Proactive System for Digital Forensic Investigation

by

Soltan Abed Alharbi
B.S., Florida Institute of Technology, USA, 1998
M.S., Florida Institute of Technology, USA, 2000

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

Proactive System for Digital Forensic Investigation

by

Soltan Abed Alharbi
B.S., Florida Institute of Technology, USA, 1998
M.S., Florida Institute of Technology, USA, 2000

Supervisory Committee

ABSTRACT

Digital Forensics (DF) is defined as the ensemble of methods, tools and techniques used to collect, preserve and analyse *digital* data originating from any type of digital media involved in an incident with the purpose of extracting valid evidence for a court of law.

DF investigations are usually performed as a response to a digital crime and, as such, they are termed *Reactive Digital Forensic* (RDF). An RDF investigation takes the traditional (or post-mortem) approach of investigating digital crimes after incidents have occurred. This involves identifying, preserving, collecting, analyzing, and generating the final report.

Although RDF investigations are effective, they are faced with many challenges, especially when dealing with anti-forensic incidents, volatile data and event reconstruction. To tackle these challenges, *Proactive Digital Forensic* (PDF) is required. By being proactive, DF is prepared for incidents. In fact, the PDF investigation has the ability to proactively collect data, preserve it, detect suspicious events, analyze evidence and report an incident as it occurs.

This dissertation focuses on the detection and analysis phase of the proactive investigation system, as it is the most expensive phase of the system. In addition, theories behind such systems will be discussed. Finally, implementation of the whole proactive system will be tested on a botnet use case (Zeus).

Contents

List of Tables

List of Figures

Acronyms

CC	Command Control
CP	Control Panel
CT	Control Terms
DF	Digital Forensics
DFA	Deterministic Finite Automaton
DFR	Digital Forensic Rules
FSM	Finite State Machine
GUI	Graphical User Interface
HPC	High Performance Computing
I/O	Input/Output
IB	InfiniBand
IDS	Intrusion Detection Systems
MAC	Modification Access Creation
MPI	Message Passing Interface
PA	Proactive Agent
PDF	Proactive Digital Forensic
PS	Primary Studies
PSA	Proactive Sub-Agent

RDF	Reactive Digital Forensic
RDMA	Remote Direct Memory Access
SLR	Systematic Literature Review
SSH	Secure Shell
TM	Turing Machine
UML	Unified Modeling Language
VM	Virtual Machines

Glossary

`Feedback Dynamical System` — an approach to understand the behavior of complex system over time. Also, it deals with feedback loops to adjust the system accordingly, as is the case in our proactive system.

`Feedback System` — the proactive component that takes the output from the forward system (system under investigation) and takes proactive measures to feed it in again as input to the forward system as an adjustment.

T — a set of elements of the digital forensic investigation called Targets.

E — a set of elements of the digital forensic investigation called Events.

A — a set of elements of the digital forensic investigation called Actions; each action is viewed as a transfer function of targets and events.

f — Single target.

$S(f)$ — possible states for target f which it can be in.

$\mathcal{T}$ — the state space of the system's targets.

e — single event.

$S(e)$ — possible status for event e.

$\uparrow$ — triggered.

$\downarrow$ — not triggered.

$\uparrow^t e$ — the event e is triggered at time t.

$\mathcal{E}$ — the state space of all the system's events.

a — an action.

Γ — set representing time.

$\vec{r}$ — vector of targets.

$\vec{e}$ — vector of events.

ψ — the *evolution function* ψ is defined from $\Gamma \times (\mathcal{T} \times \mathcal{E}) \times A$ to $\mathcal{T} \times \mathcal{E}$ by

$$\psi(t, (\vec{r}, \vec{e}), a) = a(t, \vec{r}, \vec{e}).$$

$\emptyset_T$ — empty target.
$\emptyset_E$ — empty event.
$\emptyset_A$ — empty action.
$P^{-1}(T)$ — targets in the domain of P.
$F = (T, E, t)$ — the full space of investigation.
$[E]$ — events equivalence class.
$\sim_E$ — dependency relation: when an event happens the other has to happen.
$\approx_E$ — equivalence relation.
$\cong_E$ — equivalence class relation.
Ψ — mapping from targets to events that associates each target with its change of status event.
$\bar{A}$ — composite actions.
F — full forensic space.
F' — sub-space.
$F = \mathcal{T} \times \mathcal{E} \times \Gamma$ — digital forensic space containing targets, events and time.
$P(S, t_c)$ — incomplete profile of the system.
$P_\alpha(S)$ — family of profiles for the system.
$C(h, g)$ — the correlation between the two profiles h and g.
C_α — the correlation (C) between all the profiles.
$\pi_{(T', E', \Gamma')}$ — projection function that takes a profile in the space F and projects it onto F' to produce a reduced profile.
Θ — digital forensic reduction operator, considered to be an extension operator for the projection function.
MPI_Allreduce — performs a reduction operation (MPI operation) and is used to compute the mean and the standard deviation.
MPI_SUM — performs a collective operation (MPI operation) and is used to compute the mean and the standard deviation collectively.
MPI_MAXLOC — performs a collective operation (MPI operation) to find the maximum value and its location. It is used in selecting the largest value and its location.
size — size as file attribute.
mtime — modification time as file attribute.
atime — access time as file attribute.
ctime — creation time as file attribute.
inode — inode number as file attribute.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my thesis advisers Dr. Jens Weber and Dr. Issa Traore for their continuous support and valuable comments. With their help and encouragement, I was able to put this work together.

Moreover, I wish to thank Dr. Fayez Gebali, the supervisory committee member for his valuable discussions and support.

Furthermore, I would like to thank Dr. Belaid Moa from Compute Canada for his help and support during the implementation phase which enabled me to use their resources. The suggestion and feedback from him was a valuable resource.

Special thanks to my wife, Enas, who was so patient with me during my journey. Also, I want to extend my thanks to my father, mother, sister and brothers.

