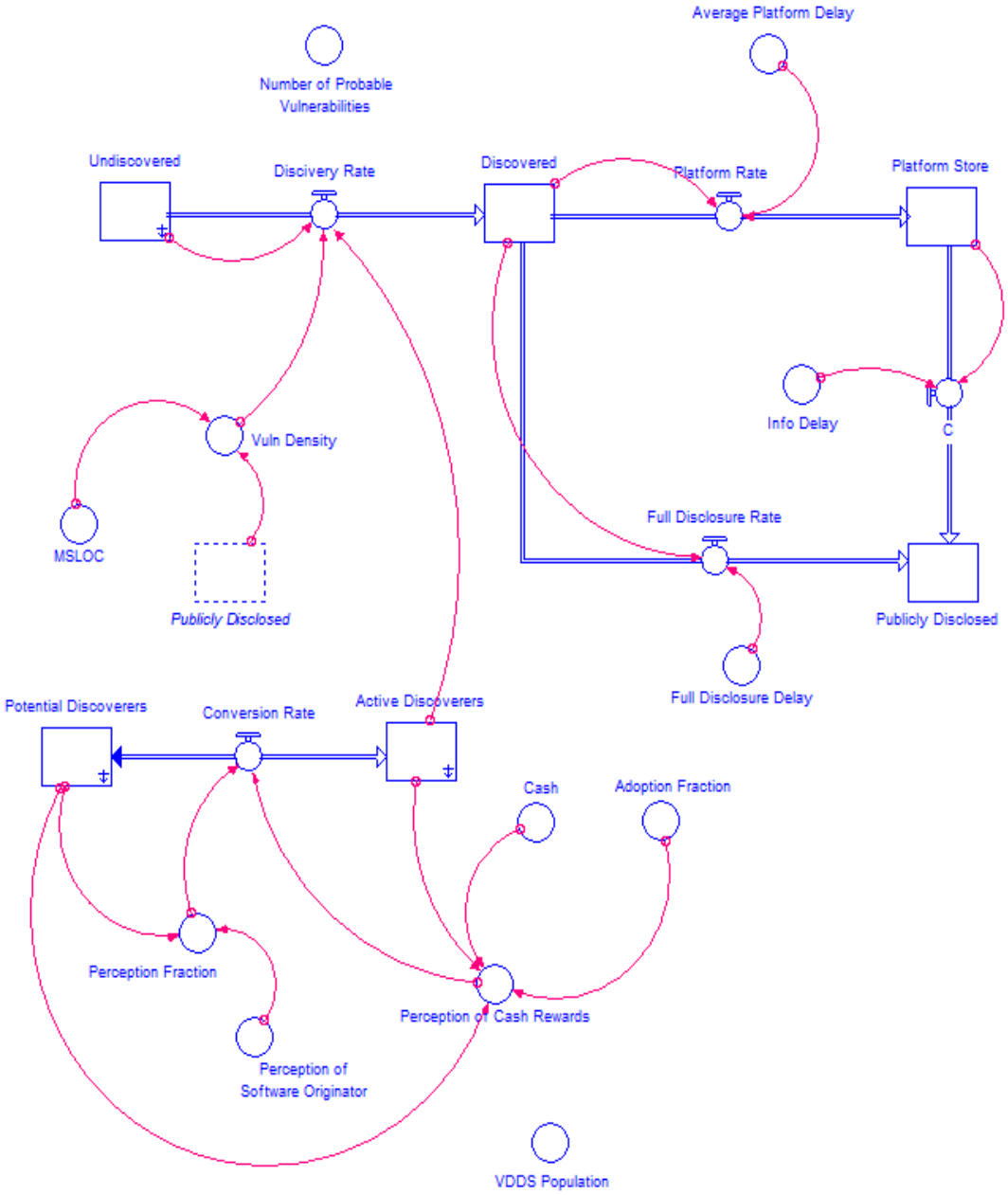


Figure 56 – Full VDDS Stock and Flow Diagram

6.7.5 End State - Public Disclosure Equation

The ultimate end state of the VDDS is the public acknowledgement of the vulnerability details, which may or may not result in a fix to the vulnerability. All vulnerabilities that are present within the system result, albeit after a potentially significant delay are recorded by the National Vulnerability database or via the software originator. In extreme edge cases the vulnerabilities may enter the VDDS and may never reach the end state due to inadequacies in software originator process (i.e. lost communication) or discoverer disinterest. Given this, the end stock of `publicly disclosed` must be in equilibrium to the discovered `vulnerability stock`. This equilibrium may take several months and years to result, but will ultimately occur due to the natural draining of the VDDS.

$$PD(t) = PD(t - dt) + (FDR + PDR)\, dt$$

Equation 16 – Publicly Disclosed

6.8 Model Validity and Testing

When considering any model that has been constructed from empirical observation it is necessary to test the validity and robustness of that model and check the utility of it under extreme conditions. The VDDS model is no exception to this rule, and as such a suite of tests have been performed to ensure the efficacy of the model. The accuracy of the qualitative aspects of the model have been validated via chapters 4 and state variables via chapter 5, what now follows is a set of test to check the validity of the model. Sterman (2000 p.859-p.861) provides a comprehensive list of tests that can be applied to systems dynamic models. These tests range from model boundary adequacy to sensitivity analysis and provide confidence that the model under test is accurate and reflects reality as precisely as possible. From the list provided by Sterman a selection of the most applicable were chosen and is given in Table 42 below.

Test	Purpose & Checks	Procedures for Checking	Addressed In Sections
Boundary Adequacy	1. Important concepts addressed? 2. Behaviour of model change when boundary changes?	• Model boundary charts • Subsystem Diagrams • Causal, Stock and Flow Diagrams	**6.2.4, 6.3.1** **6.4.1 – 6.7.5**
Structure Assessment	1. Structure consistent with problem description? 2. Model conform to basic physical laws? 3. Decision rules capture behaviour of the system?	• Causal and Stock and flow diagrams • Interviews, workshops, expert opinion • Partial model tests	**Chapter 46.4.1 – 6.7.5 &Chapter 7**
Dimensional Consistency	1. Are the equations consistent without use of parameters without real world meaning?	• Direct inspection of equations	**6.4.1 – 6.7.5**
Parameter assessment	1. Are parameter values consistent with relevant descriptive and numerical knowledge of the system? 2. Do parameters have real world counterparts?	• Use statistical model to estimate parameters • Use partial models to calibrate subsystems • Use judgemental methods based on interviews and expert opinion	**Chapter 5**
Extreme Conditions	1. Does each equation make sense when inputs are extreme? 2. Does the model respond plausibly when subjected to extreme policies, shocks and parameters?	• Equation inspections • Test response to extreme values • Subject model to large shocks and extreme conditions • Examine conformance to basic physical laws (no inventory etc)	**6.4.1 – 6.7.5** **Chapter 7**
Sensitivity Analysis	1. Numerical sensitivity, do the numerical values change the behaviour of the system significantly when assumptions are altered? 2. Behavioural Sensitivity, do the modes of behaviour change when assumptions are altered? 3. Policy Sensitivity do policy implications change when assumptions are altered?	• Perform univariate and multivariate sensitivity analysis • Use analytic methods such as global stability analysis	**Chapter 7**
Behaviour reproduction	1. Does the model reproduce the behaviour of the phenomena under investigation? 2. Does it endogenously generate symptoms of difficulty motivating the study? 3. Does the model generate the various modes of the system? 4. Do frequencies and phase relationships among variables match the data?	• Compute statistical measures of correspondence between model and data; descriptive statistics; • Compare model output to and data quantitatively including modes of behaviour, shape of variables etc.	**Chapter 7**
Behaviour Anomaly	1. Do anomalous behaviours result when assumptions of the model are changed?	• Replace equilibrium assumptions with disequilibrium assumptions	**Chapter 7**

Table 42 – Model Tests Source :Adapted from (Sterman, 2000, pp.859–891)

Model checking and testing was performed throughout the collection of data, production of analysis and construction of the VDDS model in vivo. As such the steps that address the model tests are indicated in the rightmost column of Table 42 above, with references to the appropriate section.

6.9 Chapter Summary

Thus, far this thesis has argued that both causal loop and stock and flow diagrams can adequately represent the complexity of the VDDS. This chapter has described the methods used in this investigation and has shown the structure of the VDDS, interactions between entities and introduced delays which are analogous to delays in processing vulnerabilities. It has also shown the differing interactions between four subsystems and how those sub systems interact. A key system archetype that is prevalent throughout the VDDS was used to represent how the flow of vulnerabilities are mediated and governed throughout the VDDS. Two fundamental archetypes (Limits to Growth and fixes that fail) characterises the interactions of the sub systems. Within reinforcing loops depleting the initial stock of vulnerabilities and discoverers and passing these to the subsequent stocks of disclosed vulnerabilities or active discoverers. Furthermore, a suite of tests was provided and evidence of the robustness of the model given. The next chapter describes the procedures and methods to evaluate and test both the model and the hypotheses presented in chapter one – what is driving the increase of vulnerabilities with commercial off the shelf software.

7 Simulation and Analysis

The aim of this investigation is to understand the factors that underlying the increase of vulnerabilities within the vulnerability discovery and disclosure system. This chapter now turns to exploring the numerical and policy sensitivity of the constructed VDDS model. To do this quantitative data was drawn from empirical observations (chapter five) indicating behaviour and initial parameter space. I will present the principle findings of both current policy decisions that shape the existing system and potential strategies that may improve the status quo. These policies do not only suggest differing values for current variables and rates, but new process flows and stocks to reduce vulnerability centred risk.

To restate the main aspects of this investigation outlined in in the initial chapter:

What are the constituent parts, dynamics and structures of the COTS legitimate vulnerability discovery and disclosure system?

And;

What are is the potential impacts of strategies employed to reduce vulnerability centred risk?

It is the latter part of this research question that is the focus of this penultimate chapter. To observe the behaviour of the VDDS under simulated conditions allows the exploration of key interactions and variables. This exploration can provide the basis for the development of new policies and strategies for dealing with vulnerability based risk going forward. Given this need the appropriate strategies have been selected to both test the model for real world comprehension, and record how changes in in key variables affect the simulated outcome.

7.1 Simulation Scenarios

The vulnerability discovery and disclosure system is influenced by endogenous variables alongside the exogenous variables sitting outside the system boundary. As such the endogenous state variables form part of the policy makeup, and influence the behaviour of the system. To understand how the

system behaves in the existing, or baseline configuration of the model real world values a series of initialisation values have been selected from previous analysis. These values are derived from empirical observation from previously published work, and via Chapters four and five.

To explore the model three simulation scenarios have been chosen. These scenarios have been selected to represent plausible scenarios that could be adopted by entities whose impact on the VDDS is material. The initial simulation (not including the calibration and baseline simulation) is the variation of the rewards that are available for discovery of vulnerabilities. As stated in chapter four, thematic analysis, the provision of monetary rewards for details of vulnerabilities is a relatively recent phenomenon. Nevertheless, the advent of the rewards system has generated an increase in the number of discoverers that have entered the VDDS as active discoverers and provided resources to devote more effort to finding vulnerabilities. Given this increase in rewards, and the recompense that is on offer, it is logical to assume that the larger, or more readily available the rewards that are, the greater the number of vulnerabilities that will be discovered via the coordinated disclosure flow.

The second scenario that has been chosen is the varying of the quality of the software that contains vulnerabilities. The concept under investigation within this scenario is the premise that the higher the quality of software that is produced, the lower the number of inherent vulnerabilities that are available for discovery and hence the lower potential risk. Thirdly, a key aspect of the VDDS is the social or human characteristics within it. As such the notion of sentiment was identified, and is us used as a proxy for theme known as perception of punishment (sections 4.7.3 and 5.4.1) as key to the motivation and the subsequent decision points within the discoverers place within the VDDS. Aspects under investigation are the variation and impact of both positive and negative sentiment upon the VDDS and the outcomes forthwith. Finally, the last scenario is the elongation and reduction of delays within the VDDS. As shown, delays in communication between software discoverers and software originators

are a source of frustration, and ultimately risk, as the default position of full disclosure is often exercised when tensions overflow (Dan, 2011).

The VDDS can be split into 3 specific epochs when characterising the evolution of the system. The initial epoch can be describing as uncontrolled and constrained growth of vulnerabilities, with the time span running from the mid 1990's through to approximately 2009-10. This defining characteristic of epoch 1 is the demonisation of the vulnerability discoverer, and increasing dependence of society upon a vulnerable information system to conduct commerce and communications. Epoch 2, commences around 2010 with the advancement of vulnerability disclosure platforms, and a change in perception towards vulnerability discoverers from software originators and rewards being offered. The defining characteristic of epoch 2 is the emergence of new markets for vulnerabilities and associated rewards. Epoch 2 is currently underway. Epoch 3, is speculative and almost certainly consists of new markets and services, with an acknowledgement that vulnerabilities are central to the risk mitigation process. Epoch three may take the form of one of the three possible futures, flat, steady or exponential dependent upon the conditions of the system at that time.

7.1.1 Simulation Tool

The simulation tool that was chosen to construct both the CLD, SFD and undertake simulation runs was ISEE Systems iThink v10.0.6. iThink is an industry standard tool to construct system dynamic models, and undertake simulations to investigate the behaviour of implemented models.

7.1.2 Sensitivity Analysis

Sterman (2000, p.883) argues that all models are wrong, and so require testing to establish the robustness of models, and confidence in any conclusions made. Therefore, in addition to the comprehensive set of robustness tests outlined in Table 42, a set of analytical tests to characterise the sensitivity of policy and numerical assumptions is presented. The numerical or parameter sensitivity tests are performed by varying a local variable parameter and noting how the

behaviour changes (Hekimoglu and Barlas, 1996). Parameters in the case of the tests that have been used to test the VDDS model are both incremented within the parameter space of the variable, or drawn from observed probability distributions.

7.2 Baseline Simulation

Establishing a baseline of how the model behaves, with normal, or average parameters is of importance. However how the baseline reflects the reality of the VDDS in the real-world is of critical importance. As such values were selected from the exploratory data analysis thus providing an observed set of initial conditions upon which to explore the VDDS model. The initial conditions for the baseline simulation run are presented in Table 43 below, showing values for all state variables. The initial values of these parameters were chosen to be the most representative of an average software application, in our case Microsoft Windows NT 4.0. The values provide both an average set of values to explore the dynamics of the system, and to gain a comparison of future simulation runs. Undiscovered vulnerabilities exist within the software, and are latent within the code since the primary compilation of the software, therefore this value is set to 212, which is the total number of detected vulnerabilities within the Microsoft Windows NT 4.0 software application (Lewis et al., 2015). The desire to secure the software from the outset by the software originator is set at 50%, with delays corresponding to the average recorded in chapter 5, exploratory data analysis. The parameters to initialise the system were set at rewards, $1012.00 and sentiment, 50% with the caveat that rewards of any kind, other than good will, did not exist when the Microsoft Windows NT 4.0 discovery period was active. The size of Windows NT 4.0 has been estimated to be around 11 million source lines of code. The run time for the simulation was set at 240 months (10 years), and delta time set to 0.25. An overview of the initialisation values drawn from chapter five and previously publish work is provided in Table 43 below.

Local Variable Name	Initial Value	Units	Section or Reference Source
Undiscovered vulnerabilities	212	Vulnerabilities	(Lewis et al., 2015)
Discovered vulnerabilities	1	Vulnerabilities	N/a
Sentiment (perception)	50%	Dimensionless	Section 5.4.1
Potential Discoverers	100	People	Section 5.4.2
Discoverers (initial value)	1	People	N/a
Total Number of VDDS Participants	10000	People	Section 5.4.2
Rewards	$1012.00	Dollars	Section 5.4.6
Full Disclosure Delay	1	Days	Section 5.5
Coordinated Disclosure Delay	1	Days	Section 5.5
Coordinated Platform Disclosure Delay	30	Days	Section 5.5
MSLOC	11	Million Lines of Code	(Lewis et al., 2015)
Number of VDDS Participants	10000	People	Estimated

Table 43 - Simulation Initial Condition Parameters

The initial simulation run using values from Table 43, show several behaviours of the state variables, Disclosure Platform, Undiscovered, Potential and Active Discoverers and Publicly Disclosed Vulnerabilities. Initially the Undiscovered vulnerability stock depletes quickly (200 by month 12) as there are numerous vulnerability discoverers available to find issues within the software. The flow from the undiscovered stock continues to approximately month 240, where an increase of the vulnerability discovery rate decreases from 20 vulnerabilities discovered every month to less than 1 per year. This decreases the stock depletion rate significantly, resulting in a tipping point whereby the undiscovered stock depletes and the balancing loop starts to dominate around month 12. The undiscovered stock is completely depleted at month 240, with no further discoveries taking place as all vulnerabilities are now within the disclosure subsystem. The corresponding stock residing at the end of the system is publicly disclosed. All vulnerabilities must flow to this end state, unless permanently residing in the discovered or platform disclosure stocks in an undisclosed state.

The publicly disclosed stock is the corollary of Undiscovered Vulnerabilities as all details about the vulnerabilities are public, with appropriate fixes created to address any issues. The publicly disclosed stock may receive vulnerabilities via the full or platform disclosure flows, mediated via appropriate disclosure rates and delays. Undiscovered and Publicly Disclosed stocks intersect at month 4 with 73 of 212 vulnerabilities depleted and publicly disclosed. At month 240 (end of simulated time) the quantity of disclosed vulnerabilities is 201, showing that almost all vulnerabilities have traversed the VDDS in this time, with 13 remaining within the system. Both Undiscovered and Public Disclosed stocks are closing stocks, at the beginning and end of the model. Whereas Platform Disclosure is a transitory stock, or buffer that provides a temporary accumulation of vulnerabilities. Platform Disclosure shows an increase in the quantity of vulnerabilities traveling via this flow, peaking at month 9 at 97 vulnerabilities and gradually reducing.

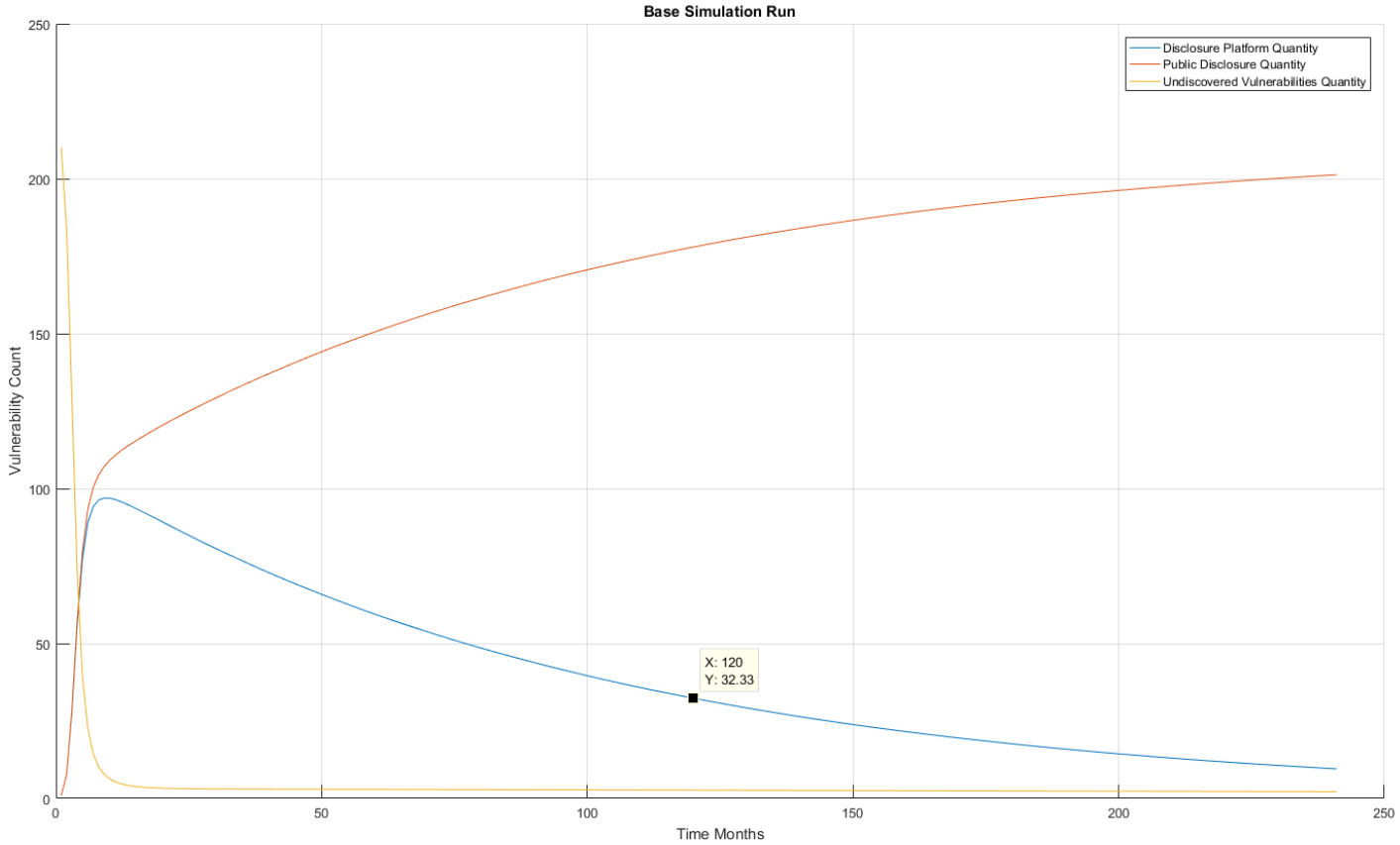


Figure 57 – Baseline Simulation Run Using Typical Parameters

7.2.1 Delays

Within the coordinated and full disclosure flow two delay points disrupt the movement of vulnerabilities from the discovered state to the publicly disclosed state. These are denoted as delays 2,3,4,5 and retard the flow of vulnerabilities via full and platform flows. Three rates are shown in Figure 58 below. Dissemination (coordinated) Disclosure Rate, Vulnerability Platform Rate and Full Disclosure Rate. The full disclosure flow is a simple movement of vulnerabilities from the discovered state to publicly disclosed state, mediated by a short time delay of <1 day. The smooth movement of vulnerabilities from state to state is due to no delay being in place as once a choice has been made by the discoverer then disclosure is almost immediate. The full disclosure rate is however dependent upon the introduction of vulnerability discoverers entering the VDDS, with a delay related to the rate of entry - the maximal rate of full disclosure occurs at month 2 with 15 vulnerabilities per month.