Many thanks go to my sponsors in Saudi Arabia: Taif University, the Saudi Government and Saudi Arabian Cultural Bureau in Canada.

Finally, thanks to all of those friends and loved ones that I have met here in Victoria.

DEDICATION

To my father, mother, wife, sister and brothers. To all of those who supported me during my journey.

Chapter 1

Introduction

1.1 Motivations

Our daily lives depend, now more than ever, on digital data on many fronts: healthcare, banking, socializing, national and international security, and so on. As such, these digital data need to be protected and, more importantly, to be enabled and ready for digital forensic investigation. Instead of thinking about a forensic investigation only in the aftermath of a breach, one should be proactive and equip the system with the necessary forensic capabilities before any incident. Such capabilities will earn the trust of the customers and ease the tasks of law enforcement and service providers to carry out legal actions and prosecution against the offenders under any circumstances.

With the increase of digital crimes both in number and sophistication, a Proactive Investigation System is becoming a must in Digital Forensics (DF). According to the FBI annual report 2010 [16], the size of data processed during the 2010 fiscal year reached 3,086 TB (compared to 2,334 TB in 2009) and the number of agencies that requested Regional Computer Forensics Laboratory assistance increased from 689 in 2009 to 722 in 2010. Since most investigation tools are reactive in nature (or post-mortem) and can be easily challenged with anti-forensic methods (see Chapter 2), the next-generation digital forensic tools are required to be proactive and distributed [52]. The proactive nature of these tools allows them to better handle anti-forensic attacks by performing collection and preservation before hand. The need for distribution is even more evident when on-site investigation, requiring intensive computational resources, or Proactive Digital Forensic (PDF) analysis, or requiring semi-real time processing, is to be performed. This is also the case for investigating crimes on clouds.

According to Garfinkel [19], the golden age of "reactive" digital forensics has come to an end and is faced with an inevitable crisis due to the advances and fundamental changes in the digital world:

- Digital device storages are diverse and so large that imaging and processing their contents is becoming expensive and time-consuming. In fact, a hard drive with 2 TB may take more than 7 hours to image.
- Embedded flash storages are widespread and challenging to remove and image.
- The proliferation of operating systems and file formats are increasing the complexity of attacks and the cost of developing digital forensic tools.
- The use of multiple devices is making the usual single-device analysis incomplete as the evidence is usually spread across multiple devices.
- The increased use of encryption makes it difficult to process the data even if it is successfully recovered.
- The widespread use of clouds for remote processing and storage renders local investigation useless as the data and the code cannot be found locally but "somewhere" on the cloud.
- The increased number of attacks that use RAM instead of a persistent storage medium, as well as those defeating encryption, requires expensive DF RAM-based tools.

Moreover, the existing digital forensic tools are becoming inadequate as they are reactive and evidence-oriented, and designed to investigate crimes carried out against people using computers [19]. Digital forensics is now in dire need of tools that are proactive and investigation-oriented, that address "computers against computers and/or people" crimes and that scale up with data. Instead of only helping investigators to locate a specific evidence after the harm is done, these tools should be proactive in collecting and preserving the necessary data and evidence and allowing for automated investigation and analysis. In fact, they should be able to perform, among other functions, data exploration, outlier and anomaly detection, and report generation.

1.2 Problem Statement

Every successful digital forensic investigation is supposed to answer the following major questions:

- Did a digital crime happen?
- What happened?
- When did it happen?
- Who committed it?
- How did it happen?
- What damage did it do after it happened?
- Is the evidence provided strong enough to hold up in a court of law?

Given the advanced state of digital crimes and their anti-forensic [18, 50] capabilities, we are interested in providing the necessary framework to automate digital forensic investigation and answer most, if not all, of the above questions. Since analysing the crime after it happens, which is usually called *reactive* forensic analysis, is usually not enough and is limited in providing information about the above questions—especially what happened (i.e., event reconstruction) and when (i.e., timeline)—we propose a *proactive* digital forensic framework which complements the reactive framework. The main questions that we need to address are:

1. What should the proactive digital forensic process look like and how is it related to the reactive digital forensic process?
2. How to theoretically model a proactive DF system and how should it be implemented in practice?
3. To what extent can the proactive system provide an answer to the major questions above?

The first question is addressed in Chapter 2. The second question is tackled in Chapters 3, 4 and 5. The last question is handled in Chapters 3, 4 and 5.

1.3 Contributions

Our main contributions are as follows:

1. Propose a framework for a functional Proactive Digital Forensic system in [60] and [2] (see Chapter 2).

2. Extend in [1] the iterative z algorithm [36, 12] to different elements of DF investigation including events and targets, and take into account the fact that files are weighted differently (see Chapter 4). In fact, old files are usually legitimate and should be weighted more than the new ones. This weighting can be done locally (i.e., inside a directory) or globally (across all directories) and carried out using different probability distributions (see Chapter 5).

3. Parallelize the extended algorithms in [1] using Message Passing Interface (MPI) so that the Reactive Digital Forensic (RDF) and Proactive Digital Forensic (PDF) analysis can be done in parallel and across distributed worker nodes (see Chapter 5).

4. Generalize the extended algorithm to the information-based iterative z algorithm in [1] (see Chapter 4). These algorithms are introduced as a novel approach to express the outlier detection from an information theory perspective. As an example, the attribute and summary functions in the iterative z algorithm are replaced with the information and entropy functions respectively.

5. Introduce a multi-resolution approach in [1] to tackle a large set of DF investigation elements for which the parallel outlier detection algorithms above may take a long time or produce many outliers (see Chapters 4 and 5). Under this approach, the outlier detection algorithms are treated as reduction operators that can be applied as many times as desired.