In contrast to full disclosure, coordinated and platform rates are ‘smoother’ in their behaviour. Vulnerabilities flow from the discovered stock entering first the coordinated vulnerability delay, mediated by the rate on entering the system for processing, denoted as platform entry delay. The maximum entry rate take place at month 2 with average 1.5 vulnerabilities taking place per month. Alongside the first delay is Platform Disclosure Rate following a smoother and elongated behaviour, with a maximum rate of 2.5 vulnerabilities per month at month 13. The largest delay of 30 days is encountered once a vulnerability has entered the Platform Stock, representing the average time that is taken for a vulnerability to traverse from discovered to publicly disclosed. In the base simulation this is based upon the average time drawn from chapter five.

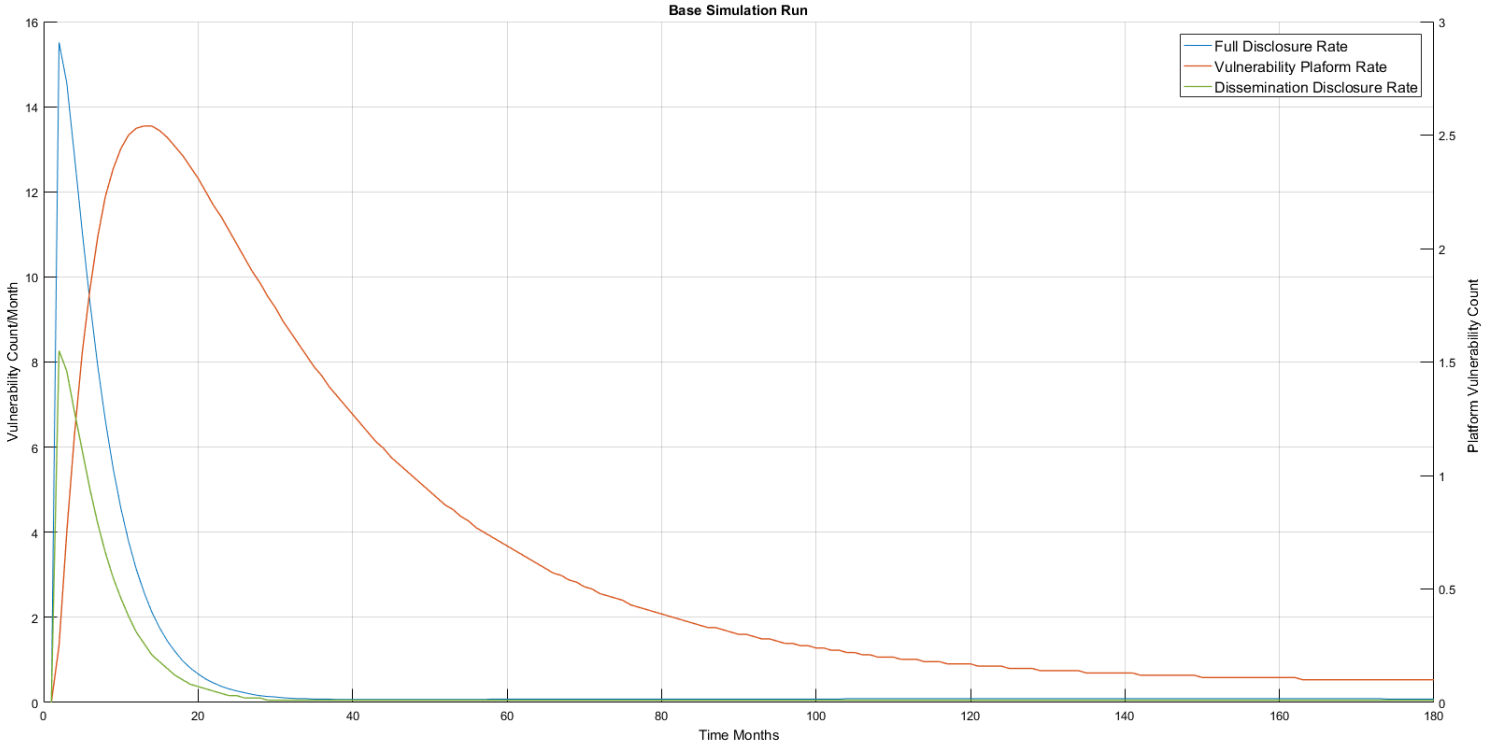


Figure 58 – Baseline Simulate Vulnerability Disclosure Rate Plot

7.3 Critical Variable 1: Perceived Available Rewards

The provision of monetary rewards for details of vulnerabilities is a relatively recent phenomenon. Nevertheless, the advent of the rewards system has generated an increase in the number of discoverers that have entered the VDDS as active discoverers and provided resources to devote more effort to finding vulnerabilities. Given this increase in rewards, or more specifically the *perception* of recompense that is on offer, it is logical to assume that the larger, or more readily available that the rewards are, the greater the number of vulnerabilities that will be discovered via the coordinated disclosure flow. Given this assumption, local numerical sensitivity analysis (also known as one-at-a-time or OAT) was performed upon the rewards variable to characterise the impact of variation. OAT sampling or variation is the technique whereby a variable changes between consecutive simulations (Saltelli et al., 2007, p.67).

7.3.1 Numerical Sensitivity Parameters

The `rewards` variable influence the quantity of researchers that are available to discover vulnerabilities. More specifically, the variable represents the probable rewards that are on offer, as there is no certainty that the Vulnerability Discoverer will receive the reward due to factors such as failure to discover or duplicate discovery. Therefore, the rewards variable is coded within the model as a perception factor, representing both the probability of discovery and the currency on offer. Sensitivity of the cash variable ranges between [$100 - $10000] with 11 simulations performed over 240 months (See Appendix A for raw data table).

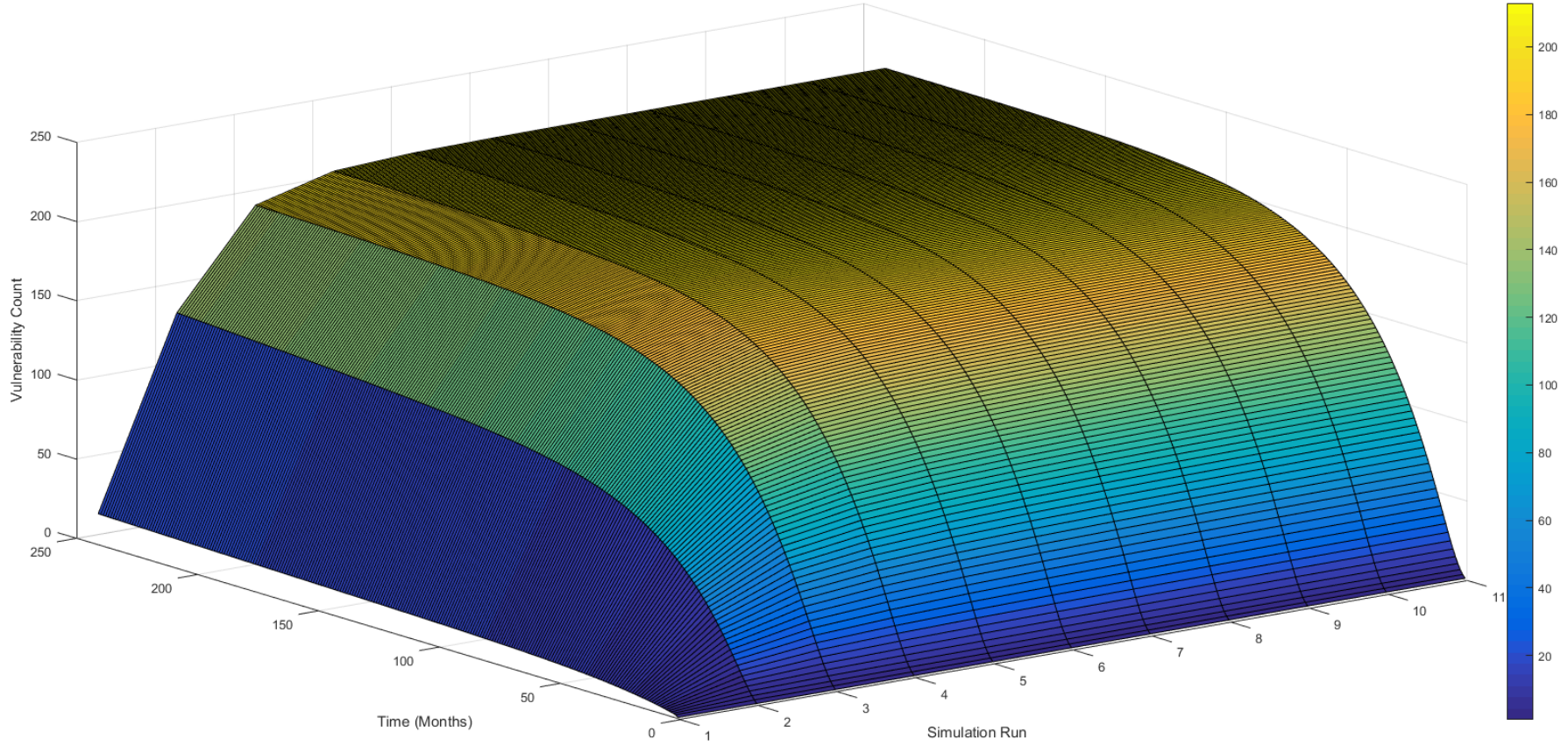


Figure 59 - 10 Simulations Runs for Rewards Variation Between $100 - $1,000 over 240 months

7.3.2 Model Behaviour Analysis

Vulnerability Disclosure and discovery is influenced by the monetary rewards. The initial simulation using the reward of $100 shows a slow increase of vulnerabilities disclosed over time of 240 months (bottom left in Figure 59). Starting at 0 public disclosure grows to a maximum of 20 disclosed vulnerabilities, which is consistent with a low reward offer. Increasing the reward to $200 shows a sharp increase of publicly disclosed vulnerabilities. At month 50, 95 vulnerabilities have been discovered, rising to 120 at month 100. The rate of public disclosure then plateaus toward month 150 and 200 with 128 and 133 vulnerabilities disclosed respectively. Finally, at month 240, 137 vulnerabilities were disclosed. Increasing the reward to $300 and $400 increases the vulnerabilities disclosure by an order of magnitude to 145 at month 50, in the case of $300 and 158 in the case of $400. With the range of rewards between $500 and $1000 insignificant improvements are gained with only +/- 10 vulnerabilities gained with double the rewards. Therefore, the scenario of diminishing returns, where more money does not mean more vulnerabilities, seems to take effect with the maximal number of vulnerabilities verses the rewards on offer around the $500-$600 level.

Simulation	Reward (US Dollars)	Month all Vulns Found
1	$100	Beyond End of Sim
2	$200	Beyond End of Sim
3	$300	Beyond End of Sim
4	$400	Beyond End of Sim
5	$500	213
6	$600	174
7	$700	169
8	$800	167
9	$900	166
10	$1000	165

Table 44 – Rewards simulation Summary Showing Results

7.4 Critical Variable 2: Software Quality Variation

Software quality within the field of information security has been a contentious issue for almost 20 years, with ideas on how to address deficiencies polarising debate on both sides (Anderson, 2001b; Anderson et al., 2013). These ideas range from using high quality software standards and languages to economic incentives. Given the apparent cognitive dissonance, there is acceptance that increasing the quality of produced software can reduce the number of vulnerabilities present within software, and therefore result in a reduction in risk. Given this, the quality of software that is modelled within the VDDS, is taken as a percentage and using Alhazmin and Malaiya (2005) vulnerability density we can simulate the impact of increasing and decreasing software quality.

7.4.1 Numerical Sensitivity Parameters

The quality of the software ranges from [10% - 100%] in incremental steps, expressing the ability to remove vulnerabilities from software prior to release. The quality of software is related to the number of vulnerabilities that are available for discovery within software, with the tacit assumption that the higher the quality the lower the quantity of vulnerabilities. Eleven simulations were performed over 240 months (See Appendix A for raw data table).

7.4.2 Model Behaviour Analysis

Initially at 10% quality, and very high vulnerability density (1.9×10^{-5} per line of code) we can see a maximum rate of discovery at month 4 of 69 vulnerabilities per month decreasing to 0 by month 22 when all vulnerabilities have been discovered. Conversely, examining software with 100% quality indicating vulnerability free code, we can see in total 3 vulnerabilities were disclosed within 10 years, all discovered within months 1,2 and 3.

Simulation Run Number	Quality	Max Rate	Month
1	10%	67	4
2	20%	61	4
3	30%	51	4
4	40%	39	4
5	50%	28	5
6	60%	18	5
7	70%	10	5
8	80%	4	4
9	90%	2	2
10	100%	0	0

Table 45 – Quality Simulation Showing 10% - 100% Variations

Clearly the production of vulnerability free software is unrealistic, and therefore simulations 8,9 and 10 should be viewed as an idealistic target. However, the simulation runs to point to a possible aspect for policy building.

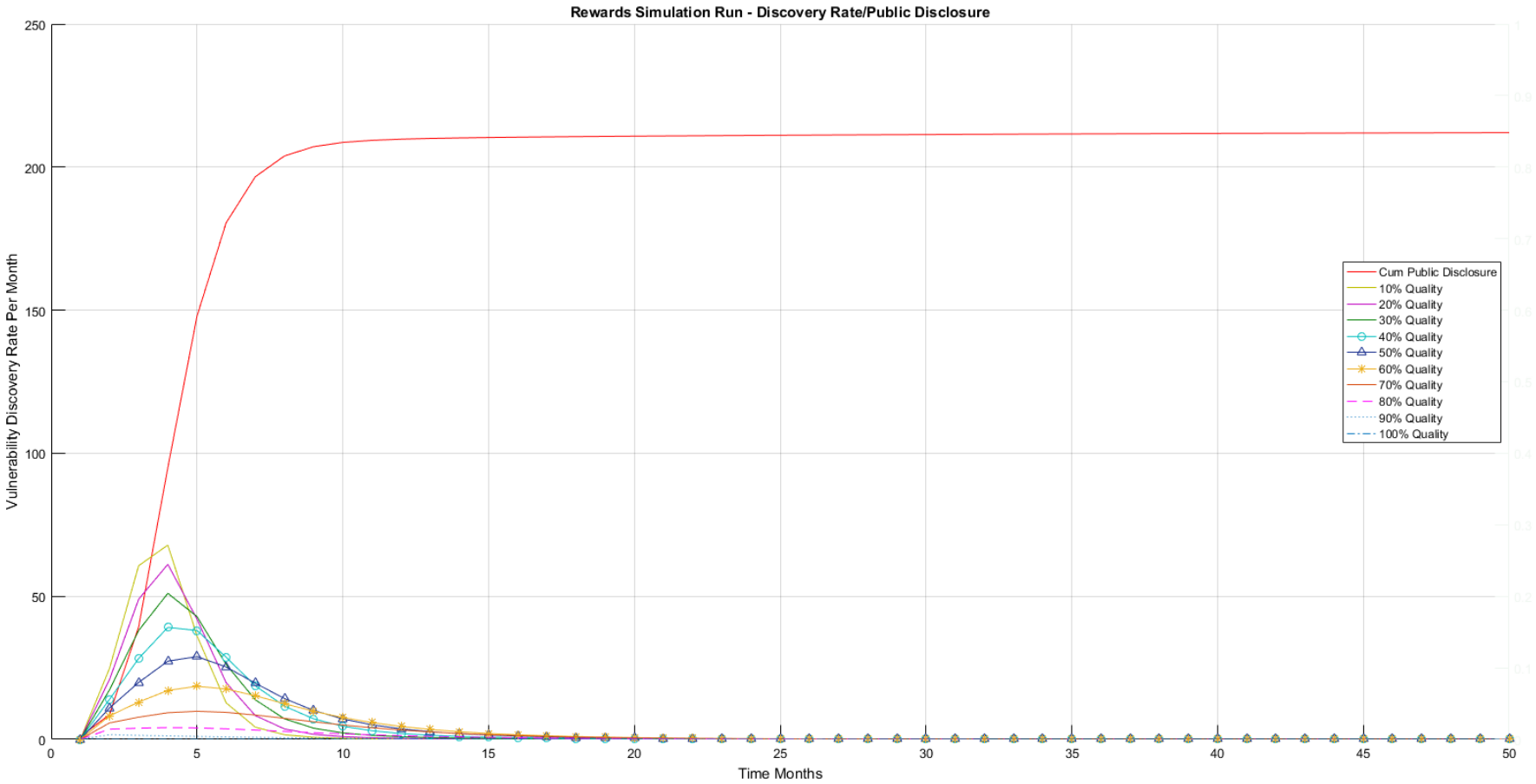


Figure 60 – Quality Variation Simulation with 10% Software Quality Public Disclosure Indicated

7.5 Critical Variable 3: Sentiment

Sentiment within the VDDS makes a significant impact upon both the number of active discoverers available to discover vulnerabilities, and the disclosure stance taken. As outlined in chapter 4, the level of positive or negative sentiment that is perceived toward a software originator materially impacts which disclosure stance is taken by the Vulnerability Discoverer. The assumption here is that if sentiment is high, and that software originators are seen in a positive light, then discoverers will act more favourably towards the originator and disclosure vulnerabilities directly, or via a disclosure platform. Therefore, the opposite must also occur, if sentiment is negative then discoverers will tend to adopt a full disclosure stance.

7.5.1 Numerical Sensitivity Parameters

Sentiment governs two aspects of the VDDS model, conversion rate and full disclosure rate. Specifically, the perception of sentiment toward the software originator, which is encoded within the VDDS model in two places, discoverers subsystem and disclosure subsystem. Sensitivity of the Sentiment variable ranges between [0.1 - 1] with 0.5 representing neutral, and 1 representing overwhelming positive sentiment. Eleven simulations were performed over 240 months (See Appendix A for raw data table), however for visualisation purposes graphs are terminated at 40 months.

7.5.2 Model Behaviour Analysis

At 10% (low) sentiment the flow is evenly spread between full disclosure and vulnerabilities disclosed via the disclosure platform, with a peak of 35 via full disclosure (month 3), and 32 per month for platform disclosure (month 4). This split suggests that, despite a low level of sentiment vulnerabilities are still disclosed via the ethical, platform flow, however, a slight tendency to disclose via the full disclosure route still exist.