1.4 Dissertation Outline

The next few chapters of the dissertation will be organized as follows. In Chapter 2, an overview of Proactive and Reactive Digital Forensic Investigation based on a Systematic literature review is presented. Chapter 3 provides the details of the theory behind our proactive digital forensic system. Chapter 4 presents an automation for the analysis phase of the proactive digital forensics. Chapter 5 describes the implementation done and the results obtained. Finally, Chapter 6 presents our conclusions as well as the main future work.

Chapter 2

Overview of Proactive and Reactive Digital Forensic Investigation Processes: A Systematic Literature Review

2.1 Introduction

Computer crimes have increased in frequency and their degree of sophistication has also advanced. An example of such sophistication is the use of anti-forensics methods as in the Zeus Botnet Crimeware toolkit (see Section 5.4) that can sometimes counteract digital forensic investigations through its obfuscation levels. Moreover, volatility and dynamicity of the information flow in such a toolkit require some type of a proactive investigation method or system. The term *anti-forensics* refers to methods that prevent forensic tools, investigations, and investigators from achieving their goals [18, 50]. Two examples of anti-forensic methods are *data overwriting* and *data hiding*. From a digital investigation perspective, anti-forensics can do the following [18, 50]:

- Prevent evidence collection.
- Increase the investigation time.
- Provide misleading evidence that can jeopardize the whole investigation.

- Prevent detection of digital crime.

To investigate crimes that rely on anti-forensic methods, more digital forensic investigation techniques and tools need to be developed, tested, and automated. Such techniques and tools are called proactive forensic processes. Proactive forensics has been suggested in [21, 18, 19, 42]. To date, however, the definition and the process of proactive forensics have not been explicated [21].

In order to develop an operational definition for the proactive forensic process and related phases, we have conducted a Systematic Literature Review (SLR) to analyze and synthesize the results published in the literature concerning digital forensic investigation processes. This SLR has ten steps, described in sections 2.3.1 to 2.5.2, grouped under three main phases: planning, conducting, and documenting the SLR [8]. As result of this SLR, a proactive forensic process has been derived.

The SLR approach was selected for a couple of reasons. Firstly, SLR results are reproducible. Secondly, since all resources (databases) will be queried systematically, there is less chance of missing an important reference.

The rest of the chapter is organized as follows. Section 2.2 outlines the related work and the motivation behind the proactive investigation process. Section 2.3 lays out the plan of the systematic literature review prior to implementation. Section 2.4 describes the implementation of the review and the extraction of the primary studies from the selected resources. Section 2.5 generates the report of the review after synthesizing the data collected in the previous section. Section 2.6 presents the review findings, results, and the proposed process. Section 2.7 provides the summary of this chapter as well as a few suggestions for future direction.

2.2 Related Work and Motivation for the Proactive Investigation Process

Inspecting the literature, only a few papers have proposed a proactive digital forensic investigation process. Some of these papers have mentioned the proactive process explicitly, while in others the process was implicit, but all have emphasized the need for such a process.

In [54], Rowlingson stated that in many organizations, the incident response and crime prevention team already performs some activities of evidence collection proactively. But he added that collecting that evidence and preserving it with a systematic proactive approach is not yet addressed and implemented.

In [18], Garfinkel implicitly suggested that, in order to investigate anti-forensics, organizations need to decide in advance what information to collect and preserve in a forensically sound manner.

In [21], Grobler et al. proposed structuring DF into proactive, active (live) and reactive DF. The authors defined proactive DF as "the DF readiness and the proactive responsible use of DF to demonstrate good governance and enhance governance structures." As such, proactive DF was considered as a set of specific policies and general guidelines on DF as required by an organization. Therefore, the proactive stage in their perspective does not contain an operational process and, hence, cannot be automated. Their general bird's-eye view of proactive DF should be enhanced with a concrete DF protocol and explicit phases as they did for the active and reactive DF. Moreover, doing alive (active) investigation only after the IDS Incident Detection/Alert is triggered is still passive, as the detection component itself need to be forensically sound and be a part of a more general and operational proactive system.

In [19], Garfinkel summarized digital forensics investigation processes that have been published in the literature. In his summary, he stated that it would be unwise to depend upon "audit trails and internal logs" in digital forensic investigation. In addition, he noted that a future digital forensic investigation process will only be possible if future tools and techniques make a proactive effort at evidence collection and preservation.

In [42], Orebaugh emphasized that the quality and availability of the evidence collected in the reactive stage of DF is a passive aspect of the investigation. Conversely, the proactive DF is an active stage involving collecting and preserving potential evidence. In addition, a high-level proactive forensic system was proposed and its ideal components were briefly discussed. As future work, the author suggested that in order to address anti-forensics crimes, methods should be identified to handle proactive evidence collection and forensic investigation.

In summary, previous papers have shown the importance of a proactive digital forensics investigation process. The proposed notion of proactiveness is, however, still insufficient and imprecise, and more work needs to be done. To this end, we will follow a systematic literature review and derive the missing components.

2.3 Planning the Systematic Literature Review (SLR)

The planning stage of the systematic literature review consists of the following steps:

2.3.1 Specify Research Questions

This step defines the goal of the SLR by selecting the research question that has to be answered by the review. The research question is: “What are all the processes in digital forensics investigation?”

Processes include the phases of any digital forensics investigation. According to [43], the six phases of digital forensics investigation are: identification, preservation, collection, examination, analysis, and presentation. The reader can refer to [43] for elaboration of these phases.

2.3.2 Develop Review Protocol

The review protocol is outlined in steps 2.4.1 through 2.5.2 below. These steps show how data for the review is selected and summarized.

2.3.3 Validate Review Protocol

The review protocol was validated by querying the selected databases and looking at the search results. Those results were meaningful and showed the feasibility of the developed protocol.

2.4 Conducting the Systematic Literature Review

The review was conducted by extracting data from the selected sources using the following steps:

2.4.1 Identify Relevant Research Sources

Five well-known database sources were selected as being most relevant to the fields of computer science, software engineering, and computer engineering. The expert engineering librarian at the University of Victoria recommended another indexed database that is considered to contain reliable sources: *Inspec*. Two extra public indexed databases were used for sanity check: *CiteSeer* and *Google Scholar*. The *International Journal of Computer Science and Network Security* (IJCSNS) was located while conducting a sanity check in Google Scholar using "digital forensic investigation process" as keywords.

All of the searches were limited in date from 2001 to 2010.

1. IEEE Xplore: http://ieeexplore.ieee.org/Xplore/dynhome.jsp
2. ACM Digital Library: http://portal.acm.org/dl.cfm
3. Inspec: http://www.engineeringvillage2.org/
4. SpringerLink: http://www.springerlink.com
5. ELSEVIER: http://www.sciencedirect.com
6. IJCSNS: http://ijcsns.org/index.htm
7. CiteSeer: http://citeseerx.ist.psu.edu (*indexed database*)
8. Google Scholar: http://scholar.google.ca (*indexed database*)

The queries used to search the databases above, except for IJCSNS, were as follows:

(Computer OR Digital) AND (Forensic OR Crime) AND (Investigation OR Process OR Framework OR Model OR Analysis OR Examination)

For IEEE Xplore, the basic search screen window was used to search only within title and abstract (metadata, not a full text).

In ACM Digital Library, the basic search screen window was used to search for the queries within the database.

In the case of SpringerLink, the advanced search screen window was used to search within the title and abstract. Furthermore, in SpringerLink the search field for queries could not take all of the queries, so the last two keywords, "Analysis" and "Examination," had to be excluded.

In the case of ELSEVIER, the advanced search screen window was used to search within abstract, title, and keywords.

Running the above queries against the databases gave the following numbers of papers:

- IEEE Xplore: 42 (on Nov 1, 2010)
- ACM Digital Library: 27 (on Nov 3, 2010)
- SpringerLink: 158 (on Nov 3, 2010)
- ELSEVIER: 346 (on Nov 4, 2010)

For IJCSNS, as an exception, the keywords "Digital Forensic Investigation" were used in the search screen window. The search returned this number of papers:

- IJCSNS: 86 (on Nov 24, 2010)

Since using the above queries for Inspec and CiteSeer would result in a considerable number of irrelevant Primary Studies (PS), Control Terms (CT) were used instead. In addition, CT were run against previous databases as well, to be able to capture more relevant PS. The CT recommended by the Inspec database as well as the subject librarian are:

(Computer Crime) OR (Computer Forensics) OR (Forensic Science)

The first two CT (computer crime OR computer forensics) were used to search IEEE Xplore, ACM, SpringerLink, and ELSEVIER. "Forensic Science" was excluded since it returns PS out of the scope of this study. For IEEE Xplore, the advanced search screen window was used in searching the metadata only. In the ACM digital library, the advanced search screen window was used to fetch the database within the keywords field. In SpringerLink, the advanced search screen window was used to search within title and abstract. For ELSEVIER, the advanced search screen window was used to search within the keywords.

In the case of Inspec, using the CT above, the database was searched in three categories. In the first category, all of the CT (including "forensic science") were used with an AND Boolean operator between them in the quick search screen window for searching within CT fields in the database. In the second category, only "computer forensics" was used in the quick search screen window to search within the CT field. In the third category, "forensic science" was used in the quick search screen window to search within CT.

For CiteSeer, the advanced search screen window was used. In addition, since CiteSeer does not have the option to search within CT, it was necessary to search its database using keywords. These keywords were “Computer Crime” OR “Computer Forensics” OR “Digital Forensic”. The search was conducted in two categories. First, an OR operator was used between all the keywords in the abstract field. Second, only the first two keywords were used, with an OR operator between them, in the keywords field.

When the above CT and keywords were run on different dates, the following numbers of papers were returned from the databases listed above:

- IEEE Xplore: 1,053 (on Nov 6, 2010)
- ACM Digital Library: 134 (on Nov 8, 2010)
- SpringerLink: 128 (On Nov 10, 2010)
- ELSEVIER: 69 (on Nov 14, 2010)
- Inspec: 459 (on Nov 5, 2010). The PS were distributed as follows:
  - Category 1: 13
  - Category 2: 290
  - Category 3: 156
- CiteSeer: 162 (on Nov 15, 2010)
  - Category 1: 143
  - Category 2: 19

Finally, the primary studies that were collected from running all the above queries are [21, 47, 10, 4, 33, 63, 62, 24, 5, 28, 31, 46, 57, 6, 67, 55, 58, 32, 56, 45]. Additional primary studies were collected by examining the previous primary studies [43, 9, 14, 53, 17, 30].

2.4.2 Select Primary Studies

Selection Language

Publications in the English language only were selected from the above database resources.

Selection Criteria

Primary studies were selected and irrelevant ones were excluded using three filters. The criteria for those filters are as follows:

- The first filter excludes any papers whose titles bear no relation to the question in Section 2.3.1. According to this filter, the total number of papers is 32.
- The second filter excludes any papers that do not target processes of the digital forensics investigation in their abstract or title. After this filter, the total number of papers is 26.
- The third filter excludes any papers that do not discuss processes of the digital forensics investigation in more detail in their full text. This leaves only the primary studies that need to be included in the systematic review. With this filter, the total number of PS remaining is 20, as follows: [21, 47, 10, 4, 33, 63, 62, 24, 5, 28, 31, 46, 57, 6, 67, 55, 58, 32, 56, 45]. Six additional primary studies were found by investigating the 20 PS. Out of these 26 primary studies only 18 papers dealt with the processes of digital forensics investigation.