With 50%, or neutral sentiment, vulnerability disclosure is skewed toward the platform flow, with a peak of 33 vulnerabilities per month in month 5, and previous month value of 31. On the other hand, the full disclosure flow has a

significant decrease in the number of vulnerabilities, with a peak of 18 in month 3. Finally, with 100% sentiment (high), the max values via the platform flow were 30, 33 and 30 in months 5,6 and 7 respectively, marking a dramatic reversal of flow with no vulnerabilities disclosed via the full route.

Simulation	Max Vuln Per Month	
	Full	Plat
Sentiment 10%	35	32
Sentiment 50%	18	33
Sentiment 100%	0	33

Table 46 – Sentiment Simulation Variation Showing 10% and 100% Values.

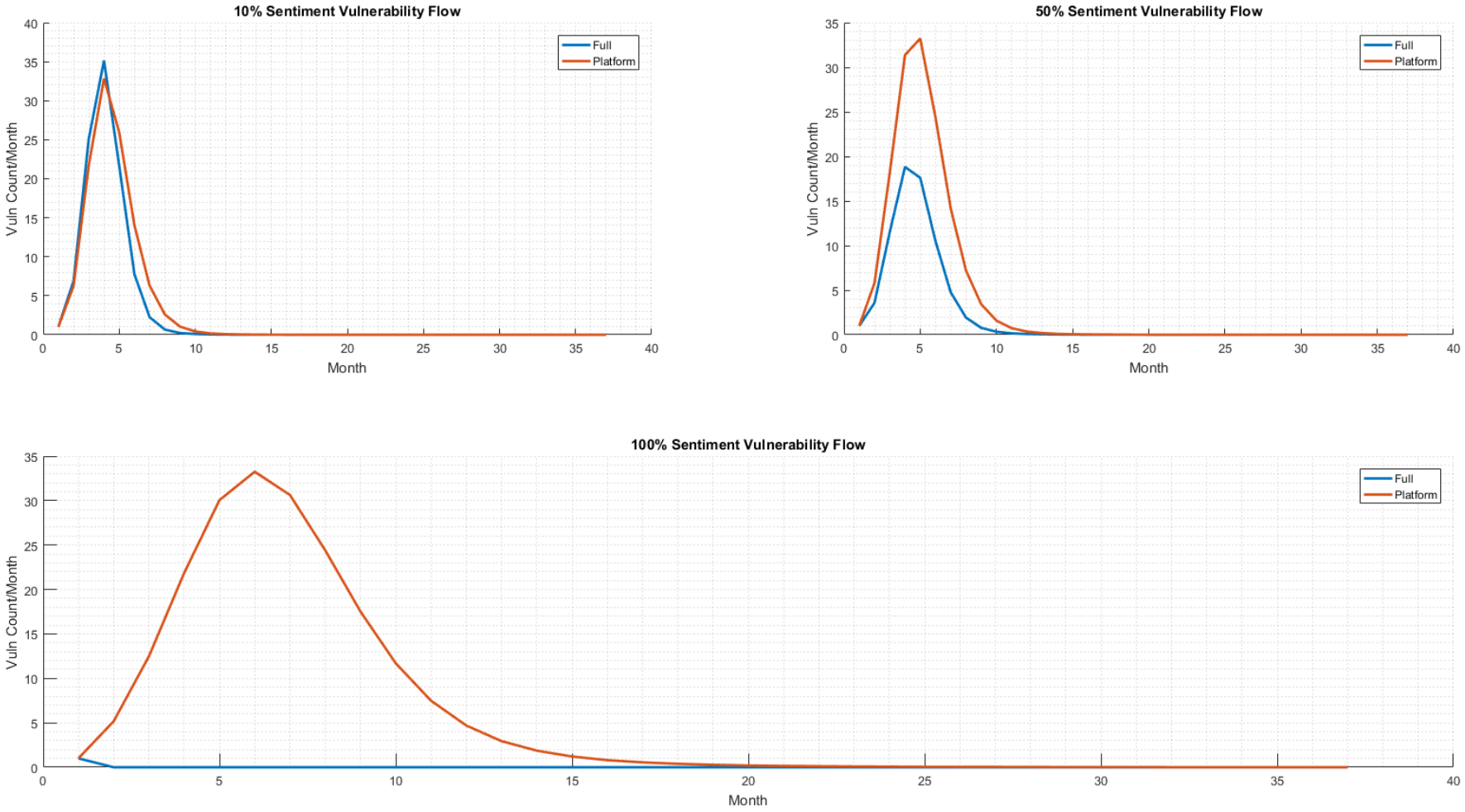


Figure 61 – Sentiment Impact upon Full and Platform Disclosure Flows

7.6 Critical Variables 4: Disclosure Delay

Time within the VDDS makes a direct impact upon all variables within the VDDS, specifically the time taken to disclose a vulnerability publicly via both full and coordinated routes. As outlined in chapter 5, the level of delay that a discoverer may encounter ranges significantly. Moreover, the delay that is encountered via the platform route is assumed to impact both the sentiment and subsequent disclosure choice, however no data is available to substantiate this. It is, however, possible to increase and decrease the time upon which a vulnerability takes to transit the platform flow. It is worth noting that there are assumed to be no delays associated with full disclosure as details are released immediately.

7.6.1 Numerical Sensitivity Parameters

Delays govern many aspects of the VDDS, such as acquiring discovery skills, and increasing software quality and sentiment toward software originators. However, the most visible and impactful delays are via the disclosure system, particularly the quantity of vulnerabilities via the platform flow. Specific industry policy suggests 30, 45 or 90 days to allow software originators to deal with triage and creation of fixes to the issue (CERT, 2017; ISO/IEC, 2014). Therefore, delays that are analogous to industry policy are tested. Sensitivity of the platform delay variable ranges between [1 - 90] days. Eleven simulations performed over 240 months (See Appendix A for raw data table), however for visualisation purposed graphs are terminated at 40 months.

7.6.2 Model Behaviour Analysis

Vulnerabilities flow from the discovered state, which is the decision point whereby discoverers choose to disclose in a full, platform or direct manner. Once a disclosure stance has been adopted, vulnerabilities are mediated via the rate equations outlined in the previous section. Inspecting Figure 62 we can see that small simulated delays within the platform flow generate large transit rates, with a peak of 72.1 (indicated by the blue line) being accumulated over 50 months within the platform and standard deviation of 14.7 resulting in a more

peaked distribution (Kurtosis value of 12.5). Conversely, a delay time of 100 days results in a peak value of 97 accumulated in the stock (Kurtosis value 2.2), over 240 Months. This increased delay slows the dispersal rate into publicly disclosure and resulting in the stock not being fully drained by the end of the simulation.

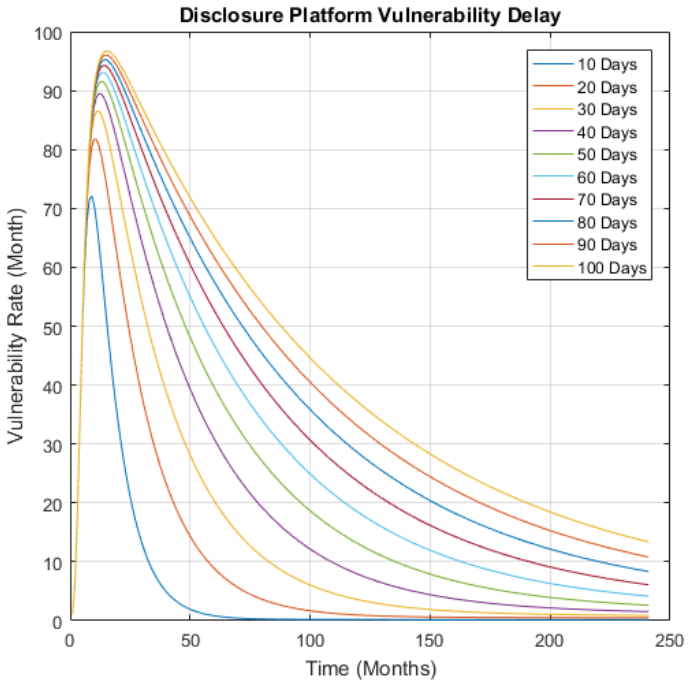


Figure 62 – Disclosure Flow for Platform with Incremental Increase in Delay

Figure 63 below shows the increase in the spread of accumulated vulnerabilities with mean, and standard deviation values increasing with increasing delays. Figure 63 shows a series of boxplots indicating the progression of mean (red line), standard deviation (blue box) and outliers (red crosses).

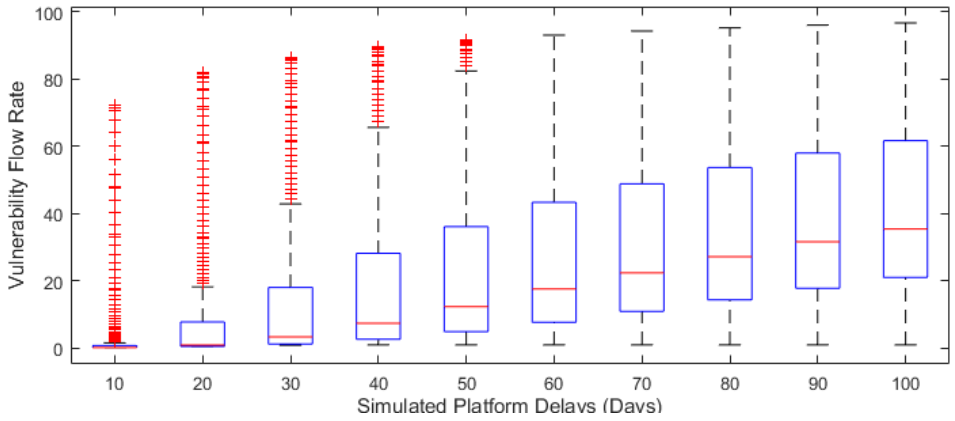


Figure 63 – Boxplot Showing Mean, Standard Deviation and Spread of Vulnerability Accumulation

7.7 Scenario Testing

Performing both baseline simulations with observed values, and simulated OAT sensitivity analysis enables a good foundational overview to be established. However, varying more than one variable to simulate policy impact is required. Sterman (2000, p.885) advocates simulating both worst and best case scenarios outcomes of policies that require testing. Hence, three policy scenarios were chosen that are analogous to worst, best and current scenarios. Scenario A outlines the worst possible state of the VDDS system where low-quality software is produced with a high quantity of vulnerabilities present; vulnerability discoverers are not incentivised to disclose vulnerabilities and sentiment toward software originators is very negative. Scenario B is considered the moderate state whereby sentiment towards software originators is neutral, rewards are high to incentivise vulnerability discoverers and quality of software is low. Finally, Scenario C is the best case, where software quality is high, essentially vulnerability free, incentives are also high and the sentiment toward software originators is high. Each scenario represents the policy choices that *could* be implemented within real world organisations given the appropriate amount of time, resource and political impetus. Each scenario broadly attempts to explore the possible inferred future outlined in chapter five, section 5.3.1 with exponential, steady and Flat futures relating to scenarios A, B and C respectively.

	Future	Sentiment	Money	Quality
Policy Scenario A	Exponential	0.1 (10%)	$100	0.1 (10%)
Policy Scenario B	Steady	0.5 (50%)	$1000	0.1 (10%)
Policy Scenario C	Flat	1 (100%	$1000	0.9 (90%)

Table 47 – Policy Scenario Runtime Parameters

7.7.1 Policy Scenario A: Low Quality Software, Low Rewards, Low Sentiment (Worst)

In Figure 64 below, low quality software, low rewards and low sentiment all contribute to a mixed outcome. The final number of vulnerabilities that were publicly disclosed is 22 out of 212 (10.3%). Whilst this may seem a desirable outcome as the risk that end users are subjected to is low, due to the number of vulnerabilities, vulnerabilities potentially discovered by malicious actors outside of the legitimate VDDS. This scenario is also exacerbated by the increase in full disclosures peaking a month earlier in month 4. Incentives to draw discoverers to a coordinated disclosure stance via the platform is scant, with only $100 used to incentivise discoverers. Finally, the sentiment toward software originators is low, causing the full disclosure rates to be slightly higher, peaking at 3.85 in month 4. The publicly disclosed vulnerabilities follow a typical logistic (S-shaped) growth curve, however with the limited number of vulnerabilities found, it will never reach the asymptotic limit of 212 vulnerabilities. This is due to limited incentives on offer and the low sentiment toward software originators.

This scenario closely resembles the period when the VDDS existed in a pre-disclosure platform and self-organised software originator bounty schemes (pre-2010) predominated. The issues that have stemmed from the uncontrolled increase of software vulnerabilities have been well documented (Anderson et al., 2013; Bilge and Dumitras, 2012; Frei, 2009; NCA, 2017). Given the unknown quantity of vulnerabilities being potentially discovered outside of the legitimate VDDS, this scenario is an undesirable set of policy choices.

	Standard Deviation	Mean	Max
Full Disclosure	0.94	0.34	3.85
Discovery	2.32	0.65	12.49
Disclosure Platform	0.93	0.39	3.7

Table 48 – Summary Statistics for Scenario A

Within this scenario there are three variables that represent poor policy decisions ultimately resulting in a high number of vulnerabilities that are unknown, and therefore uncontrolled. As such the potential risk that organisations face is very high. This uncontrolled increase of vulnerabilities is broadly in line with the exponential inferred future reference mode. The exponential increase of vulnerabilities, which in turn moves discoverers toward the black-market due to insufficient motivation or rewards, coupled with poor software quality are the defining characteristic of epoch one (see section 5.3.2). If the low quality, rewards and sentiment factors do not increase then this inferred future is probable. The assumption here is that a) new vulnerabilities are introduced to new software and software practices are not significantly improved from today's levels; b) the number of participants within the system increases significantly; c) opportunities to sell vulnerabilities increases significantly; and d) the desire for vulnerabilities increases.

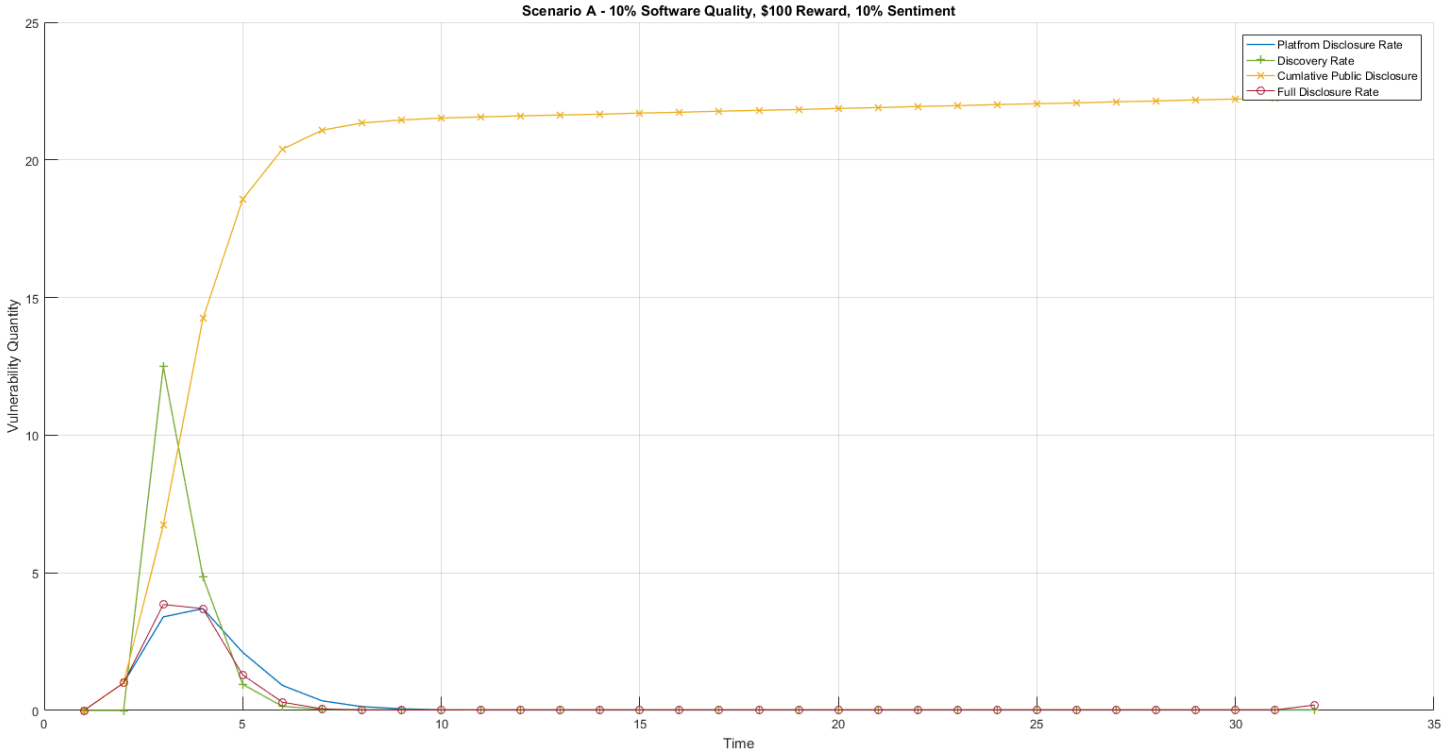


Figure 64 – Scenario A: Cumulative Publicly Disclosed Vulnerabilities, Discovery Rate and Platform Disclosure.

7.7.2 Policy Scenario B: Low Quality Software, High Rewards, Neutral Sentiment (Current)

Figure 65 below shows low quality, high rewards and neutral sentiment, contributing to a mixed outcome. The final number of vulnerabilities that were publicly disclosed is 212, completely exhausting the maximum undiscovered amount within the software. In month 10 almost all (202) vulnerabilities have been publicly disclosed. Whilst this may seem an undesirable outcome at first analysis, the risk that vulnerabilities are discovered outside the legitimate VDDS is reduced to zero. As all vulnerabilities are publicly disclosed they must have traversed via the full or platform disclosure flows, hence the removal or malicious discovery. The scenario is influenced by the high rewards on offer ($1000), incentivising discoverers to uncover issues within software, and disclose via the platform flow. The maximum rate of disclosure via this route is 30 in month 6, falling to zero in month 27. Full disclosure occurrences also exist, with the maximum rate of full disclosure peaking at 16 in month 6, reducing to 0 in month 24.

Scenario B closely resembles todays (2017) current set of policy decisions that have been made. Software quality consistently remains low, providing both the opportunity for vulnerability discovery and the means to do so. Additionally, importance of discovered vulnerabilities in reducing the risk profile of organisations is reflected in the high rewards on offer, incentivising discoverers to choose a coordinated disclosure flow. Finally, sentiment has increased, representing a higher probability of coordinated discovery. Whilst an improvement on Scenario A, the activities surrounding the disclosure process (contracting platforms etc.), economic outlay of money, and incentives and post incident clean-up are potentially larger than increasing the quality of the software prior to release, thus eradicating vulnerabilities at source.

	Standard Deviation	Mean	Max
Full Disclosure Rate	4.9	2.73	16.71
Discovery Rate	15.9	8.36	56.1
Disclosure Platform Rate	9.22	5.44	30.85

Table 49 – summary Statistics for Scenario B

With average sentiment, quality and high rewards the quantity of vulnerabilities that are discovered within the legitimate VDDS continues along the steady inferred future. As vulnerabilities are discovered within the VDDS, as opposed to the black market, or not disclosed at all the rate of discovery continues in a controlled manner, and steady rate. This inferred future is the most probable given the recent advancement in markets and awareness of cyber security within organisations. This scenario assumes that a) new vulnerabilities are discovered in existing software, but improved software practices reduce the number of vulnerabilities with new software; b) the vulnerability discovery system continues to add new vulnerability opportunities to sell or receive bounties; and c) The number of participants increases.