2.4.3 Assess Study Quality

The primary studies were assessed according to the following categorizations, starting from the highest level to the lowest:

1. Peer-reviewed journals: Level 5 (Highest)
2. Peer-refereed book chapters: Level 4
3. Peer-reviewed conference papers: Level 3
4. Peer-reviewed workshop papers: Level 2
5. Non-peer refereed papers: Level 1 (Lowest)

Table 2.1 shows the summary of the primary studies genre. Nine of the 18 primary studies were journals; these reveal the maturity of the processes listed in this chapter and its patterns.

Genre	Number of Primary Studies
Peer-reviewed journals	9
Peer-refereed book chapters	1
Peer-reviewed conference papers	7
Peer-reviewed workshop papers	1
Non-peer-refereed papers	0

Table 2.1: Paper genre and the number of primary studies.

2.4.4 Extract Required Data

The processes of digital forensics investigation that were extracted from the total 26 primary studies are grouped in Table 2.2.

2.4.5 Synthesize Data

The processes of digital forensics investigation were mapped to the proposed investigation process in Table 2.3.

2.5 Documenting the Systematic Literature Review

This stage is about generating the systematic literature review report.

2.5.1 Write Review Report

The review report is contained in the current chapter.

2.5.2 Validate Report

The same review protocol was used to validate the systematic literature review twice during execution of the review.

2.6 Research Findings

All the processes of digital forensics investigation, as shown in Table 2.3, share the reactive component, but only one [21] includes the proactive component. (In [21], this proactive component has been named the active component.) The reactive component of all processes was inspired by [43]. Recent papers such as [21, 18, 19, 42] have suggested that there is a need for advancement in the area of proactive forensic systems.

In [21], a multi-component view of digital forensics process is proposed. This process is at a high level and consists of three components: proactive, active, and reactive. The term "proactive" as it is used in [21] deals with the digital forensics readiness of the organization as well as the responsible use of digital forensics tools. The active component, considered a part of the proactive component in the current study, deals with the collection of live evidence in real time while an event or incident is happening. The active component of the investigation is not considered to be a full investigation since it lacks case-specific investigation tools and techniques. The reactive component is the traditional approach to digital forensics investigation. The process proposed in our study is derived from [21], but has only two components, proactive and reactive [60], [2] (see Figure 2.1). Our proposed proactive component encompasses the active component described in [21].

Both our proposed process and the multi-component process share the reactive component. Table 2.3 maps phases of the proposed proactive and reactive digital forensic investigation process to phases of the existing processes.

Description of the two components in the proposed process is as follows:

1. **Proactive Digital Forensic Component** has the ability to proactively collect data, preserve it, detect suspicious events, gather evidence, carry out the analysis and build a case against any questionable activities. In addition, an automated report is generated for later use in the reactive component. The evidence gathered in this component is the proactive evidence that relates to a specific event or incident as it occurs [42]. As opposed to the reactive component, the collection phase in this component comes before preservation since no incident has been identified yet.

   Phases under the proactive component are defined as follows:

   - *Proactive Collection:* automated live collection of predefined data in the order of volatility and priority, and related to a specific requirement of an organization or incident.
   - *Proactive Preservation:* automated preservation, via hashing, of the evidence and the proactively collected data related to the suspicious event.

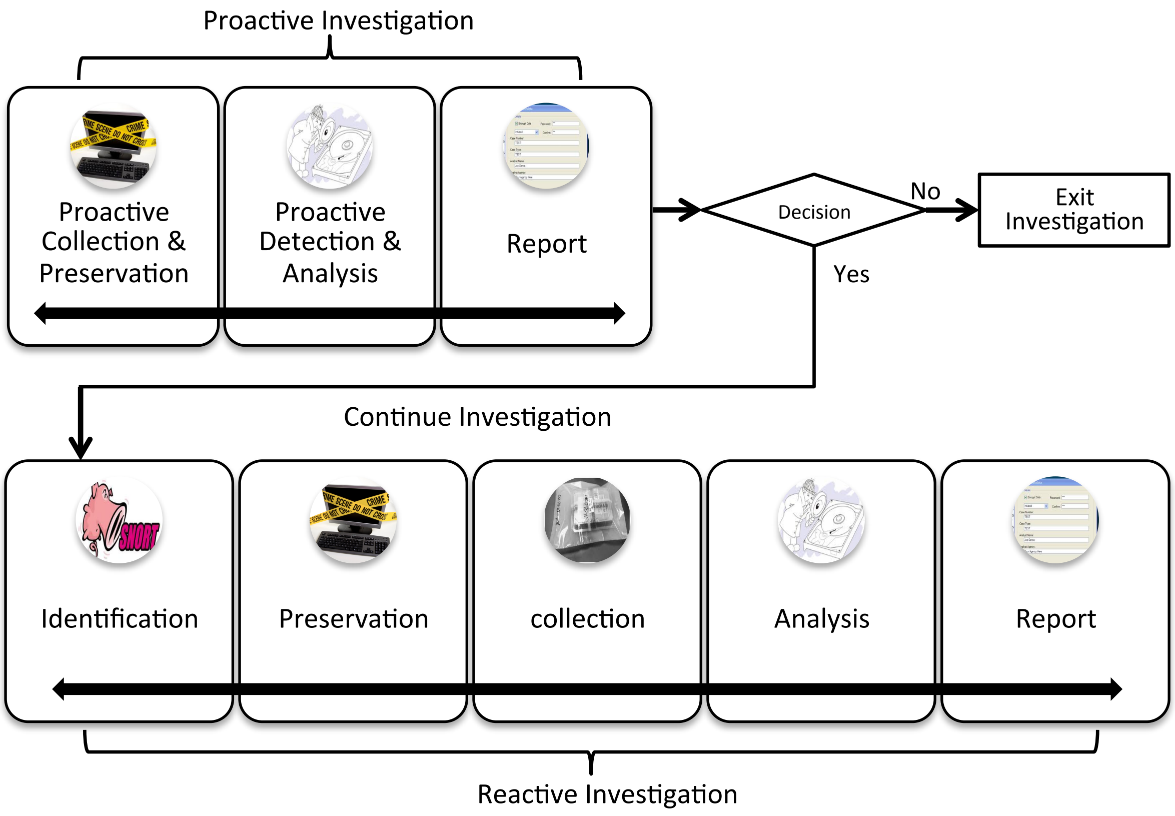


Figure 2.1: Proactive and Reactive Digital Forensic Investigation Framework.

- *Proactive Event Detection:* detection of suspicious event via an intrusion detection system or a crime-prevention alert.
- *Proactive Analysis:* automated live analysis of the evidence, which might use forensics techniques such as data mining and outlier detection to support and construct the initial hypothesis of the incident.
- *Report:* automated report generated from the proactive component analysis. This report is also important for the reactive component and can serve as the starting point of the reactive investigation.

This proactive component differs from common Intrusion Detection Systems (IDS) by ensuring the integrity of evidence and preserving it in a forensically sound manner (maintain the chain of custody to ensure the admissibility of evidence in a court of law [21]). An IDS can be used in a proactive system as its event detection component. In addition, the analysis of the evidence will be done in such a way as to enable prosecution of the suspect and admission to a court of law.

2. **Reactive Digital Forensics Component** is the traditional (or post-mortem) approach of investigating a digital crime after an incident has occurred [43]. This involves identifying, preserving, collecting, analyzing, and generating the final report. Two types of evidence are gathered under this component: active and reactive. Active evidence refers to collecting all live (dynamic) evidence that exists after an incident. An example of such evidence is processes running in memory. The other type, reactive evidence, refers to collecting all the static evidence remaining, such as an image of a hard drive.

   Phases under the reactive component are defined in [43]. It is worth mentioning that the examination and analysis phases in [43] are combined in the proposed process under a single phase called *analysis*.

   In order to see how the two components work together, let us take the scenario that electronic health records with an elevated risk will be proactively collected all the time for any read access of such records. This live collection is automated and is conducted without the involvement of the investigator. When a suspicious event is triggered and detected during collection, all evidence related to that event will be preserved by calculating MD5 hashing function. Thereafter, a forensic image will be made from the preserved evidence, and this image must produce the same MD5 number. Next, a preliminary analysis will be conducted on the forensic image and maybe some data mining and/or outlier detection techniques will be applied to identify whether the event is attributed to a crime and its severity. Finally, an automated report will be generated and given to the person in charge to decide if the reactive component needs to take over or not.

   Next, if needed, the reactive component will conduct a more comprehensive investigation by taking the proactive report as a preliminary evidence for the occurrence of the incident. Since this is a post-mortem of an incident or an event, the evidence will be preserved first by calculating the MD5 hashing function. Then a forensic image will be made from the original source of evidence. This forensic image must produce the same MD5 number to preserve the integrity of the original evidence. Thereafter, a deeper analysis will be conducted using forensic tools and techniques to enable the investigator to find the necessary clues and reach a conclusion. A report will be generated accordingly.

A proactive component should aim at achieving the following goals:

- Develop new proactive tools and techniques to investigate sophisticated digital crimes, including the ones using anti-forensic methods.
- Capture more accurate and reliable evidence in real time while an incident is happening [21, 19, 42].
- Promote automation and minimize user intervention in all proactive phases: collection, preservation, event detection, analysis, and report.
- Provide strong cases and reliable leads for the reactive component.
- Save time and money by reducing the resources needed for an investigation.

As opposed to the multi-component process proposed in [21], our system has the following features:

- It offers a functional proactive process with the above goals.
- It specifies explicitly two functional processes compared to the high-level view of the multi-component framework.
- It can be used to develop techniques and automated tools to investigate anti-forensic attacks [50].
- It automates most if not all the phases of the proactive component.
- It encompasses the active component of [21] in a more reliable component, namely the proactive component.

One of the disadvantages of the proposed process is as follows:

- The investigator will have to decide whether to move from the proactive to the reactive component or to exit the whole investigation. This decision is not automated yet.

2.7 Summary

In order to investigate anti-forensic attacks and to promote automation of the live investigation, a proactive and reactive functional process has been proposed. The proposed process came as result of a SLR of all the processes that exist in the literature. The phases of the proposed proactive and reactive digital forensics investigation process have been mapped to existing investigation processes. The proactive component in the proposed process has been compared to the active component in the multi-component framework. All phases in the proactive component of the new process are meant to be automated. To this end, a theory for the proactive digital forensics is necessary to lay down a strong foundation for the implementation of a reliable proactive system. This is the purpose of the next chapters.

Process No.	Reference No., Genre	Digital Forensic Investigation process name	No. of Phases
1	[43], Conference	Investigative Process for Digital Forensic Science	6 phases
2	[47], Journal	An Abstract Digital Forensics Model	9 phases
3	[9], Journal	An Integrated Digital Investigation Process	17 phases organized into 5 major phases
4	[63], Journal	End-to-End Digital Investigation Process	9 phases
5	[4], Journal	The Enhanced Digital Investigation Process	5 major phases including sub-phases
6	[14], Journal	The Extended Model of Cybercrime Investigations	13 phases
7	[10], Conference	An Event-based Digital Forensic Investigation Framework	5 major phases including sub-phases
8	[24], Journal	The Lifecycle Model	7 phases
9	[5], Journal	The Hierarchical, Objective-based Framework	6 phases
10	[33], Conference	The Investigation Framework	3 phases
11	[30], Journal	The Forensic Process	4 phases
12	[53], Conference	The Computer Forensics Field Triage Process Model	6 major phases including sub-phases
13	[28], Journal	FORZA - Digital Forensics Investigation Framework Incorporating Legal Issues	8 phases
14	[17], Conference	The Common Process Model for Incident Response and Computer Forensics	3 major phases including sub-phases
15	[31], Workshop	Two-Dimensional Evidence Reliability Amplification Process Model	5 major phases including sub-phases
16	[57], Conference	Digital Forensics Investigation Procedure Model	10 phases including sub-phases
17	[6], Book Chapter	An Extended Model for E-Discovery Operations	10 phases
18	[21], Conference	A Multi-component View of Digital Forensics	3 major phases including sub-phases

Table 2.2: Processes of digital forensics investigation.