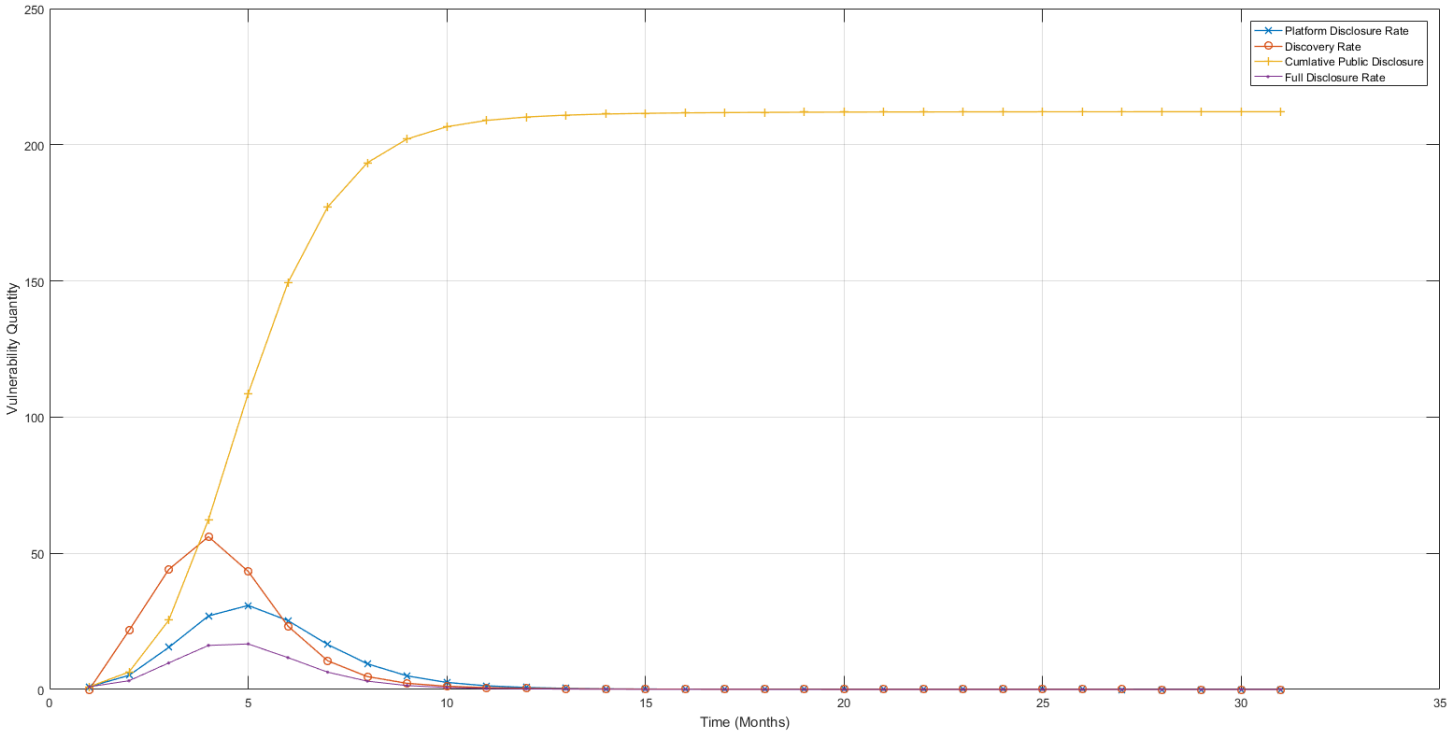


Figure 65 - Scenario B Cumulative Publicly Disclosed Vulnerabilities, Discovery Rate and Platform Disclosure Values.

7.7.3 Scenario C: High Quality Software, High Rewards, High Sentiment (Best)

Figure 66 below provides an overview of the best-case scenario with high rewards, high positive sentiment and high-quality software. This is reflected in the final number of publicly disclosed vulnerabilities which at the end of the simulation results in a maximum of 12 discoveries. This low number is due to the high quality of the software, and resultant low number of discoverable vulnerabilities. As the motivation and sentiment is high to disclose any identified vulnerabilities, all flow via the platform route, with none flowing via the full disclosure route.

With all vulnerability factors at the most beneficial to organisations, the flat inferred future is the least improbable as it assumes that high motivation, high sentiment and high-quality software will be the norm. These factors result in a flat increase in the number of vulnerabilities disclosed, thus a modest increase in risk is apparent. This future exists where the number of recorded vulnerability disclosures and therefore the number of discovered vulnerabilities that are found decreases to zero. This assumes several things: a) the perfect debugging of all new vulnerabilities upstream within the development process and removal of vulnerabilities within software; b) once a vulnerability is removed from the software ecosystem, it is not reintroduced and that no new vulnerabilities are introduced via new software versions.

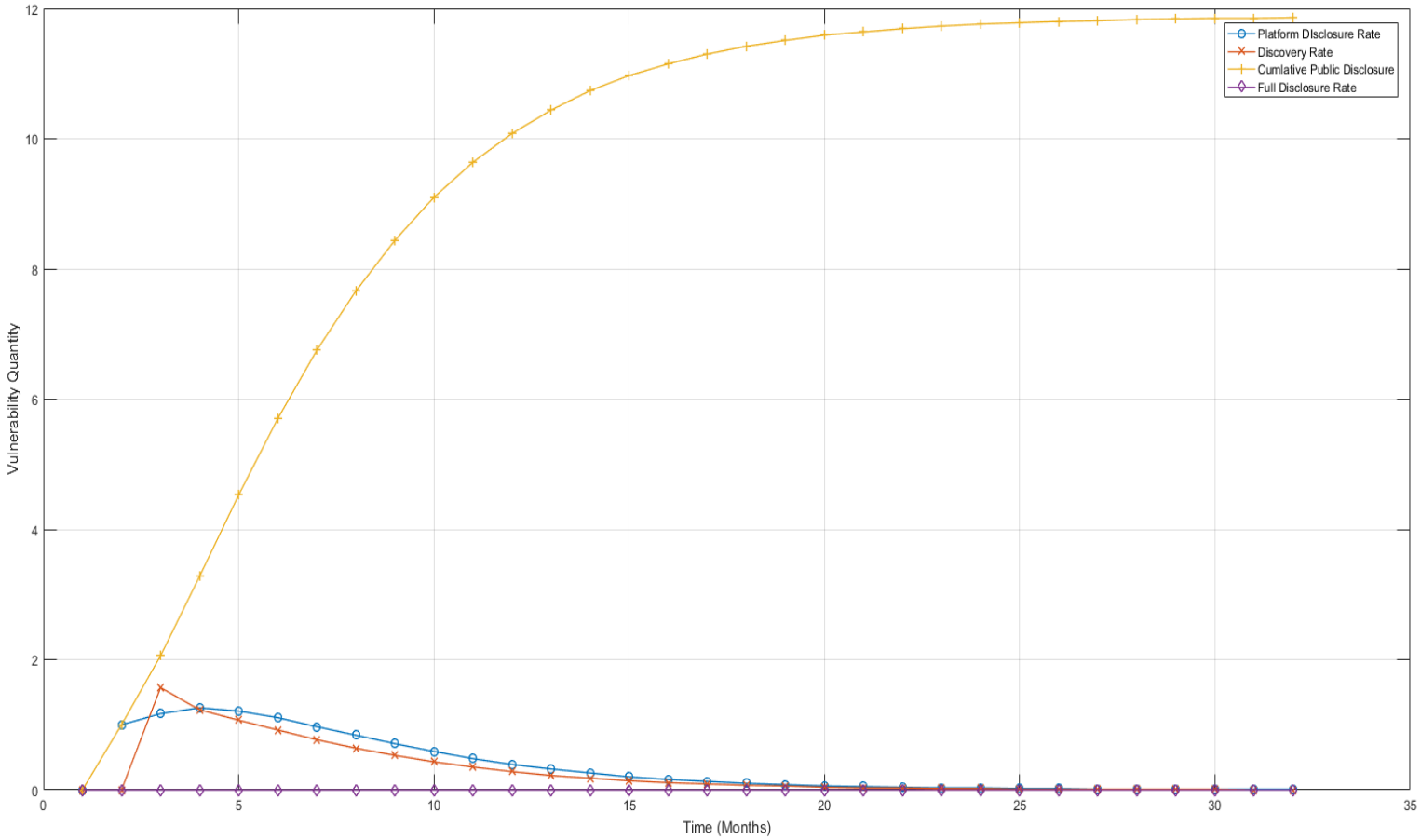


Figure 66 - Scenario C Cumulative Publicly Disclosed Vulnerabilities, Discovery Rate and Platform Disclosure Value

7.8 Analysis of Policy Implications

Choosing the right policy for the right reasons is difficult, with vulnerability discovery and disclosure being no exception. Vulnerabilities are by their very nature potentially harmful to the stability of the software that they affect. Therefore, a key question that must be considered when choosing a set of polices is which causes the least harm, most efficiently and with least disruption? If we take the introduction of rewards for vulnerability details, we can see that it will entice both existing and new discoverers to enter the VDDS, thus increasing the number of vulnerabilities within the system.

However, if the correct procedures, with minimal delays and good communication are not followed then there is a distinct possibility that a proportion of the vulnerabilities will be disclosed in an uncontrolled way, increasing the potential harm to organisations. Similarly, if the rewards that are on offer from brokers increase significantly so that software vendors are no longer able to access the details of the issue, the risk of stockpiling and unauthorised or accidental disclosure may occur (Goodin, 2017a).

Turning to sentiment, the increase of sentiment toward software originators is the most influential, and importantly cost effective. By increasing the sentiment toward the originators. Discoverers on the whole disclose more vulnerabilities, and do so without recompense via money. Finally, increasing the quality of any software that is produced by originators is typically a good option, unless resources and skills are at a premium and cause organisational issues, such as bankruptcy.

The first recommendation is centred on the time that is taken to traverse the disclosure process. This has been measured to be on average 77 days, with extreme examples of 619 and 1612 days encountered. Whilst producing a fix to potentially complex software is not considered a trivial undertaking, taking around four and a half years from initial contact is inexcusable. This ultimately results in both anger from the point of the discoverer, and increased risk as the default position is the resumption of a full disclosure stance. **Therefore,**

decreasing the length of time taken to disclose vulnerability details is the first recommendation.

Secondly, and related to the length of time taken to disclose vulnerabilities, is communication with discoverers. There are numerous examples of communication delays, communication failures and miscommunication between discoverers and software originators. Given the nature of the VDDS, communication normally occurs via email, normally with initial contact via a general email address such as product-security@apple.com (Apple, 2017). As such, dialogue, unless carefully managed, could fail, resulting again in a resumption of default full disclosure practices. Alongside this, the number of communication steps that are taken has been measured to be an average of 4, yet a maximum number of 39 has been recorded. **Consequentially, the second recommendation is that software originators adopt a more proactive and managed communication regime.**

A third proposed policy is the adoption of a pan-industry Memorandum Of Understanding (MOU) concerning the discovery and disclosure of vulnerabilities. This would take a similar form of the MOU between the Crown Prosecution Service (CPS) and the Association of Chief Police Officers (ACPO) concerning Section 46 Sexual Offences Act 2003 (Service, 2017). The agreement sets out the protection principles for professionals from prosecution and arrest from both the police and CPS with regards the use of the electronic communication network when indecent images of children have been transmitted. A similar agreement could be adopted here, thus removing the threat of legal action if reasonable efforts have been taken to disclose details in a coordinated and ethical way. **Hence recommendation three is to adopt an industry wide memorandum to set out principles of *Nolle Prosequi* with respect to vulnerability disclosure.**

Controversially, the fourth policy aspect is directed toward the vulnerability discoverer. As stated, the default position for failures on the part of the software originator is to resort to full disclosure. This results in unconstrained risk, and potential harm toward end users, for example the recent shadow broker leak of

0day vulnerabilities is a form of full disclosure (Goodin, 2017a). **As such the fourth recommendation is to remove the default position of last resort by software vulnerability discoverers and replace it with a more inclusive policy.**

Finally, and most impactful is the adoption of secure software practices. It has been shown that by adopting secure software practices, a significant proportion of vulnerabilities within software can be removed prior to public release. As such the **fifth and final recommendation is that software originators adopt secure coding practices within software engineering teams.**

7.9 Chapter Summary

This chapter has described four key variables within the VDDS, and the impact and interactions they have on the VDDS behaviour. Alongside this, delays in the processing of vulnerabilities via disclosure platforms or directly via full disclosure routes were examined. Furthermore, simulations outlining the behaviour of the VDDS under differing numerical values, known as sensitivity analysis was presented.

Establishing a baseline of how the model behaves, with normal, or average parameters is of importance. As such values were selected from the exploratory data analysis thus providing an observed set of initial conditions upon which to explore the VDDS model. By modifying the variables, simulations were performed to characterise the impact upon simulated behaviours. The advent of the rewards system has generated an increase in the number of discoverers that have entered the VDDS. Given these increases, it is logical to assume that the larger, or more readily available that the rewards are, the greater the number of vulnerabilities that will be discovered via the coordinated disclosure flow. Alongside the simulated increases in rewards software quality was also simulated and considered with significant impact upon the number of discovered vulnerabilities, thus suggesting a possible policy choice. Finally, sentiment within the VDDS was simulated, also illustrating a significant impact upon both the number of active discoverers available to discover vulnerabilities, and the disclosure route taken. Time within the VDDS makes a direct impact upon all variables within the VDDS, specifically the time taken to disclose a vulnerability publicly via both full and coordinated routes. Moreover, the delay that is encountered via the platform route is assumed to impact both the sentiment and subsequent disclosure choice.

Three Scenarios were presented illustrating the different behaviours that are exhibited when key state variables are changed. All Scenarios can be aligned into 3 specific epochs when characterising the evolution of the system. The initial epoch can be described as uncontrolled and constrained growth of vulnerabilities, with the time span running from the mid 1990’s through to

approximately 2009-10. Epoch 2 is suggested to commence around 2010 with the advancement of vulnerability disclosure platforms, and a change in perception towards vulnerability discoverers from software originators and rewards being offered. Finally, epoch 3 is suggested to behave like one of the inferred futures.

In summary, by adopting a set of policies based upon simulated Scenarios several vulnerability reduction or vulnerability control mechanisms can be brought to bear. With a modest reduction in time, increase in rewards and resultant increase in sentiment a flow of controlled and coordinated vulnerability discovery is possible. This is outlined in Scenario B in figure 65.

8 Conclusion and Discussion

The aim of this research is to advance the underlying theory of software vulnerability, ground it in real empirical data and provide a systemic model to increase the understanding of policy choice upon the VDDS. To do this an investigation of the factors that underlie the software Vulnerability Discovery and Disclosure System was performed. This was achieved by understanding and characterising the Vulnerability Discovery and Disclosure System by providing an in-depth and systematic analysis, a mathematically-based model of how the system operates and an impact analysis of 'what if' policy simulations to guide risk reduction interventions.

This research used a mixed methods approach, coupled with additional modelling and simulation phases. What has emerged is a complex, yet quantifiable system that is heavily influenced by the entities that inhabit it, and policy choices made by those entities. Rooted in direct observations of the VDDS, and how it has evolved, this study found that there are five key themes that pervade the VDDS, shaping how vulnerabilities are discovered and disclosed once uncovered.

8.1 Discussion

Based on the phenomenological philosophy, this research considers the VDDS to be constructed and subjective. The phenomena under investigation is the growth, of software vulnerabilities and is underpinned by both qualitative and quantitative methods in an inductive framework. The initial stage of the research was to identify key text, data sources and narratives to build a thematic model of the VDDS, based on recorded experiences. This Thematic Analysis method was used to code, and categorise a wide range of data items, collected from online sources. Five key themes emerged that pervade the VDDS, namely Perception of Punishment, Software Originator Interactions, Ethics and Disclosure Stance, Motivation for Discovery and Emergence of New Vulnerability Markets. Throughout the data the repeated concept of punishment and fear of punishment emerged when actions of disclosure, specifically

coordinated disclosure were undertaken. Furthermore, repeating words and phrases such as “nasty”, “threat”, “threatening” suggest an adversarial punitive relationship between both discloser and the recipient of the vulnerability information. What emerges from the data is the notion that researchers or hackers are not taken seriously by software vendors, or when they are listened to then the normal course of action is generally punitive in nature.

Furthermore, there seems to be a distinct division between actors within the system, specifically an adversarial relationship between researchers and software originators. An overarching feeling of persecution and lack of control seems to be present from within the researcher group. In addition, there is a theme that researchers who disclose vulnerabilities to software vendors do, or more importantly *did* so for the right reasons, but as the level of perceived persecution has increased this ethical stance has been eroded over time. Disclosure of vulnerabilities is closely linked to the interactions between the discoverer and software originator. A key finding is that the communication between originators and vulnerability discoverers has been significantly lacking, and in most cases, continues to be. Disclosure of vulnerabilities within the VDDS has been categorised into two main stances, full and responsible with polarising effects upon the participants.

Dependent upon which disclosure stance is taken the time frames are dramatically different ranging from immediate to several months, but crucially if the communication is good, and previous experiences are positive, these deadlines imposed by the discovers are quite flexible (Sintonen, 2017). As with most psychosocial systems the emotional state of a vulnerability researcher and the social constructs that surround them is a critical factor in establishing the course of action the researcher may take. For example, when researchers consider which disclosure stance to take - either full or coordinated - this choice is directly influenced by either personal experience or, more commonly, via reflecting on community experiences published within social media when soliciting advice.

In general, careful consideration is given when researchers choose to divulge the vulnerability they have uncovered and typically try to do the right thing. The level of sophistication that is apparent here, is startling, as the software vulnerability discoverer has been characterised as reckless, with a disposition to favour anarchy than harbour a sense of social responsibility. Within the motivation theme, the quality of software is an important consideration which is often discussed by all entities within the discovery system. There is sentiment from two specific groups of entities, users and discoverers, that more could and should be done around removing software vulnerabilities in the software development process, rather than post release. Within the VDDS the advent of new market forces, coupled with the overwhelming need to adopt an ethical stance when disclosing a vulnerability has driven new innovative market instruments. These factors have shaped and influenced the creation of a so called 'grey market' environment, and provided an alternative to underground black-markets. Finally, the value of a vulnerability is a significant factor that is discussed at length within the VDDS. The ability to sell a vulnerability in a legitimate manner is a recent phenomenon, with the first auction site known as WabiSabiLabi being launched in 2007 (Bradbury, 2007).

The VDDS is a socio-technical system that is paradoxically both global and individual at the same time. This is due to the fact that a single vulnerability discoverer can affect change on a massive global scale (Goodin, 2017a). With this level of duality, it is not surprising that the VDDS seems to be chaotic, however there are structures and mechanisms that regulate how the system behaves. This internal regulation takes the form of narratives, ethical frameworks and institutional memories embedded within the VDDS shaping the behaviours of VDDS entities. An example of this is outlined in section 4.7.5 where vulnerability discoverers regulate the disclosure of vulnerabilities, specifically when making a choice to disclose without software originator coordination. Alongside this the concept of time weighs heavily and influences the VDDS in a significant way. Most significantly is the time impact of the discovery phase, whereby skills and technical details of the software are being established, and the disclosure of any detected vulnerabilities.

Delays encountered in the disclosure phase are the primary reason why full disclosure stances are taken, thus increasing the potential for exploitation. Within the VDDS the concept of elapsed time is a significant influencing factor on all that is undertaken - from the amount of time that vulnerability discovery takes, through to the delays that exist when communicating with software originators to disclose a vulnerability. Consequentially, the time that elapses between steps that make up a complete interaction between entities forms a fundamental element of the discourse within the VDDS. These two delays make up a significant proportion of all VDDS delay.

8.2 Research Contributions

Alhazmi and Malaiya (2005) initiated the field of vulnerability discovery modelling, with Anderson (2001b), Frei (2010) and Radianti (2006) investigating the economics and black markets for vulnerabilities. However, no evidence has been found of researchers who have investigated the systems aspect of the legitimate VDDS, quantified the structure or simulated risk reduction strategies presented here. This research contributes to the literature on software vulnerability discovery and disclosure, not from a technical view, but from a systems and policy perspective. Three key contribution have been made.

Firstly, the creation of theory as to how the VDDS is structured, behaves and, importantly, the relationships between entities. The collected data used to build this model can be reused to assist future researchers in vulnerability discovery and disclosure investigations. Secondly, the creation of a System Dynamics model allows for the simulation of policy choices, and the potential avoidance of policy that could cause harm or unintended consequences. Finally, and crucially, the aim of this research – what is driving the growth of software vulnerabilities.

The implications of this new knowledge, and framework could assist within risk management processes, and within professional practice by systematising the process of vulnerability risk analysis. It is now possible to test scenarios prior to potentially costly, or disruptive policy being applied. Additionally, future studies may build upon the raw collected data corpus, extend the thematic model and

augment the system dynamic model. Indeed, there has always been an argument to increase software quality within COTS, this research goes some way to reinforcing that argument. The novel model structures that were codified to construct the VDDS show a significant number of balancing and reinforcing loops, all of which change in dominance over the lifespan of the VDDS. These new structures help to refine the current mental models of alternative policies organisations may make. For example, the component of time is pervasive as it influences almost every choice or policy decision that is made by discoverer and software originator alike.