Digital Forensic Investigation Process Name & Reference No.	Pro. Invest.					Rea. Invest.				
	Proactive Collection	Proactive Preservation	Proactive Event Detection	Proactive Analysis	Report	Identification	Preservation	Collection	Analysis	Report
Investigative Process for Digital Forensic Science [43]						√	√	√	√	√
An Abstract Digital Forensics Model [47]						√	√	√	√	√
An Integrated Digital Investigation Process [9]						√	√	√	√	√
End-to-End Digital Investigation Process [63]								√	√	
The Enhanced Digital Investigation Process [4]						√	√	√	√	√
The Extended Model of Cybercrime Investigations [14]						√	√	√	√	√
An Event-based Digital Forensic Investigation Framework [10]						√	√	√	√	√
The Lifecycle Model [24]						√	√	√	√	√
The Hierarchical, Objective-based Framework [5]						√	√	√	√	√
The Investigation Framework [33]						√	√	√	√	√
The Forensic Process [30]							√	√	√	√
The Computer Forensics Field Triage Process Model [53]						√	√	√	√	
FORZA - Digital Forensics Investigation Framework Incorporating Legal Issues [28]						√	√	√	√	√
The Common Process Model for Incident Response and Computer Forensics [17]						√	√	√	√	√
Two-Dimensional Evidence Reliability Amplification Process Model [31]						√	√	√	√	√
Digital Forensics Investigation Procedure Model [57]						√	√	√	√	√
An Extended Model for E-Discovery Operations [6]						√	√	√	√	√
A Multi-component View of Digital Forensics [21]	√	√	√	√	√	√	√	√	√	√

Table 2.3: Mapping phases of the proposed proactive and reactive digital forensics investigation process to phases of the existing processes.

Chapter 3

Theory for Proactive Digital Forensics

The complexity of digital crimes, in general, and anti-forensic attacks, in particular, requires a well-founded formalism for digital forensic tools. This requirement is even more stringent for proactive systems; as they need to be formally defined, validated and verified, and ready for anti-forensic attacks. In this chapter, we present an intuitive theory for proactive digital forensics and show how it can be used as a novel formalism for implementing proactive digital forensic systems.

3.1 Complexity of Digital Forensic Investigation from the First Principles

As opposed to the usual crimes, digital attacks are so complex that it is hard to investigate them forensically. The elements involved in a digital crime are located in a large multidimensional space and cannot be easily identified. With the increase of storage and memory sizes, and the use of parallelism, virtualization and cloud, the parameters to take into account during an investigation can even become unmanageable.

The complexity of the multidimensional space is an immediate consequence of the fundamental principles of computer forensics, discussed next.

3.1.1 Fundamental Principles of Computer Forensics

Peisert et al. [44] identified five fundamental principles for an ideal computer forensic investigation. These principles are so critical that any tool that does not take them all into account is doomed to fail in providing the full picture of a digital incident [44]. A tool that only follows some but not all of the principles will still fail to identify many scenarios and events or do so incorrectly.

The five fundamental principles are stated below:

Principle 1 Consider the entire system. This includes the user space as well as the entire kernel space, file system, network stack, and other related subsystems.

Principle 2 Assumptions about expected failures, attacks, and attackers should not control what is logged. Trust no user and trust no policy, as we may not know what we want in advance.

Principle 3 Consider the effects of events, not just the actions that caused them, and how those effects may be altered by context and environment.

Principle 4 Context assists in interpreting and understanding the meaning of an event.

Principle 5 Every action and every result must be processed and presented in a way that can be analyzed and understood by a human forensic analyst.

The complexity of the investigation space can immediately be inferred from the five principles: they require considering the whole state of the entire operating system, including user and kernel space events, files, network interfaces, and the rest of subsystems, at all times and in all possible contexts. In addition, the investigators should interpret all the events generated from the system at different levels of abstraction within the environment in which they occur. Moreover, the sequence of events that led to a specific incident needs to be reconstructed from the collected data with a high degree of certainty. On top of all of these considerations, every element of any investigation needs to be analyzed and presented in a more readable form to be ready for deeper investigation.

3.1.2 Fundamental Principles of Proactive Digital Forensics

Based on a few observations and assumptions, Bradford et al. [7] introduced three proactive forensic principles, which are listed and described below.

- The small-security-breach principle: Small attacks should not be ignored, as they might lead to a fatal one.
- The small-user-world principle: Employees usually use a small number of systems and applications and they do so in a manner similar to their peers.
- The incremental violation principle: Internal violators usually go through incremental baby steps and a noticeable learning curve before being proficient in their attacks.

In our point of view, the above principles suffer from the following drawbacks:

- They are, in general, based on assumptions that may not hold. In particular, they are restricted to the insiders of an organization and may fail when an outsider performs the attack. As such, they go against the second principle of the five fundamental principles discussed in Section 3.1.1.
- They are limited and do not capture sophisticated crimes that use advanced techniques as in the anti-forensics realm.
- The entirety of the process of the forensic investigation is not considered, and the principles were geared towards the analysis phase of the process. For example, the preservation phase, which is considered a critical phase of any proactive forensic investigation, is not taken into account.

Therefore, in addition to the five fundamental principles, we need better principles for conducting an automated proactive investigation in real time. They should be general enough, forensically sound and compatible with the five fundamental principles. The following observations are necessary to synthesize the additional principles:

- *Intruders can compromise the system at any time, thus one should expect an attack to happen at any time. One should also expect an attack to tamper with any element including the logs. This implies that the full history of the system is important.*
- *By nature, a proactive component should monitor the system forensically and have the ability to preserve the state of the system for further investigation. More precisely, it should permit the investigator to compare the current state of the system with its previous states and be able to restore the system to a*

good state if the current one is detected to be illegitimate. In addition, it should ensure that the preserved data is protected.