Three policy scenarios were simulated, representing three policy decisions that are broadly analogous to epochs 1, 2 and 3 (see section 5.3.2). Scenario 1 showed the movement of the vulnerability problem, from the legitimate VDDS to the illegitimate vulnerability market or via cyber criminals due to the low quality of software and lack of incentives. This scenario shows the unintended consequences of inaction due to low software quality, or more importantly a lack of activity that is considered to be harmful. Finally, an abstraction of the VDDS was created using the System Dynamics approach to simulate the impact of differing policy scenarios, mimicking current and future cyber security futures for the VDDS and wider cyberspace, validated with existing VDDS data. The policy simulations indicated that three approaches can be implemented to comprehensively reduce vulnerability centred risk: a) improved software quality; b) increased rewards; and c) better relationships between software originators and discoverers.

8.3 Limitations of this Research

As with all phenomenological research, the observations of an issue, as opposed to taking part and living the research, can abstract away detail and consequentially the fidelity of the resultant analysis may not be as detailed as wished. However, this potential lack of fidelity has been reduced by triangulating analysis and collection of multiple data sources.

Furthermore, the scope of the model boundary was chosen to include the major entities that are assumed to impact the VDDS. Therefore, the potential for

exogenous entities to influence the modelled system exists, for example criminal or national governments. Also, the implementation of any recommendations must be tempered by the reality of economic policy and political will of national and international governments. Finally, the possible scenarios that have been explored within the scope of this research were limited to three. This is obviously, a smaller subset of the potential set than is possible. The number of potential scenarios is only limited by the different parameters that can be simulated. Therefore, if only Quality, Sentiment and Rewards changed in increments of 0.1 for each, the total number of combinations is 1000 different combinations (10^3).

Finally, the mixed methods approach, whilst flexible, is very time consuming and cognitively expensive as the investigator is required to immerse themselves within the subject matter. Alongside this, to use mixed methods there is the requirement of being competent in both quantitative and qualitative research techniques. As such investigators, may need to learn multiple methods and techniques, and understand how to mix and track between them. Upon reflection other research frameworks maybe appropriate, not from a subject perspective, but from a time and cognitive load perspective.

8.4 Policy implication

The results presented show that under certain circumstances, if software originators and vulnerability discoverers operate in concert favourable behaviours can occur. However, these favourable outcomes can be disrupted if feedback loops occur, as is the case of Scenario A, where vulnerability discovery is low, yet vulnerabilities may be discovered outside the legitimate VDDS. This means that a policy that solely looks inwards, and does not take into consideration the human aspects of the VDDS is potentially doomed to failure. Failure to focus on critical structures and behaviours, could repeat the initial epoch of unprecedented vulnerability growth. A number of simple policy recommendations, can impact the VDDS in a positive manner, and in a relatively inexpensive way.

The first recommendation is centred on the time that is taken to traverse the disclosure process. **Therefore, decreasing the length of time taken to disclose vulnerability details is the first recommendation.**

Second, and related to the length of time taken to disclose vulnerabilities is communication with discoverers. **Consequentially, the second recommendation is that software originators adopt a more proactive and managed communication regime.**

A third proposed policy is the adoption of a pan industry memorandum of understanding concerning the discovery and disclosure of vulnerabilities. **Hence recommendation three is to adopt an industry wide memorandum to set out principles of *Nolle Prosequi* with respect to vulnerability disclosure.**

Controversially, the fourth policy aspect is directed toward the vulnerability discoverer. **As such the fourth recommendation is to remove the default position of last resort by software vulnerability discoverers and replace it with a more inclusive policy.**

Finally, and most impactful is the adoption of secure software practices. As such the **fifth and final recommendation is that software originators adopt secure coding practices within software engineering teams.**

8.5 Further Work

If the debate is to be moved forward, a richer understanding of the key entities and subsystems need to be developed. This research, whilst comprehensive at a system level, requires further research at the subsystem levels of discovery, disclosure and implications of other human activities such as training and education. More broadly, research is needed to determine the power of cyber criminals on the legitimate VDDS, and the influence upon the wider VDDS. For example, it would be interesting to compare experiences of vulnerability discoverers who have been both white hat and black hat discoverers, and why they in inhabit both worlds. Therefore, it is suggested that the augmentation of cybercriminal and illegitimate system dynamic models are incorporated into the

VDDS model and scenarios simulated and investigation between VDM's and the hypothesised vulnerability lifecycles.

Additionally, further work could usefully explore how the quality assurance policies of large software originator organisations (i.e. Microsoft) have affected the VDDS, over time. This would establish a historical record of the quality of software, and how it has improved. Moreover, the precise mechanism of how sentiment and emotional aspects of participants within the VDDS is transmitted between each other (outside of publicly accessible forums) is unknown. There is, therefore, a definite need for investigations or direct data collection from participants to fully characterise this. A practical limitation of this is the covert, or demonised nature of the VDDS, however this is changing, so opportunity may arise in the future.

8.6 Conclusion

What are the factors that contribute to the growth in software vulnerabilities? The conclusion of this research is that no one factor is the sole reason for the growth, but a set. This set includes definite reasons such as poor software quality, increased number of software discoverers and the introduction of rewards. However, these factors are a small subset of the probable causes. External aspects such malicious threat actors, an economy dependent upon information systems and human curiosity, whilst not examined within this research in any depth anecdotally have been uncovered. The resultant picture is a vulnerability discovery and disclosure system that is complex, yet operates within strict rules, ethical boundaries and adopts a sophisticated level of self-regulation.

The key findings can be summed up into two key points: 1) if a vulnerability discoverer finds and reports a vulnerability then treat them with respect; and 2) improve communication with vulnerability discoverers. This coupled with the single most important factor that has contributed to the growth of disclosed vulnerability; the poor quality of software that is produced, and continues to be produced by software originators. The increase in the quality would mitigate and

decrease the quantity of vulnerabilities within software, thus reducing the risks faced.

In conclusion, we can deduce that the factors that are driving the growth of software vulnerabilities are incentives via the introduction of new market instruments, such as the HackerOne disclosure platform and poor-quality software making the ease of discovery higher and the number of participants actively looking for vulnerabilities. Coupled together these three factors have contributed, and arguably will continue to contribute to the growth of vulnerabilities. There are two distinct epochs in the evolution of the VDDS, with a third almost certainly to emerge over the coming years. The first epoch is an uncontrolled and unchecked growth of vulnerabilities that has caused harm to citizens, governments and commercial organisation alike. It seems growth, whilst unchecked, has, not prevented the rise of online commerce and human communication via the world-wide-web. The second, is centred around a realisation that vulnerabilities are critically important to controlling risk within information systems. As we are still close to the commencement of the second epoch, the utility of rewards and incentives to increase the flow of vulnerabilities via a coordinated and arguably more equitable arrangement is unknown, yet encouraging.

Ultimately, we must recognise that whilst we must aspire towards a utopian ideal, we must out of necessity accept the Faustian pact between software vulnerabilities and use of information systems to conduct trade and communicate globally. Failure to do so will result in significant economic loss, economic cyber fatalities or worse, loss of life. Hopefully this research is the first step in driving evidence based policy making in the critically important area of software vulnerability discovery and disclosure, whilst taking a practical approach to vulnerability centred risk.

REFERENCES

4chan (2017) *4chan.org*. Available at: http://www.4chan.org/ (Accessed: 16 March 2017).

Ablon, L. et al. (2017) *Zero Days , Thousands of Nights The Life and Times of Zero-Day Vulnerabilities and Their Exploits*. RAND Corporation.

Ablon, L. et al. (2014) *Markets for Cybercrime Tools and Stolen Data: Hackers' Bazaar*. RAND Corporation.

Ahmad, N.H. et al. (2011) 'Understanding vulnerabilities by refining taxonomy', *Proceedings of the 2011 7th International Conference on Information Assurance and Security, IAS 2011*, , pp. 25–29.

Akcam, B.K. et al. (2011) 'Major Issues in Mixed Use of Grounded Theory and System Dynamics Approaches in Qualitative Secondary Data', *The 29th International Conference of the System Dynamics Society*., pp. 1–37.

Albin, S. et al. (2001) *Building a system dynamics model : Part 1: conceptualization*. Cambridge, Mass.: MIT.

Algarni, A.M. et al. (2013) 'Most Successful Vulnerability Discoverers: Motivation and Methods', Daimi, K. et al. (eds.)*Proceedings of the 2013 International Conference on Security and Management*. Las Vegas Nevada, USA: CSREA Press, pp. 3–10.

Alhazmi, O. (2006) *Assessing Vulnerabilities in Software Systems: A Quantatative Approach*, P.hD. Thesis, Colorado State University, USA. Available at: http://www.cs.colostate.edu/~malaiya/pub/omar-dissertation.pdf (Accessed: 22 March 2014)

Alhazmi, O.H. et al. (2005) 'Quantitative vulnerability assessment of systems software', *Reliability and Maintainability Symposium, 2005. Proceedings. Annual*, , pp. 615–620.

Alhazmi, O.H. et al. (2007) 'Measuring, analyzing and predicting security vulnerabilities in software systems', *Computers and Security*, 26(3), pp. 219–228.

Allison., P. (2010) *Survival Analysis Using SAS: A Practical Guide; Second Edition*. SAS Institute.

Allodi, L. et al. (2014) 'Comparing Vulnerability Severity and Exploits Using Case-Control Studies', *ACM Transactions on Information and System Security*, 17(1), pp. 1–20.

Anderson, R. (2001a) 'Why information security is hard - An economic perspective', *Proceedings - Annual Computer Security Applications Conference,*

ACSAC, 2001–Janua, pp. 358–365.

Anderson, R. (2001b) 'Why Information Security is Hard-An Economic Perspective', *Proceedings of the 17th Annual Computer Security Applications Conference*. Washington, DC, USA: IEEE Computer Society. ACSAC '01, p. 358--.

Anderson, R. et al. (2013) 'Measuring the cost of cybercrime', *The Economics of Information Security and Privacy*, , pp. 265–300.

Anderson, R. et al. (2006) 'The Economics of Information Security', *Science*, 314(5799), p. 610 LP-613.

Apple (2017) *Contact Apple About Security Issues*. Available at: https://support.apple.com/en-us/HT201220 (Accessed: 1 May 2017).

Arbaugh, W.A. et al. (2000) 'Windows of Vulnerability: A Case Study Analysis', *Computer*, 33(12) Los Alamitos, CA, USA: IEEE Computer Society Press, pp. 52–59.

Arciszewski, S. (2015) *Just Don't Use of Trust Bullhorn*. Available at: http://seclists.org/fulldisclosure/2015/Sep/18 (Accessed: 19 December 2015).

Arora, A. et al. (2010) 'Competition and patching of security vulnerabilities: An empirical analysis', *Information Economics and Policy*, 22(2), pp. 164–177.

Arora, A. et al. (2005) 'An Empirical Analysis of Vendor Response to Software Vulnerability Disclosure', *SSRN Electronic Journal*.

Aslam, T. (Purdue U. (1995) *A Taxonomy of Security Faults in the UNIX Operating System*. M.Sc. Thesis, Purdue University, Available at: https://cwe.mitre.org/documents/sources/ATaxonomyofSecurityFaultsintheUNIXOperatingSystem[Aslam95].pdf (Accessed: 27 May 2016).

Ayyub, B.M. et al. (1997) *Solutions manual for Probability, statistics, and reliability for engineers*. Boca Raton, Fla: CRC Press.

Baccianella, S. et al. (2010) 'SentiWordNet 3 . 0 : An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining SentiWordNet', *Analysis*, 0, pp. 1–12.

Banks, J. (2015) '*The Heartbleed bug: Insecurity repackaged, rebranded and resold*', 11(3) SAGE Publications Ltd, pp. 259–279.

Barkin, S. et al. (1999) 'What pediatricians can do to further youth violence prevention--a qualitative study', *Injury Prevention*, 5(1), pp. 53–58.

Bashar, M.A. et al. (1997) 'Low-threat security patches and tools', *Conference on Software Maintenance*., pp. 306–313.

Baxter, P. et al. (2008) ‘Qualitative Case Study Methodology: Study Design and Implementation for Novice Researchers’, *The Qualitative Report Volume*, 13(4), pp. 544–559.

Bazerley, P. et al. (2013) *Qualitative Data Analysis with Nvivo*. SAGE Publications Ltd.

Beattie, S. et al. (2002) ‘Timing the application of security patches for optimal uptime’, *Proceedings of the 16th USENIX conference on System administration*, , pp. 233–242.

Bellovin, S.M. (1989) ‘Security Problems in the TCP / IP Protocol Suite’, *Communication*, 19(2), pp. 32–48.

BenDor, T.K. et al. (2012) ‘A theory of spatial system archetypes’, *System Dynamics Review*, 28(2), pp. 109–130.

Bilge, L. et al. (2012) ‘Before we knew it: An empirical study of zero-day attacks in the real world’, *Proceedings of the ACM Conference on Computer and Communications Security*.

Böhme, R. (2005) ‘Vulnerability markets’, *In Chaos Communication Congress*, (December), pp. 27–30.

Borgen, C. (2013) *Regulating the Global Market of Zero-Day Exploits*., *Opinio Juris*

Bradbury, D. (2015) ‘Why fair disclosure is so difficult’, *Computer Fraud & Security*, 2015(11) Elsevier Ltd, pp. 5–8.

Bradbury, D. (2007) ‘WabiSabiLabi launches vulnerability market’, *Network Security*, 2007(8), pp. 1–2.

Braun, V. et al. (2006) ‘Using thematic analysis in psychology’, *Qualitative Research in Psychology*, 3(May 2015), pp. 77–101.

Braun, V. et al. (2003) Liability or Asset? Women Talk About Their Vagina *Psychology of Women*.

Brocious, C. (2015) *Responsible Disclosure Can Be Anything But*. Available at: http://daeken.com/2012-12-06_Responsible_Disclosure_Can_Be_Anything_But.html (Accessed: 24 February 2015).

Brodie, I. et al. (2014) ‘A general phenomenological model for work function’, *Surface Science*, 625, pp. 112–118. Available at: 10.1016/j.susc.2014.03.002 (Accessed: 14 March 2016).

Brodsky, J. et al. (2011) *Corporate Hacking and Technology-Driven Crime*, Information Science Reference.

Browne, H.K. et al. (2001) ‘A trend analysis of exploitations’, *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*. IEEE Comput. Soc, pp. 214–229.

BugCrowd Homepage (2017) ‘*Bugcrowd*' (Accessed: 23 Feb 2016)

Caballero, J. et al. (2011) ‘Measuring Pay-per-Install : The Commoditization of Malware Distribution’, *USENIX Security Symposium*, , pp. 13–13.

Casagrande, J.B. et al. (1967) Semantic relationships in Papago folk-definitions *Studies in Southwestern Ethnolinguistics: Meaning and History in the languages of the American Southwest*.

Cash, D. (2005) *Mediated Modeling:A System Dynamics Approach to Environmental Consensus Building*.

Castelfranchi, C. et al. (2001) ‘The role of trust and deception in virtual societies’, *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, 6(3), pp. 55–70.

Cavusoglu, H. et al. (2007) ‘Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge’, *IEEE Transactions on Software Engineering*, 33(3), pp. 171–185.

CERT (2017) *Vulnerability Disclosure Policy*. Available at: http://www.cert.org/vulnerability-analysis/vul-disclosure.cfm? (Accessed: 19 March 2017).

Chambers, J.M. (1983) ‘*Graphical methods for data analysis*’, Belmont, Calif.; Boston: Wadsworth International Group ; Duxbury Press.

Chavas, J. (2004) *Risk analysis in theory and practice*. Elsevier Inc.

Checkland, P. (1981) *Systems thinking, systems practice*. Chichester: John Wiley.

Chen, K. et al. (2010) ‘*Multi-cycle vulnerability discovery model for prediction*’, 21(9), pp. 2367–2375.

Chidamber, S.R. et al. (1994) ‘A Metrics Suite for Object Oriented Design’, *IEEE Trans. Softw. Eng.*, 20(6) Piscataway, NJ, USA: IEEE Press, pp. 476–493.

Chirgwin, R. (2017) *Google’s Project Zero reveals another Microsoft flaw*., *The Register* Available at: https://www.theregister.co.uk/2017/02/27/google_project_zero_reports_flaw_in_ie_edge/ (Accessed: 30 May 2017).

Chowdhury, I. et al. (2011) ‘Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities’, *Journal of Systems Architecture*, 57(3) Elsevier B.V., pp. 294–313.

Church, R.M. (1979) ‘How to look at data: A review of John W. Tukey’s Exploratory Data Analysis’, *Journal of the Experimental Analysis of Behavior*, 31(3), pp. 433–440.

Churchman, C. (1967) ‘Free for all’, *Managment Science*, 14(April 2016), pp. 141–142.

Clark, S. et al. (2010) ‘Familiarity breeds contempt: The honeymoon effect and the role of legacy code in zero-day vulnerabilities’, *Acsac*, , pp. 251–260.

Clarke, V. et al. (2004) ‘Lesbian and gay parents on talk shows: resistance or collusion in heterosexism?’, *Qualitative Research in Psychology*, 1(3) Routledge, pp. 195–217.

Cohen, E.A. (1997) Analysis for Military Decisions *Foreign Affairs*.

Computer, B.S. (2002) *A glossary of computing terms*. Harlow, England; New York: Addison-Wesley in association with the British Computer Society.

Concas, G. et al. (2007) ‘Power-laws in a large object-oriented software system’, *IEEE Transactions on Software Engineering*, 33(10), pp. 687–708.

Coyle, R.G. (1977) *Management system dynamics*. Chichester: John Wiley and Sons.

Coyle, R.G. (1996) *System dynamics modelling : a practical approach*. London: Chapman & Hall.

Creswell, J.W. (2013) *Qualitative inquiry and research design : choosing among five approaches*. Los Angeles: SAGE Publications.

Creswell, J.W. (2014) *Research design : qualitative, quantitative, and mixed methods approaches*. Thousand Oaks, California: SAGE Publications.

Culy, C. et al. (2010) ‘Double Tree: An Advanced KWIC Visualization for Expert Users’, *2010 14th International Conference Information Visualisation*. IEEE, pp. 98–103.

Damontoo (2012) *Etsy has been one of the best companies I’ve reported holes to.*, *Reddit/r/netsec* Available at: https://www.reddit.com/r/netsec/comments/vbrzg/etsy_has_been_one_of_the_best_companies_ive/ (Accessed: 24 February 2015).

Dan (2011) *0day Full disclosure: American Express*. Available at: http://qnrq.se/full-disclosure-american-express/#comment-52 (Accessed: 18 April 2016).

Delamore, B. et al. (2015) ‘*A global, empirical analysis of the shellshock vulnerability in web applications*’, Institute of Electrical and Electronics Engineers Inc., Vol.1, pp. 1129–1135.

Douce, C.R. et al. (1999) '*Spatial measures of software complexity*', , p. 10.

Dunn, M. (2005) 'The socio-political dimensions of critical information infrastructure protection (CIIP)', *International Journal of Critical Infrastructures*, 1(2/3), p. 258.

Easley, D. et al. (2010) 'Networks , Crowds , and Markets : Reasoning about a Highly Connected World', *Science*, 81, p. 744.

Eduard Kovacs (2011) *Softpedia Exclusive Interview: Benjamin Kunz Mejri, Vulnerability Laboratory Founder*. Available at: http://news.softpedia.com/news/Softpedia-Exclusive-Interview-Benjamin-Kunz-Mejri-Vulnerability-Laboratory-Founder-228545.shtml (Accessed: 17 November 2014).

Egelman, S. et al. (2013) 'Markets for zero-day exploits', *Proceedings of the 2013 workshop on New security paradigms workshop*, , pp. 41–46.

Elish, K.O. et al. (2008) 'Predicting defect-prone software modules using support vector machines', *Journal of Systems and Software*, 81(5), pp. 649–660.

Endgame (2017) *Endgame About*. Available at: https://www.endgame.com/why-endgame (Accessed: 21 March 2017).

Enoughalready et al. (2012) *What to do when a company refuses to fix a vulnerability I disclosed to them?*. Available at: https://www.reddit.com/r/hacking/comments/ruf9w/what_to_do_when_a_company_refuses_to_fix_a/ (Accessed: 18 April 2016).

Esuli, A. et al. (2006) 'Sentiwordnet: A publicly available lexical resource for opinion mining', *Proceedings of LREC*. Citeseer, Vol.6, pp. 417–422.

ExploitDB (2017) *ExploitDB*.(Accessed: 17 Feb 2017)

Facebookxss (2010) *I found an XSS on facebook.com and am able to steal httpOnly cookies, now what?*. Available at: https://news.ycombinator.com/item?id=1708255 (Accessed: 18 March 2017).

Fidler, M. (2014) *Anarchy or Regulation: Controlling the Global Trade in Zero-Day Vulnerabilities*. Stanford University Center for International Security and Cooperation.

Finifter, M. et al. (2013) 'An empirical study of vulnerability rewards programs', *Proceedings of the 22nd USENIX Security Symposium*, , pp. 273–289.

Flick, U. (2014) *An introduction to qualitative research*. Los Angeles [etc.]: SAGE.

Flick, U. (2009) 'An Introduction To Qualitative Fourth Edition Research', *SAGE*

Publications, , p. 506.

Forrester, J.W. (1975) *The collected works of Jay W. Forrester.* Cambridge, Mass.: Wright-Allen Press.

Forrester, J.W. (1958) 'Industrial dynamics-a major breakthrough for decision makers', *Harvard business review*, 36(4) HARVARD BUSINESS SCHOOL PUBLISHING CORPORATION 60 HARVARD WAY, BOSTON, MA 02163, p. 37.

Francis, W.N. (1965) 'A Standard Corpus of Edited Present-Day American English', *College English*, 26(4) National Council of Teachers of English, pp. 267–273.

Frei, S. (2009) *Security econometrics: The dynamics of (in)security*, P.hD. Thesis, ETH Zurich, Switzerland. Available at: https://doi.org/10.3929/ethz-a-005887804 (Accessed: 27 May 2013)

Frei, S. (2013a) *The Known Unknowns*. Austin, TX, USA.

Frei, S. (2013b) *International Vulnerability Purchase Program*. Austin, TX, USA.

Frei, S. et al. (2010) 'Modeling the Security Ecosystem -- The Dynamics of (In)Security', *Economics of Information Security and Privacy*, , pp. 79–106.

Gallagher, R. (2013) *Cyberwar's Gray Market*., *Slate.com* Available at: http://www.slate.com/articles/technology/future_tense/2013/01/zero_day_exploits_should_the_hacker_gray_market_be_regulated.html (Accessed: 12 November 2014).

Gallagher, T. et al. (2006) *Hunting Security Bugs*. Redmond, WA, USA: Microsoft Press.

Garfinkel, S. et al. (2009) 'Finding and Archiving the Internet Footprint', *Digital Lives Research Conference: Personal Digital Archives for the 21st Century*, (February), pp. 1–14.

Garrison, K. (1993) 'Estimating Defects in Commercial Software During Operational Use', *IEEE Transactions on Reliability*, 42(1), pp. 107–115.

Gastmans, R. et al. (2011) 'Higgs production at the large hadron collider: Phenomenological model and theoretical predictions', *Nuclear Physics B*, 850(1), pp. 53–95. Available at: 10.1016/j.nuclphysb.2011.04.012 (Accessed: 1 May 2016).

Geers, K. (2010) 'The challenge of cyber attack deterrence', *Computer Law and Security Review*, 26(3) Elsevier Ltd, pp. 298–303.

Gimenes, M. et al. (2016) '*Worldlex: Twitter and blog word frequencies for 66 languages*', 48(3) Springer New York LLC, pp. 963–972.

Giordano, F.R. (2009) *A first course in mathematical modeling*. [S.l.]: Brooks/Cole Cengage Learning.

Goldsmith, D. et al. (2013) 'Systematic Approaches to Cyber Insecurity', *The 31st Conference of the System Dynamics Society*, , p. 11.

Goncharov, M. (2014) *Russian Underground Revisited*. Irving, TX, USA.

Goodin, D. (2017a) NSA-leaking Shadow Brokers just dumped its most damaging release yet, *Ars Technica*,

Goodin, D. (2017b) *Wanna Decryptor: The NSA-derived ransomware worm shutting down computers worldwide*., *Ars Technica* Available at: https://arstechnica.co.uk/security/2017/05/what-is-wanna-decryptor-wcry-ransomware-nsa-eternalblue/ (Accessed: 14 May 2017).

Goodwin, B.C. (1970) 'Model of the bacterial growth cycle: Statistical dynamics of a system with asymptotic orbital stability', *Journal of Theoretical Biology*, 28(3), pp. 375–391. Available at: 10.1016/0022-5193(70)90076-7 (Accessed: 19 February 2017).

Google (2017) *Google Security Reward Programs*. Available at: https://www.google.com/about/appsecurity/reward-program/index.html (Accessed: 12 February 2017).

Google (2016) *Project Zero* Available at: http://googleprojectzero.blogspot.co.uk/ (Accessed: 4 June 2016).

Green, J. (2016) *The end of globalisation?*., *Speri.Comment:Pollitical Economy Blog* Available at: http://speri.dept.shef.ac.uk/2016/12/13/the-end-of-globalisation/ (Accessed: 19 February 2017).

Grier, C. et al. (2012) 'Manufacturing Compromise: The Emergence of Exploit-as-a-Service', *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*, , pp. 821–832.

Gupta, S. et al. (1991) '*Towards a theory of penetration-resistant systems and its applications*', , pp. 62–78.

Hackerone (2016) *HackerOne Homepage*., *Hackerone Homepage* Available at: https://hackerone.com/ (Accessed: 4 June 2016).

Hackerone (2017) *How should we price bounties?*., *HackerOne Help Centre* Available at: https://support.hackerone.com/hc/en-us/articles/204950689-How-should-we-price-bounties- (Accessed: 25 February 2017).

HackerOne (2017) *Vulnerability Disclosure Guidelines*. Available at: https://www.hackerone.com/disclosure-guidelines (Accessed: 9 April 2017).

Halstead, M.H. (1977) *Elements of software science*. New York; Oxford;

Amsterdam: Elsevier.

Hamer, P. et al. (1982) ‘MH Halstead’s Software Science-a critical examination’, *… of the 6th international conference on Software …*, , pp. 197–206.

Hartwig, F. et al. (1979) *Exploratory data analysis*. Beverly Hills: Sage Publications.

Hassan, A.E. (2009) ‘Predicting faults using the complexity of code changes’, *Proceedings - International Conference on Software Engineering*, , pp. 78–88.

Hekimoglu, M. et al. (1996) *An Introduction to Sensitivity Analysis*.

HMG (2016) *National Cyber Security Strategy*.

Holm, H. et al. (2013) ‘Effort estimates on web application vulnerability discovery’, *Proceedings of the Annual Hawaii International Conference on System Sciences*, , pp. 5029–5038.

Holt, T.J. et al. (2012) ‘Know Your Enemy: The Social Dynamics of Hacking’, *The Honeynet Project*, , pp. 1–17.

Howard, M. et al. (2002) *Writing secure code* Microsoft Press, Redmond, Wash.

Huang, C. et al. (2016) ‘An Empirical Study of Web Vulnerability Discovery Ecosystems’, *Computers and Security*, 58(0) Elsevier Ltd, pp. 1105–1117.

I_didnt_do_that (2011) *I found a serious security flaw in a social website. What’s the next step?.*, *Reddit/r/netsec* Available at: https://www.reddit.com/r/netsec/comments/f647p/i_found_a_serious_security_flaw_in_a_social/ (Accessed: 24 February 2015).

Ibe, O.C. (2011) *Fundamentals of stochastic networks* John Wiley & Sons, Hoboken, N.J.

Initiative, Z.D. (2017) *Disclosure Policy*. Available at: http://www.zerodayinitiative.com/advisories/disclosure_policy/ (Accessed: 19 March 2017).

ISO/IEC (2005) *Information technology — Security techniques — Code of practice for information security management*.

ISO/IEC (2014) *ISO/IEC 29147:2014(en) Information technology — Security techniques — Vulnerability disclosure*.

ISO/IEC 13335-1 (2004) ‘*Information technology — Security techniques - Management of information and communications technology security- Part1: Concepts and models for Information and communications technology Security Management*’, 2004

Janes, A. et al. (2006) 'Identification of defect-prone classes in telecommunication software systems using design metrics', *Information Sciences*, 176(24), pp. 3711–3734.

Jayaratna, N. (1994) *Understanding and evaluating methodologies : NIMSAD, a systemic framework*. London [u.a.: McGraw-Hill.

Jessaustin (2012) *What Can Happen in the Course of Vulnerability Disclosure*. Available at: https://news.ycombinator.com/item?id=10206083 (Accessed: 18 April 2016).

Joh, H. et al. (2010) *Modeling Skewness in Vulnerability Discovery Models in Major Operating Systems, Available at:https://core.ac.uk/display/21400423*

Joh, H.C. et al. (2014) 'Modeling skewness in vulnerability discovery', *Quality and Reliability Engineering International*, 30(8), pp. 1445–1459.

Johnson, P. et al. (2016) 'Time between vulnerability disclosures: A measure of software product vulnerability', *Computers and Security*, 62, pp. 278–295.

JonnieCache (2014) *4chan announces vulnerability disclosure program*. Available at: https://news.ycombinator.com/item?id=7705419 (Accessed: 18 April 2016).

Kaipower (2002) *Techniques for Vulneability discovery.*, *Seclists.org* Available at: http://seclists.org/vuln-dev/2002/Apr/18 (Accessed: 24 February 2015).

Kapur, P.K. et al. (2015) '*A comparative study of vulnerability discovery modeling and software reliability growth modeling*', Institute of Electrical and Electronics Engineers Inc., pp. 246–251.

Katal, A. et al. (2013) 'Big data: Issues, challenges, tools and Good practices', *2013 6th International Conference on Contemporary Computing, IC3 2013*, , pp. 404–409.

Kdobb (2012) *Ask netsec: what are you experiences reporting vulnerabilities to software companies?*.

Kelly, R.A.. B. et al. (2013) 'Selecting among five common modelling approaches for integrated environmental assessment and management', *Environmental Modelling and Software*, 47(November 2013) Elsevier Ltd, pp. 159–181.

Kim, J. et al. (2007) 'Vulnerability Discovery in Multi-Version Software Systems', *10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*. IEEE, pp. 141–148.

Kim, S. et al. (2016) '*Software Vulnerability Detection Methodology Combined with Static and Dynamic Analysis*', 89(3) Springer New York LLC, pp. 777–793.

Klein, T. (2011) *A Bug Hunter's Diary: A Guided Tour Through the Wilds of Software Security*.

Kock, N. (2007) *Information Systems Action Research*.Kock, N. (ed.) Boston, MA: Springer US, Integrated Series in Information Systems.

Lab, V. (2017) *Bug Bounties, Rewards and Acknowledgements*. Available at: https://www.vulnerability-lab.com/list-of-bug-bounty-programs.php (Accessed: 3 April 2017).

Larsson, E. et al. (2016) '*Papering over the cracks: The effects of introducing best practices on the web security ecosystem*', IEEE Computer Society, Vol.2016–March, pp. 1–6.

Lewis, J.A. et al. (2013) '*The Economic Impact of Cybercrime and Cyber Espionage*', (July), pp. 1–20.

Lewis, P. et al. (2015) 'A Statistical Analysis of Vulnerability Discovery: Microsoft Operating Systems', *Engineering & Technology Reference*

Lewis, T.G. (2006) *Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation*. Wiley-Interscience.

Li, P. et al. (2007) 'An examination of private intermediaries' roles in software vulnerabilities disclosure', *Information Systems Frontiers*, 9(5), pp. 531–539.

Lin, C.-T. (2011) 'Analyzing the effect of imperfect debugging on software fault detection and correction processes via a simulation framework', *Mathematical and Computer Modelling*, 54(11–12), pp. 3046–3064. Available at: 10.1016/j.mcm.2011.07.033 (Accessed: 4 March 2017).

Litchfield, D. (2002) '*Hackproofing Oracle Application Server*', (January), pp. 1–28.

Littlewood, B. (1979) 'The Littlewood-Verrall model for software reliability compared with some rivals', *Journal of Systems and Software*, 1, pp. 251–258.

Malaiya, A.M.A. and Y.K. (2014) 'Software Vulnerability Markets: Discoverers and Buyers', *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 8(3), p. 479–490 EM–algarni2000@hotmail.com.

Malone, E.F. et al. (2013) 'The "wicked problem" of cybersecurity policy: analysis of United States and Canadian policy response', *Canadian Foreign Policy Journal*, 19(2), pp. 158–177.

Manzuik, S. et al. (2007) *Network security assessment : from vulnerability to patch* Syngress Pub. ; O'Reilly Media [distributor], Rockland, Mass.; Sebastopol, Calif.

Marconato, G.V. et al. (2012) 'Security-related vulnerability life cycle analysis',

7th International Conference on Risks and Security of Internet and Systems, CRiSIS 2012

Martin, B. (2000a) *Full Disclosure - Effective or Excuse?*., *synthesis.net*

Martin, B. (2000b) *Full Disclosure - Effective or Excuse?*. Available at: http://attrition.org/~jericho/works/security/full_disclosure.html (Accessed: 27 December 2016).

Martinez, W.L. et al. (2011) *Exploratory data analysis with MATLAB*. Boca Raton, Fla.: CRC Press.

Massie, R. (2015) *Allocating effort: risk and complexity in board directors? engagement with information*. City University London.

Matt4077 (2016) *Remote code execution, git, and OS X*., *Hackernews* Available at: https://news.ycombinator.com/item?id=11519481 (Accessed: 27 December 2016).

Matwyshyn, A.M. et al. (2010) 'Ethics in security vulnerability research', *IEEE Security and Privacy*, 8(2), pp. 67–72.

Maurushat, A. (2013) *Disclosure of Security Vulnerabilities*. London: Springer London, SpringerBriefs in Cybersecurity.

McCabe, T.J. (1976) 'A Complexity Measure', *IEEE Transactions on Software Engineering*, SE-2(4), pp. 308–320.

McQueen, M.A. et al. (2006) '*Time-to-Compromise Model for Cyber Risk Reduction Estimation BT - Quality of Protection: Security Measurements and Metrics*', in Gollmann, D. et al. (eds.) Boston, MA: Springer US, pp. 49–64.

Mcqueen, M.A. et al. (2009) 'Empirical Estimates and Observations of 0Day Vulnerabilities 0Day Vulnerability Lifecycle Begins at Discovery Reported to', *Sciences-New York*, , pp. 1–12.

Meadows, D.H. (1972) *The limits to growth.* Universe Books.

Microsoft (2017) *Microsoft Bounty Programs*. Available at: https://technet.microsoft.com/en-us/library/dn425036.aspx (Accessed: 12 February 2017).

Mikestew (2016) *Controlling vehicle features of Nissan LEAFs remotely via vulnerable APIs*. Available at: https://news.ycombinator.com/item?id=11166852 (Accessed: 21 December 2016).

Miles, M.B. et al. (1984) *Qualitative data analysis : a sourcebook of new methods*. Newbury Park [etc.]: Sage.

Miller, C. (2011) 'Mobile attacks and defense', *IEEE Security and Privacy*, 9(4),

pp. 68–70.

Miller, C. (2007) ‘The Legitimate Vulnerability Market Inside the Secretive World of 0-day Exploit Sales’, *World*, , pp. 1–10.

Mitre (2016) *Common Weakness Enumeration*. Available at: https://cwe.mitre.org/data/definitions/122.html (Accessed: 25 August 2016).

Mohagheghi, P. et al. (2004) ‘An empirical study of software reuse vs. defect-density and stability’, *Proceedings. 26th International Conference on Software Engineering*, (4898), pp. 282–291.

Moore, T. et al. (2010) *Economics of information security and privacy* Springer, New York

Morecroft, J.D.W. (2015) *Strategic modelling and business dynamics : a feedback systems approach*

Moshtari, S. et al. (2013) ‘Using complexity metrics to improve software security’, *Computer Fraud and Security*, 2013(5) Elsevier Ltd, pp. 8–17.

Moulin, B. et al. (2014) ‘The ZoonosisMAGS Project (Part 1): Population-Based Geosimulation of Zoonoses in an Informed Virtual Geographic Environment’, in *Analyzing and Modeling Spatial and Temporal Dynamics of Infectious Diseases*. John Wiley & Sons, Inc., pp. 297–339.

Moussouris, K. et al. (2015) ‘The Wolves of Vuln Street: The 1st System Dynamics Model of the 0day Market’, *RSA Conference 2015*. San FranCisco, USA: RSA Confernces.

Murtaza, S.S. et al. (2016) ‘Mining trends and patterns of software vulnerabilities’, *Journal of Systems and Software*, 117, pp. 218–228.

Musa, J.D. et al. (1984) ‘A Logarithmic Poisson Execution Time Model for Software Reliability Measurement’, *Proceedings of the 7th International Conference on Software Engineering*. Piscataway, NJ, USA: IEEE Press. ICSE ’84, pp. 230–238.

Nagappan, N. et al. (2006) ‘Mining metrics to predict component failures’, *Proceeding of the 28th international conference on Software engineering - ICSE ’06*. New York, New York, USA: ACM Press, p. 452.

NCA (2017) *Pathways Into Cyber Crime*. Available at: http://www.nationalcrimeagency.gov.uk/publications/791-pathways-into-cyber-crime/file (Accessed 21st Jan 2017)

Neff, M. (2016) *Talos Responsible Disclosure Policy Update*. Available at: http://blog.talosintel.com/2016/11/talos-responsible-disclosure-policy.html (Accessed: 22 January 2017).

Neuhaus, S. et al. (2007) 'Predicting vulnerable software components', *Proceedings of the 14th ACM conference on Computer and communications security CCS 07*, , p. 529.

Nguyen, V.H. et al. (2012) 'An Idea of an Independent Validation of Vulnerability Discovery Models', *Proceedings of the 4th International Conference on Engineering Secure Software and Systems*. Berlin, Heidelberg: Springer-Verlag. ESSoS'12, pp. 89–96.

Nguyen, V.H. et al. (2010) 'Predicting vulnerable software components with dependency graphs', *Proceedings of the 6th International Workshop on Security Measurements and Metrics - MetriSec '10*, , p. 1.

Nissan (2017) *Nissan Leaf Front Page*. Available at: https://www.nissan.co.uk/vehicles/new-vehicles/leaf.html (Accessed: 19 March 2017).

NIST (2012) *Engineering Statistics Handbook*., *NIST* Available at: http://www.itl.nist.gov/div898/handbook/eda/section3/runseqpl.htm (Accessed: 23 November 2016).

Nom (2015) *How do I find vulnerabilities in software?*., *security.stackexchange.com* Available at: http://security.stackexchange.com/questions/91990/how-do-i-find-vulnerabilities-in-software (Accessed: 5 February 2017).

O'Gorman, Gavin McDonald, G. (2012) *The Elderwood Project*. Mountain View, CA.

Okamura, H. et al. (2009) 'Optimal security patch release timing under non-homogeneous vulnerability-discovery processes', *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, , pp. 120–128.

Okhravi, H. et al. (2008) 'Evaluation of patch management strategies', *International Journal of Computational Intelligence: Theory and Practice*, 3(2), pp. 109–117.

OpenBugBounty (2017) *Openbugbounty*. Available at: https://www.openbugbounty.org/ (Accessed: 12 February 2017).

Oracle (2017) *How to Report Security Vulnerabilities to Oracle*.

Orman, H. (2003) Erratum: The Morris worm: A fifteen-year perspective (IEEE Security & Privacy (September/October 2003)) *IEEE Security and Privacy*.

Ostler, G. et al. (1983) *The Little Oxford dictionary of current English*. Oxford: Clarendon Press.

Ozment, A. (2007) *Vulnerability discovery & software security*, P.hD Thesis, Cambridge University, UK. Available at:

http://andyozment.com/papers/ozment_dissertation.pdf. (Accessed: 12 November 2011)

Pak (2011) *0day Full disclosure: American Express*. Available at: https://news.ycombinator.com/item?id=3082498 (Accessed: 11 April 2016).

Palm, W.J. (2010) *System Dynamics*. USA: McGraw-Hill Science/Engineering/Math.

Parnas, D.L. (1972) 'On the criteria to be used in decomposing systems into modules', *Communications of the ACM*, 15(12), pp. 1053–1058.

Pasian, B. (2015) *Designs, methods and practices for research of project management*. Spinger.

Patton, M. (1990) 'Qualitative Evaluation and Research Methods', *Qualitative Evaluation and Research Methods*, , pp. 169–186.

Pham, H. (2000) *Software reliability*. Singapore; New York: Springer.

Pham, H. (2006) *Springer handbook of engineering statistics* Springer, London

Polatin-Reuben, D. et al. (2013) 'A system dynamics model of cyber conflict', *Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, , pp. 303–308.

Portnoy, A. (2010) 'Pwn2Own wrap up and analysis', *Network Security*, 2010(4), pp. 4–5.

Pye, G. et al. (2011) '*Proceedings of the 10th European Conference on Information Warfare and Security*', Ottis, R. (ed.) Tallinn, Estonia: Academic Publishing Limited, p. 341.

Radianti, J. (2010a) 'Eliciting information on the vulnerability black market from interviews', *Proceedings - 4th International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2010*, , pp. 154–159.

Radianti, J. (2010b) 'A study of a social behavior inside the online black markets', *Proceedings - 4th International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2010*, , pp. 189–194.

Radianti, J. et al. (2007a) 'A Preliminary Model of the Vulnerability Black Market ', *Proceedings of the 2007 International Conference of the System Dynamics Society*

Radianti, J. et al. (2006) 'Toward a Dynamic Modeling of the Vulnerability Black Market', *The Workshop on the Economics of Securing the Information Infrastructure*, , p. 19.

Radianti, J. et al. (2007b) 'Using a Mixed Data Collection Strategy to Uncover

Vulnerability Black Markets', *Workshop for Information Security .*

Radianti, J. et al. (2009) 'Vulnerability Black Markets: Empirical evidence and scenario simulation', *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*, , pp. 1–10.

Rahimi, S. et al. (2013) 'Vulnerability scrying method for software vulnerability discovery prediction without a vulnerability database', *IEEE Transactions on Reliability*, 62(2), pp. 395–407.

Reddit (2017) *Reddit.com*. Available at: https://www.reddit.com/ (Accessed: 16 March 2017).

Rescorla, E. (2005) 'Is finding security holes a good idea?', *IEEE Security and Privacy*, 3(1), pp. 14–19.

Rescorla, E. (2003) 'Security Holes... Who Cares?', *Proceedings of the 12th USENIX Security Symposium*, , pp. 75–90.

Ring, T. (2015) 'White hats versus vendors: the fight goes on', *Computer Fraud & Security*, 2015(10) Elsevier Ltd, pp. 12–17.

Ritchey, T. (2011) *Wicked Problems – Social Messes*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Roumani, Y. et al. (2015) 'Time series modeling of vulnerabilities', *Computers & Security*, 51(0) Elsevier Ltd, pp. 32–40.

Ruohonen, J. et al. (2016) '*Software Vulnerability Life Cycles and the Age of Software Products: An Empirical Assertion with Operating System Products*', in , pp. 207–218.

Ruohonen, J. et al. (2015) 'The sigmoidal growth of operating system security vulnerabilities: An empirical revisit', *Computers & Security*, 55 Elsevier Ltd, pp. 1–20.

Ryan, G.W. et al. (2003) 'Techniques to Identify Themes', *Field Methods*, 15(1), pp. 85–109.

Saeed, K. (1998) 'Defining a Problem or Constructing a Reference Mode', *16th International Conference of the System Dynamics Society, Québec City, Canada*, , pp. 1–29.

Saeed, K. (1992) 'Slicing a complex problem for system dynamics modeling', *System Dynamics Review*, 8(3), pp. 251–261.

Saldana, J. (2016) *The coding manual for qualitative researchers*. Los Angeles: SAGE.

Saltelli, A. et al. (2007) *Global Sensitivity Analysis. The Primer*. Chichester, UK:

John Wiley & Sons, Ltd.

Sarphim (2011) *Ask netsec: what are you experiences reporting vulnerabilities to software companies?*. Available at: https://www.reddit.com/r/netsec/comments/n2vtk/ask_netsec_what_are_you_experiences_reporting/ (Accessed: 13 November 2014).

Saunders, M.N.K. et al. (2009) *Research methods for business students*. Harlow [etc.]: Financial Times/Prentice Hall : an imprint of Pearson Education.

Scambray, J. et al. (2009) *Hacking Web Applicatipons*. McGraw-Hill/Osborne.

Schechter, S.E. (2004) *Computer Security Strength & Risk: A Quantitative Approach*. Harvard University Cambridge.

Schneier, B. (2007) 'Schneier: Full Disclosure of Security Vulnerabilities a "Damned Good Idea"', *CSO Online.*

Schneier, B. (2015) *Data and Goliath : the hidden battles to capture your data and control your world.* New York: W.W. Norton.

Schneier, B.R.M. (2008) *Face-Off: Is vulnerability research ethical?*., *Techtarget.com* Available at: http://searchsecurity.techtarget.com/magazineContent/Face-Off-Is-vulnerability-research-ethical (Accessed: 12 November 2014).

Schwartz, A. et al. (2016) *Government ' s Role in Vulnerability Disclosure Vulnerability Equities Process*. Boston, MA, USA.

Schweitzer, D. (2003) *Incident Response*. John Wiley & Sons.

Seclists.org (2002) *Full Disclosure Emal Archive*. Available at: https://nmap.org/mailman/listinfo/fulldisclosure (Accessed: 6 November 2016).

Senge, P.M. (1990) *The fifth discipline : the art and practice of the learning organization*. New York: Doubleday/Currency.

Service, C.P. (2017) *Memorandum of Understanding between Crown Prosecution Service (CPS) and the Association of Chief Police Officers (ACPO) concerning Section 46 Sexual Offences Act 2003*. Available at: http://www.cps.gov.uk/publications/agencies/mouaccp.html (Accessed: 1 May 2017).

Shahzad, M. et al. (2012) 'A Large Scale Exploratory Analysis of Software Vulnerability Life Cycles', *Proceedings of the 34th International Conference on Software Engineering*. Piscataway, NJ, USA: IEEE Press. ICSE '12, pp. 771–781.

Shen, V.Y. et al. (1983) 'Software Science Revisited: A Critical Analysis of the Theory and Its Empirical Support', *IEEE Transactions on Software Engineering*,

SE-9(2), pp. 155–165.

Shepperd, M. (1988) ‘A critique of cyclomatic complexity as a software metric’, *Software Engineering Journal*, 3(2), p. 30.

Shim, C. et al. (2014) ‘Tourism, place and placelessness in the phenomenological experience of shopping malls in Seoul’, *Tourism Management*, 45, pp. 106–114. Available at: 10.1016/j.tourman.2014.03.001 (Accessed: 16 March 2016).

Shin, Y. et al. (2011) ‘Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities’, *IEEE Transactions on Software Engineering*, 37(6), pp. 772–787.

Siegel, N.G. (2013) ‘The challenges of emerging software eco-systems (keynote)’, *Proceedings of the 2013 International Conference on Software and System Process - ICSSP 2013*, , pp. 1–8.

Singer, P.W. et al. (2014) *Cybersecurity and Cyberwar What Everyone Needs to Know*.

Sintonen, H. (2017) *QNAP QTS 4.2.x multiple vulnerabilities*. Available at: http://seclists.org/fulldisclosure/2017/Feb/35 (Accessed: 2 March 2017).

Sloman, J. et al. (2006) *Economics student workbook*. Harlow (Prentice Hall).

Smith, B. (2017) *The need for urgent collective action to keep people safe online: Lessons from last week’s cyberattack*. Available at: https://blogs.microsoft.com/on-the-issues/2017/05/14/need-urgent-collective-action-keep-people-safe-online-lessons-last-weeks-cyberattack/#sm.0000xhhvha1dy7cxupds4unkrturf (Accessed: 15 May 2017).

Sommestad, T. et al. (2012) ‘Effort Estimates for Vulnerability Discovery Projects’, *2012 45th Hawaii International Conference on System Sciences*. IEEE, pp. 5564–5573.

Sood, A.K. (2009) ‘From vulnerability to patch: the window of exposure’, *Network Security*, 2009(2), pp. 14–16.

Sowa, S. et al. (2009) ‘Managing Information Risk and the Economics of Security’, *Business*, (2006), pp. 81–97.

Sterman, J. (2000) *Business dynamics : systems thinking and modeling for a complex world*. Boston: Irwin/McGraw-Hill.

Sterman, J.D. et al. (1997) ‘Unanticipated Side Effects of Successful Quality Programs: Exploring a Paradox of Organizational Improvement’, *Management Science*, 43(4), pp. 503–521.

Stoll, C. (1989) *The cuckoo’s egg : tracking a spy through the maze of computer*

espionage. New York: Doubleday.

Stormehh (2012) *Ask netsec: what are you experiences reporting vulnerabilities to software companies?*. Available at: https://www.reddit.com/r/netsec/comments/n2vtk/ask_netsec_what_are_you_experiences_reporting/c3674hn/ (Accessed: 4 September 2014).

Strauss, Aselem. Corbin, J. (1997) *Grounded Theory in Practice*. SAGE Publications.

Succi, G. et al. (2003) 'Practical assessment of the models for identification of defect-prone classes in object-oriented commercial systems using design metrics', *Journal of Systems and Software*, 65(1), pp. 1–12.

Sutton, M. et al. (2007) *Fuzzing : brute force vulnerability discovery*. Upper Saddle River, NJ: Addison-Wesley profesional.

Takanen, A. et al. (2004) 'Agents of responsibility in software vulnerability processes', *Ethics and Information Technology*, 6(2), pp. 93–110.

Tamai, T. et al. (2002) 'Analysis of software evolution processes using statistical distribution Models', *Proceedings of the international workshop on Principles of software evolution - IWPSE '02*, , p. 120.

Tang, M. et al. (2016) 'Exploiting vulnerability disclosures: Statistical framework and case study', *Proceedings - 2016 Cybersecurity and Cyberforensics Conference, CCC 2016*.

Taylor, C Gibbs, G.R. (2005) *How and What to Code*., *Online QDA Web Site,* Available at: http://onlineqda.hud.ac.uk/Intro_QDA/how_what_to_code.php (Accessed: 4 March 2017).

Tesch, R. (2013) *Qualitative Research : Analysis Types and Software.* Routledge,

Thelwall, M. et al. (2013) 'Topic-based sentiment analysis for the social web: The role of mood and issue-related words', *Journal of the American Society for Information Science and Technology*, 64(8), pp. 1608–1617.

Thelwall, M. et al. (2012) 'Sentiment strength detection for the social web', *Journal of the American Society for Information Science and Technology*, 63(1), pp. 163–173.

Thelwall, M. et al. (2010) 'Sentiment strength detection in short informal text', *Journal of the American Society for Information Science and Technology*, 61(12), pp. 2544–2558.

Toomuchtodo (2013) *Facebook Vulnerability 2013*. Available at: https://news.ycombinator.com/item?id=6238281 (Accessed: 18 April 2016).

Trim, P. et al. (2014) *Cyber security management : A governance, risk and compliance framework*. Farnham: Gower Publishing Limited.

Tukey, J.W. (1980) 'We need both exploratory and confirmatory', *American Statistician*, 34(1), pp. 23–25.

Tukey, J.W. (1977) *Exploratory data analysis*. Reading, Mass.: Addison-Wesley Pub. Co.

Uber (2017) *Uber Disclosure Policy*. Available at: https://hackerone.com/uber (Accessed: 4 April 2017).

UK Cabinet-Office (2011) *The UK Cyber Security Strategy Protecting and promoting the UK in a digital world*.

UK Cabinet Office (2009) *Cyber Security Strategy safety , security and resilience in cyber space*.

Vagle, M.D. (2016) *Crafting phenomenological research* Routledge, Abingdon, Oxon

Vijayan, A. et al. (2016) '*Online soft-error vulnerability estimation for memory arrays*', IEEE Computer Society, Vol.2016–May.

Voinov, A. (2008) *Systems science and modeling for ecological economics*. Amsterdam [etc.]: Elsevier/Academic Press.

Wang, L. et al. (2014) 'K-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities', *IEEE Transactions on Dependable and Secure Computing*, 11(1), pp. 30–44.

Wassenaar (2015) *Wassenaar Arrangement - DUAL-USE LIST - CATEGORY 4 - COMPUTERS*. 4.D.4. International

Watson, A.H. et al. (1996) *NIST Special Publication 500-235 Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*. Gaithersburg, MD.

Wei, T. et al. (2010) 'Secure dynamic code generation against spraying', *Ccs*, , p. 738.

Westrin, P. (2001) 'Critical Information Infrastructure Protection (CIIP)', *Information & Security*, 7, pp. 67–79.

Wolstenholme, E.F. (1996) *System enquiry: a system dynamics approach*. Chichester: John Wiley & Sons.

Woo, S.W. et al. (2011) 'Modeling vulnerability discovery process in Apache and IIS HTTP servers', *Computers and Security*, 30(1) Elsevier Ltd, pp. 50–62.

Ycombinator (2017) *Hackernews.com*. Available at: https://news.ycombinator.com/ (Accessed: 16 March 2017).

Yin, R.K. (1994) *Case Study Research*.Discussion. SAGE Publications Ltd.

Younis, A. et al. (2011) 'Modeling learningless vulnerability discovery using a folded distribution', *Proc. of SAM*., Vol.11, pp. 617–623.

Younis, A. et al. (2016) 'Assessing vulnerability exploitability risk using software properties', *Software Quality Journal*, 24(1), pp. 159–202.

ZDI (2017) *Zero Day Initiative*. Available at: http://www.zerodayinitiative.com/ (Accessed: 12 February 2017).

Zerodium (2017) *Zerodium*. Available at: https://www.zerodium.com/program.html (Accessed: 12 February 2017).

Zhang, H. et al. (2007) 'Predicting Defective Software Components from Code Complexity Measures', *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*. IEEE, pp. 93–96.

APPENDICES

Appendix A – Raw Data Tables

A.1.1 Sensitivity Analysis Rewards Raw Data Sample

1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1.01	1.01	1.01	1.01	1.01	1.01
1	1	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.02
1	1.01	1.01	1.01	1.01	1.01	1.01	1.02	1.02	1.02
1.01	1.01	1.01	1.01	1.01	1.02	1.02	1.02	1.02	1.03
1.01	1.01	1.01	1.01	1.02	1.02	1.02	1.03	1.03	1.03
1.01	1.01	1.01	1.02	1.02	1.02	1.03	1.03	1.03	1.04
1.01	1.01	1.02	1.02	1.02	1.03	1.03	1.03	1.04	1.04
1.01	1.01	1.02	1.02	1.03	1.03	1.03	1.04	1.04	1.05
1.01	1.01	1.02	1.02	1.03	1.03	1.04	1.04	1.05	1.05
1.01	1.02	1.02	1.03	1.03	1.04	1.04	1.05	1.05	1.06
1.01	1.02	1.02	1.03	1.03	1.04	1.05	1.05	1.06	1.06
1.01	1.02	1.03	1.03	1.04	1.04	1.05	1.06	1.06	1.07
1.01	1.02	1.03	1.03	1.04	1.05	1.05	1.06	1.07	1.07
1.02	1.02	1.03	1.04	1.04	1.05	1.06	1.07	1.07	1.08
1.02	1.02	1.03	1.04	1.05	1.05	1.06	1.07	1.08	1.09
1.02	1.03	1.03	1.04	1.05	1.06	1.07	1.07	1.08	1.09
1.02	1.03	1.04	1.04	1.05	1.06	1.07	1.08	1.09	1.1
1.02	1.03	1.04	1.05	1.06	1.06	1.07	1.08	1.09	1.1
1.02	1.03	1.04	1.05	1.06	1.07	1.08	1.09	1.1	1.11
1.02	1.03	1.04	1.05	1.06	1.07	1.08	1.09	1.11	1.12
1.02	1.03	1.04	1.05	1.06	1.08	1.09	1.1	1.11	1.12
1.02	1.03	1.05	1.06	1.07	1.08	1.09	1.1	1.12	1.13
1.02	1.04	1.05	1.06	1.07	1.08	1.1	1.11	1.12	1.14
1.03	1.04	1.05	1.06	1.07	1.09	1.1	1.11	1.13	1.14
1.03	1.04	1.05	1.06	1.08	1.09	1.1	1.12	1.13	1.15
1.03	1.04	1.05	1.07	1.08	1.09	1.11	1.12	1.14	1.15
1.03	1.04	1.06	1.07	1.08	1.1	1.11	1.13	1.15	1.16
1.03	1.04	1.06	1.07	1.09	1.1	1.12	1.13	1.15	1.17
1.03	1.05	1.06	1.08	1.09	1.11	1.12	1.14	1.16	1.18
1.03	1.05	1.06	1.08	1.09	1.11	1.13	1.15	1.16	1.18
1.03	1.05	1.06	1.08	1.1	1.11	1.13	1.15	1.17	1.19
1.03	1.05	1.07	1.08	1.1	1.12	1.14	1.16	1.18	1.2

A.1.2 Sensitivity Analysis Sentiment Raw Data Sample

1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1
2.14	2.8	2.86	2.88	2.9	2.9	2.91	2.91	2.91	2.91
2.65	5	5.32	5.43	5.49	5.53	5.55	5.57	5.58	5.59
3.05	7.45	8.23	8.52	8.67	8.76	8.83	8.87	8.91	8.93
3.42	10.08	11.52	12.06	12.34	12.51	12.63	12.71	12.78	12.83
3.78	12.83	15.08	15.94	16.39	16.66	16.85	16.98	17.08	17.16
4.12	15.65	18.86	20.08	20.72	21.11	21.37	21.56	21.71	21.82
4.45	18.52	22.79	24.41	25.25	25.77	26.11	26.36	26.55	26.69
4.78	21.4	26.82	28.87	29.92	30.56	30.99	31.3	31.53	31.7
5.09	24.27	30.92	33.4	34.67	35.43	35.94	36.3	36.57	36.78
5.39	27.13	35.05	37.98	39.45	40.33	40.91	41.33	41.63	41.87
5.69	29.95	39.18	42.56	44.23	45.22	45.87	46.33	46.67	46.94
5.97	32.72	43.31	47.12	48.98	50.07	50.79	51.29	51.66	51.95
6.24	35.45	47.4	51.64	53.68	54.87	55.64	56.17	56.57	56.88
6.51	38.11	51.46	56.11	58.32	59.59	60.41	60.97	61.39	61.71
6.77	40.72	55.46	60.52	62.88	64.22	65.08	65.68	66.11	66.44
7.02	43.26	59.39	64.85	67.35	68.76	69.66	70.27	70.72	71.06
7.26	45.74	63.26	69.09	71.73	73.2	74.12	74.76	75.22	75.56
7.5	48.15	67.06	73.26	76.01	77.53	78.48	79.12	79.59	79.95
7.72	50.5	70.78	77.33	80.19	81.75	82.72	83.38	83.85	84.21
7.94	52.78	74.42	81.3	84.27	85.86	86.85	87.51	87.99	88.35
8.16	54.99	77.97	85.18	88.24	89.86	90.86	91.53	92	92.36
8.37	57.14	81.44	88.97	92.1	93.75	94.76	95.43	95.9	96.26
8.57	59.22	84.83	92.65	95.86	97.53	98.54	99.21	99.69	100.04
8.76	61.24	88.13	96.24	99.52	101.2	102.21	102.88	103.35	103.71
8.95	63.2	91.34	99.73	103.07	104.77	105.78	106.44	106.91	107.26
9.13	65.09	94.47	103.13	106.52	108.22	109.23	109.89	110.35	110.7
9.31	66.93	97.51	106.43	109.87	111.58	112.58	113.23	113.69	114.03
9.48	68.71	100.47	109.64	113.12	114.83	115.82	116.47	116.92	117.25
9.65	70.44	103.35	112.76	116.27	117.98	118.97	119.61	120.05	120.38
9.81	72.11	106.14	115.78	119.33	121.03	122.01	122.64	123.08	123.4
9.97	73.73	108.86	118.72	122.29	123.99	124.96	125.58	126.01	126.33
10.13	75.3	111.49	121.57	125.17	126.86	127.82	128.43	128.85	129.16

A.1.3 Sensitivity Analysis Software Quality Raw Data Sample

1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0
24.79	20.91	17.3	13.98	10.95	8.2	5.73	3.54	1.64	0
60.69	48.98	38.05	28.27	19.89	13.03	7.73	3.89	1.38	0
67.9	61.18	51.03	39.14	27.32	17.09	9.29	4.11	1.2	0
36.17	42.01	42.9	38.03	28.9	18.55	9.8	3.99	1.02	0
12.74	19.78	26.17	28.67	25.3	17.6	9.42	3.67	0.85	0
4.25	8.29	13.81	18.72	19.62	15.23	8.49	3.24	0.7	0
1.6	3.63	7.16	11.58	14.26	12.44	7.32	2.77	0.57	0
0.7	1.75	3.89	7.17	10.09	9.81	6.12	2.32	0.46	0
0.35	0.94	2.24	4.56	7.12	7.6	5.01	1.92	0.37	0
0.19	0.54	1.38	2.99	5.07	5.85	4.05	1.56	0.29	0
0.12	0.34	0.89	2.03	3.65	4.49	3.24	1.26	0.23	0
0.07	0.22	0.6	1.42	2.67	3.45	2.58	1.02	0.19	0
0.05	0.15	0.42	1.02	1.98	2.66	2.05	0.81	0.15	0
0.03	0.11	0.3	0.74	1.49	2.05	1.62	0.65	0.12	0
0.02	0.08	0.22	0.55	1.13	1.59	1.28	0.52	0.09	0
0.02	0.06	0.16	0.42	0.86	1.24	1.02	0.41	0.07	0
0.01	0.04	0.12	0.32	0.66	0.97	0.8	0.33	0.06	0
0.01	0.03	0.09	0.24	0.52	0.76	0.64	0.26	0.05	0
0.01	0.02	0.07	0.19	0.4	0.6	0.51	0.21	0.04	0
0.01	0.02	0.06	0.15	0.32	0.48	0.4	0.17	0.03	0
0	0.02	0.04	0.12	0.25	0.38	0.32	0.13	0.02	0
0	0.01	0.04	0.09	0.2	0.3	0.26	0.11	0.02	0
0	0.01	0.03	0.07	0.16	0.24	0.21	0.09	0.02	0
0	0.01	0.02	0.06	0.13	0.2	0.17	0.07	0.01	0
0	0.01	0.02	0.05	0.11	0.16	0.14	0.06	0.01	0
0	0.01	0.02	0.04	0.09	0.14	0.12	0.05	0.01	0
0	0	0.01	0.03	0.07	0.11	0.1	0.04	0.01	0
0	0	0.01	0.03	0.06	0.1	0.08	0.03	0.01	0
0	0	0.01	0.02	0.05	0.08	0.07	0.03	0.01	0
0	0	0.01	0.02	0.05	0.07	0.06	0.03	0	0
0	0	0.01	0.02	0.04	0.06	0.06	0.02	0	0
0	0	0.01	0.02	0.04	0.06	0.05	0.02	0	0

A.2 Software Code Used for Analysis

The following is source code used for data analysis and raw statistics generation.

A.2.1 – ExploitDB Participants Python Code

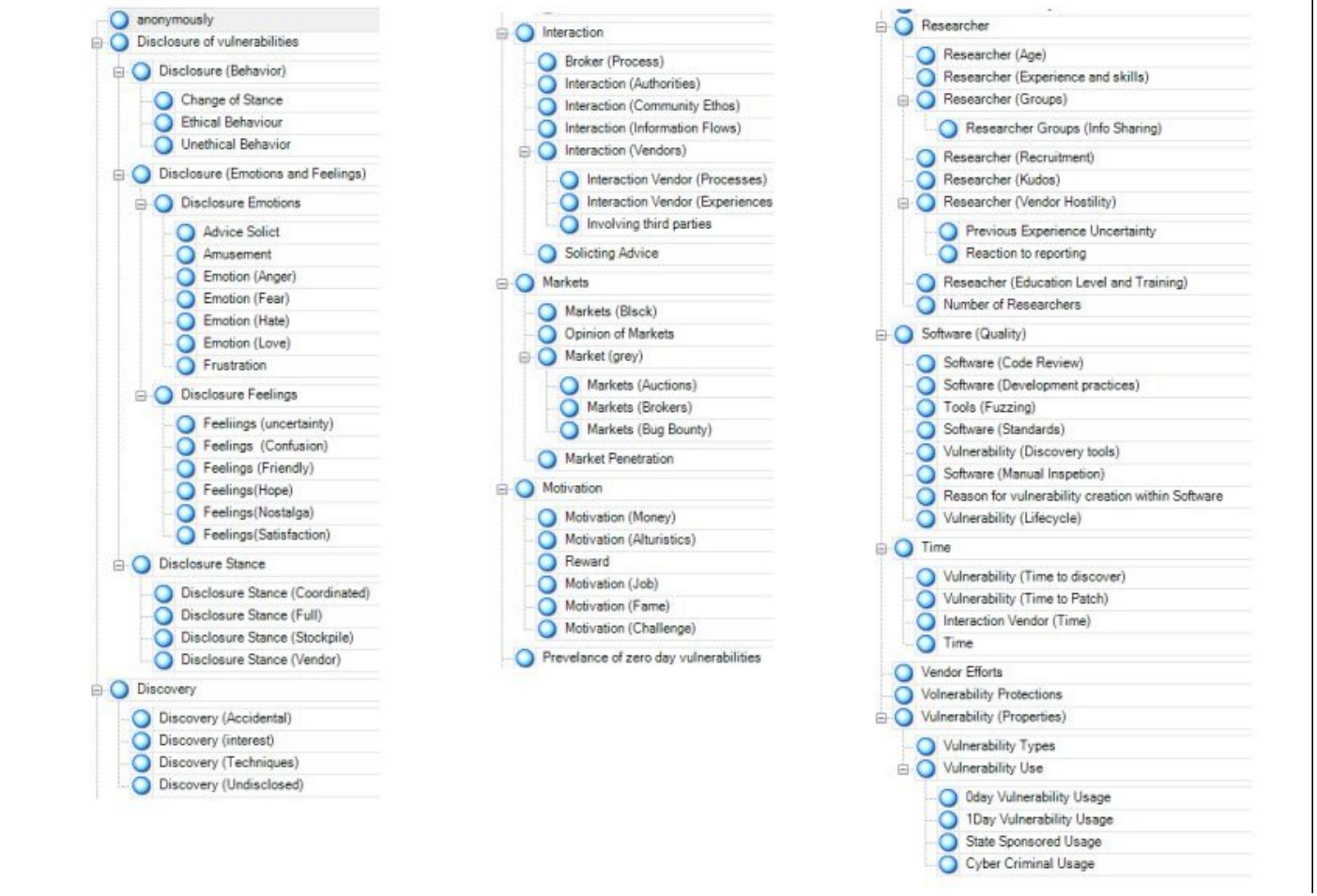
```
import csv, sys, sets
data =[]
listName = []
listDate = []
finalList = []
unique = []
position = []

with open('files.csv', 'rb') as f:
	reader = csv.reader(f)
	for data in reader:
		listDate.append(data[2])
		listName.append(data[3])

for item in listName:
	if item in unique:
		unique.append(item)
		finalList.append(listDate[listName.index(item)])

with open("output.csv", "wb") as f:
	writer = csv.writer(f)
	writer.writerow(finalList)
```

A.3 – Raw Codes


anonymously
Disclosure of vulnerabilities
Disclosure (Behavior)
Change of Stance
Ethical Behaviour
Unethical Behavior
Disclosure (Emotions and Feelings)
Disclosure Emotions
Advice Solict
Amusement
Emotion (Anger)
Emotion (Fear)
Emotion (Hate)
Emotion (Love)
Frustration
Disclosure Feelings
Feeliings (uncertainty)
Feelings (Confusion)
Feelings (Friendly)
Feelings(Hope)
Feelings(Nostalga)
Feelings(Satisfaction)
Disclosure Stance
Disclosure Stance (Coordinated)
Disclosure Stance (Full)
Disclosure Stance (Stockpile)
Disclosure Stance (Vendor)
Discovery
Discovery (Accidental)
Discovery (interest)
Discovery (Techniques)
Discovery (Undisclosed)
Interaction
Broker (Process)
Interaction (Authorities)
Interaction (Community Ethos)
Interaction (Information Flows)
Interaction (Vendors)
Interaction Vendor (Processes)
Interaction Vendor (Experiences
Involving third parties
Solicting Advice
Markets
Markets (Blsck)
Opinion of Markets
Market (grey)
Markets (Auctions)
Markets (Brokers)
Markets (Bug Bounty)
Market Penetration
Motivation
Motivation (Money)
Motivation (Alturistics)
Reward
Motivation (Job)
Motivation (Fame)
Motivation (Challenge)
Prevelance of zero day vulnerabilities
Researcher
Researcher (Age)
Researcher (Experience and skills)
Researcher (Groups)
Researcher Groups (Info Sharing)
Researcher (Recruitment)
Researcher (Kudos)
Researcher (Vendor Hostility)
Previous Experience Uncertainty
Reaction to reporting
Reseacher (Education Level and Training)
Number of Researchers
Software (Quality)
Software (Code Review)
Software (Development practices)
Tools (Fuzzing)
Software (Standards)
Vulnerability (Discovery tools)
Software (Manual Inspetion)
Reason for vulnerability creation within Software
Vulnerability (Lifecycle)
Time
Vulnerability (Time to discover)
Vulnerability (Time to Patch)
Interaction Vendor (Time)
Time
Vendor Efforts
Volnerability Protections
Vulnerability (Properties)
Vulnerability Types
Vulnerability Use
0day Vulnerability Usage
1Day Vulnerability Usage
State Sponsored Usage
Cyber Criminal Usage

A.4 Survey Structure and Ethical Approval

Please provide the following information about your research:

Title of research project or activity	Software vulnerability dynamics within commercial of the shelf software.	
Name of researcher(s) conducting the fieldwork	Paul Lewis, Lecturer in Cyber Defence (Internal PhD staff candidate)	
Email of researcher conducting the fieldwork	P.Lewis@cranfield.ac.uk	
Name and department of staff member responsible for the work (e.g. Principal Investigator / thesis supervisor)	Jeremy Hilton, Centre for Cyber Security and Information Systems, Cranfield Defence and Security, Shrivenham. PhD Supervisor (Internal)	
Email of responsible staff member	J.C.Hilton@cranfield.ac.uk	
Name of research client or sponsor	N/a	
Please indicate if the research is part of a:	Taught Masters	
	MSc by Research	
	MPhil	
	PhD	
	EngD	
	Research Contract	
If it is part of a taught Masters programme please give the title of the course	N/a	
Intended start date of fieldwork	18th Aug 2014	
Intended end date of fieldwork	1st Apr 2015	
Who are the intended research participants? (e.g. those who you will be surveying, observing, or speaking to)	Engineers, security researchers and software developers from both the open source community and commercial enterprises.	

Will the research client or sponsor be providing access to research participants?

N/a

No		
Yes		If yes, please provide detail as to how you will ensure anonymity and confidentiality for your participants in the box below:

All responses will be anonymous, and no personally identifiable data will be collected from the participants. Raw data that will be collected will be held in accordance with the data protection act 1990, and will follow UK government guidelines for data at rest (encrypted data files etc).

All results will be collected via the Cranfield University recommended survey software Qualtrics.com and the survey questions can be found in appendix A below.

A risk assessment is attached to this approval form.

We need to fully understand what information/data is being collected from your participants. Please provide a short description (approximately 150 words) of your research aims, objectives and methodology in the box below.

The aim of the research is to understand the processes that influence the discovery of vulnerabilities within software in everyday use. A theoretical model has been constructed and real-world validation of this model is required to provide confidence in it.

The data that will be collected will be in two forms, questionnaire and interview. Simple demographic data

such as age range and nationality will initially be collected within the survey. Additionally, specific research based questions will be asked which can be found in the questionnaires attached to this proposal. Depending upon which part of the theoretical model is being validated one of two survey templates will be used, both are attached.

Furthermore, specific interviews will be undertaken with a subset of participants, which will allow further exploration of the subject area and model validation.

Section B

Please answer the following questions to help us evaluate the level of risk associated with your research. If you answer 'Yes' to any of the statements in Section B you should prepare and submit a high risk to SEREC using the guidance provided here

Does your proposed research involve;	
[1]Vulnerable groups such as children, people with physiological and/or psychological impairments (e.g. the disabled, mentally impaired, people with learning difficulties)?	
Talking about or referencing sensitive topics (e.g. Sexual behaviour, illegal or political behaviour, experience of violence, abuse or exploitation, mental health, gender or ethnic status conflict situations, psychologically disturbing events?	
Questioning or activities which could risk inducing psychological stress, anxiety or humiliation or cause physical pain or harm?	
Intrusive interventions - for example, the administration of drugs or other substances, physical exercise, or techniques such as hypnotherapy?	
Groups where permission of a gatekeeper is required for initial access to members (e.g. children, residents of institutions)?	
The use of payments and / or incentives to encourage or reward participation?	
Deception, withholding information, or activities which are conducted without participants' full and informed consent at the time the study is carried out?	
Access to records of personal or confidential information, including genetic or other biological information, concerning identifiable individuals?	
The collection of human tissue or other human biological samples?	

[1]If your research involves children or other vulnerable groups; you may need to apply to the Criminal Records Bureau for clearance. Detailed guidance can be found on the CRB website (http://www.direct.gov.uk/crb)

Further details of many of the issues covered in the table can be found in the guidance available on the SEREC website

Section C

Please complete the two tables below using the check boxes on the right hand side. If you cannot confirm all the statements you should prepare and submit a high risk proposal to SEREC using the guidance provided here.

I confirm that as part of the research activity described above;	
I will secure and record the informed consent of all human subjects	
I will ensure that no-one is coerced or compelled to participate in the research	
I will not use any inducements or incentives to secure participation	
I will not use any form of deception as part of the research method	
I will explain to participants the level of confidentiality which they can expect and will aim to maintain participant confidentiality wherever practicable.	
I will design and execute the research in a way which protects participants from harm (including but not restricted to - physical, psychological, emotional, social, spiritual, career, reputational, financial or legal harm)	
I will, prior to any data gathering activity, brief participants about the project and their rights	
I will, prior to any data gathering activity, brief any individuals involved in data gathering on my behalf (e.g. translators or interviewers) about ethical research practices.	
I will, following any data collection activity, debrief participants.	
I will not be using any observationally intrusive methods	
I will store any data I obtain in accordance with the Data Protection Act	
I also confirm that:	
The information I have provided on this form is accurate to the best of my knowledge and belief.	
I have read the advice on research ethics contained on the webpage 'Basic principles of ethical research involving human subjects.'	
The project described above will abide by the University' s Ethics Policy.	
There is no potential material interest that may, or may appear to, impair the independence and objectivity of researchers conducting this project.	
Subject to the research being approved, I undertake to adhere to the project description and statements provided above.	

I undertake to inform SEREC of any significant changes to the research activity which might invalidate the statements made above	
I understand that the project, including research records and data, may be subject to inspection for audit purposes, if required in future.	
I understand that personal data about me as a researcher in this form will be held by those involved in the university ethical research review procedure and that this will be managed according to Data Protection Act principles.	

The person completing this form is the:

Researcher conducting the work

Supervisor of the project

Electronic signature of the researcher conducting the work	Paul Lewis
Electronic signature of the project supervisor	Jeremy Hilton

If you have any queries about this form or the SEREC review process, please email the SEREC administrator at serec@cranfield.ac.uk.

Please email your completed form to serec@cranfield.ac.uk

Security Researcher Survey (provisional –actual questions may vary a little)

Question 1

What is the highest educational level you have completed?
a) Secondary (GCSE, K12 Ages 11-16) b) Tertiary College (As-Level, Highers) c) Undergraduate (B.Sc., HND) d) Postgraduate (M.Sc, PhD) c) None

Question 2

In the last 12 months how many vulnerabilities or exploits have you discovered?
a) None b) 1-5 c) 6-10 d) 11-15 d) 15+

Question3

Do you work in a team or as an individual when researching vulnerabilities?
a) In a team b) Individually

Question 4

How did you learn your security researcher skills?
a) College or university b) Professional qualification (CEH, CREST, SANS, GIAC etc) c) Vendor Qualification (Checkpont, Cisco etc) d) Online (forums, IRC, twitter, Y-combinator, reddit etc)

e) Physical Meetings (2600, B-sides, DEFCON etc)

f) Other

Question 5

How many security researchers do you think are actively looking for vulnerabilities within software globally?

a) 0 - 10000

b) 10001 - 20000

c) 20001 - 30000

d) 30001 - 40000

e) 50000 +

f) Unknown

Question 6

In your opinion which is the most important when discovering vulnerabilities?

a) The tools that you use.

b) The familiarity of the software you are researching.

c) Your skills as a researcher.

d) Motivation to find vulnerabilities.

e) People to discuss ideas with

f) the quality of the software

Question 7

Would you be willing to be interviewed by a Cranfield University researcher?

a) Yes b) No

Appendix B – Software Developer (Provisional specific questions may vary a little)

Question 1

What is the highest educational level you have completed?
a) Secondary (GCSE, K12) b) Tertiary College (As-Level, Highers) c) Undergraduate (B.Sc., HND) d) Postgraduate (M.Sc, PhD) c) Other

Question 2

Do you hold any IT specific qualifications?
a) None held b) Professional Institute (British Computing Society ISEB etc) c) Vendor Certification (Microsoft MCSE, Cisco CCNP, Oracle etc) d) Vocational (City and Guilds, European Computing Driving Licence) e) Information Security Certification (CISSP, Certified ethical hacker H, CISM) f) Other

Question 3

Do you work in a team?
a) yes b) no

Question 4

Which type of software do you normally develop?
a) Application Software (word processor, database) b) Utility Software (firewall, Antivirus etc) c) System (OS Kernel, Device Drivers etc) d) Middleware.

Question 5

Do you use any of the following software development methodologies?
a) Waterfall b) Agile (Scum, Extreme, SDL) c) Incremental (Spiral etc) d) Other

Question 6

Which languages do you use when you are developing the software stated in question 5?
a) C/C++ b) Java c) C#/.Net d) Perl e) Assembler/Machine Code f) Ruby g) Ada

Question 7

How familiar are you with this methodology?
a) A sliding scale of 0-10 will be here.

Question 8

Over the past month how many lines of code have you produced (please include debugging)
b) None c) 100-500

d) 501-1000
e) 1001-1500
f) 1501+

Question 9

On average how many bugs/vulnerabilities have you removed over the past month?

g) None
h) 1-5
i) 6-10
j) 11-15
k) 15+

Appendix C – Risk Assessment

Risk Name – Loss of Survey Data (outside of Qualtrics)

Description – The data provided could be potentially sensitive, as it contains data pertaining to vulnerability discovery and software development methods. Whilst this data is not personally identifiable it could cause embarrassment if lost or stolen.

Risk Level

Low

Mitigation

1. Usage of Cranfield University approved data collection web service.
2. Storage of raw data within Shrivenham secure campus
3. Storage medium used to store survey responses encrypted with AES-256 encryption software and a complex 10 digit passphrase.
4. Storage medium locked in safe storage.

Risk Name – Reputational Damage

Description – The survey and research itself could be potentially seen as coercive as Cranfield has significant associations with the MoD and Intelligence community. This is especially relevant as the research is investigating the methods which researchers undertake when undertaking software audits looking for vulnerabilities within software. Given the target audience this may be considered intrusive in nature.

Risk Level

Medium
Mitigation 1. Ensure that the survey introduction is clear explains that this is a university research initiative and is not UK government or defence sponsored. 2. Provide clear and transparent access to the Science and Engineering ethics committee policies.

Risk Name – Loss of Data (via Qualtrics)
Description – The data provided could be potentially sensitive, as it contains data pertaining to vulnerability discovery and software development methods. The qualtrics web survey service is an unknown quantity and the security assurance of the service is unproven. Given the nature of the survey, and potential participants, there is the potential for data breach via the web service.
Risk Level Low
Mitigation 1. None as this is a third party service with no direct control from cranfield.

Consent Pro-forma

NB. Tacit consent will be sought from all participants in the form of a active indication (ie button press)

We would like to ask you to participate in the data collection for this research project which is looking at the factors that influence vulnerability research. *Before you decide whether or not to take part, it is important for you to understand why the research is being done and what it will involve. Please take time to read the following information carefully'*

The purpose of this survey is to validate a number of assumptions and ask people involved in vulnerability research, such as yourself, whether these views are correct. Your participation in this study is entirely voluntary. The survey includes a small number of questions and filling in the survey will take no more than 2-3 minutes.

The information gained from this survey will only be used for the above objectives, will not be used for any other purpose and will not be recorded in excess of what is required for the research. All responses will be anonymous, and no personally identifiable data will be collected from the participants. Raw data that will be collected will be held in accordance with the Data Protection Act 1990, and will follow UK government guidelines for data at rest.

Even though the study findings will be published in international conferences and journals, only relevant researchers will have access to the survey data itself. These researchers will be bound by the principles outlined above and by the Cranfield University Science and Engineering Ethics Committee. Details of which are available on request.

If you have any questions regarding this study or would like additional information please contact the research leader P.lewis@cranfield.ac.uk or 01793 785490

By filling in this survey you indicate that you understand its purpose and consent to the use of the data as indicated above. Should you decide not to complete the survey, the data you have entered up to that point will be used, unless you indicate otherwise . We will also provide details of the data upon request.

Thank you for your cooperation
Paul