- *A proactive system should detect the crimes in their early stages and reduce the damage they would cause.*
- *A proactive system should implement preventive measures to stop the attack and be able to predict the location of the damage and protect it forensically.*

The first and the last two observations lead to the sixth and seventh principle [1], respectively:

Principle 6 Preserve the entire history of the system.

Principle 7 Perform the analysis and report the results in real time.

By preserving the entire history of the system, we can go back in time and reconstruct what happened and answer reliably all the necessary questions about an event or incident. The reconstructed timeline is based on the actual states of the system before and after the event or incident. In addition and due to the large amount of data, events and actions involved, performing a proactive analysis and reporting require real time techniques that use high-performance computing. The analysis phase should be automated and have the necessary intelligence to investigate the suspicious events in real time and across multiple platforms.

3.1.3 DF Multidimensional Space

In addition to the actions and events that the seven principles listed above emphasize, we introduce the notion of targets. A *target* is any resource or object related to the system under investigation (e.g., a file, memory, register, etc.). We will use an *element of DF investigation* to refer to a target, an action or an event. At a time t and as shown in Figure 3.1, the system is in the process of executing an action that reacts to some targets and events, and produces new targets and events or modifies the existing ones. Therefore to describe the dynamics of the system at a single instant t, one needs to know at least the states of the targets, the events generated and the actions executed at t. For a full description of the dynamics, these elements of investigation need to be specified at every instant of time; and the complete analysis of the dynamics of the system requires a large multidimensional space [1].

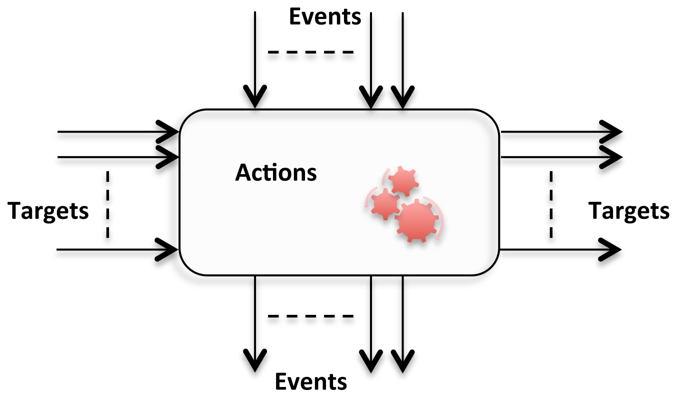


Figure 3.1: Relation between actions, targets and events.

Being proactive implies that when many systems are involved (as is the case in networks or clouds), one has to consider not only one system at a time but the ensemble in a single combined space.

3.2 Modelling the Proactive system

The diversity of the digital systems and the complexity of their spaces require building investigation tools from the ground up. As the size of the investigation space is getting larger, the tools must be able to reduce it in a systematic way without focusing on finding a specific piece of evidence [19]. This investigation-oriented aspect of the forensic tools requires a solid theory to formalize their implementations.

3.2.1 Related Work

Few attempts have been made to formalize digital forensics. Some of these attempts dealt with the analysis phase only, while others were concerned about the general methodology followed during an investigation.

Colored Petri Nets were used in [64] to model past events and the interaction between them. However, it is not general enough [23] and requires preliminary information about the attack.

Stallard et al. [61] used, in a reactive analysis context, expert systems with decision tree-based semantic integrity checking that relies on the principle of invariance in the data redundancies of a system. It requires having some prior information about the good states of the system to be able to investigate complex attacks.

In [20, 29], Gladyshev et al. used Finite State Machine (FSM) to model potential attack from the evidence found. Carrier et al. [13] proposed a framework for digital forensic investigation based on the computer history that also uses FSM. Both approaches, however, are not reliable when exposed to anti-forensic attacks [50]. In addition, Hankins et al. [23] proposed a new model based on a Turing Machine (TM) to reconstruct computer forensic events.

Arasteh et al. [3] presented an approach of analysing log files based on Computational logic and formal automatic verification. Rekhis et al.[48] introduced also a Computational logic (proof-based) approach for digital investigation of security incidents and called it Investigation-based Temporal Logic of Actions (I-TLA). I-TLA is used to prove or disprove the existence of possible attack scenarios that will lead to the evidence observed. The attack scenarios are modelled and generated to emulate how the attack was carried out. In [49], they also developed a theory for network digital forensic analysis to prove or disprove the occurrence of network attacks such as IP spoofing attacks.

Hypothesis testing [70, 71, 7] was used by Willassen to support timestamp investigation for anti-forensic attacks. Hypotheses were formulated about tampered-with timestamps and they were statistically tested using observed evidence.

Both Deterministic Finite Automaton (DFA) and hypothesis testing was used by Carrier [11] to create a mathematical model for digital forensics investigation and deal with event construction based on historical data and hypothesis testing.

Ryan et al. [34] proposed a formal framework for analysing digital crimes and it was used to construct forensic procedures to investigate attacks. It is, however, a signature-based framework, as it is restricted to known attacks.

The common feature of these digital forensic formalisms is that they are mostly dealing with reactive digital forensics or they are limited to a specific phase during the investigation process. As such, they do not assume any prior information about the normal state or any forensic-readiness of the system. Moreover, as Hankins pointed out in [23], they are not general enough to be applied in most real investigations.

3.2.2 A Model for a Proactive System

In our perspective, a proactive digital forensic system is viewed as a *feedback dynamical system*[1] in which the forward system is the system under investigation and the feedback system[2] is the proactive component, as shown in Figure 3.2.

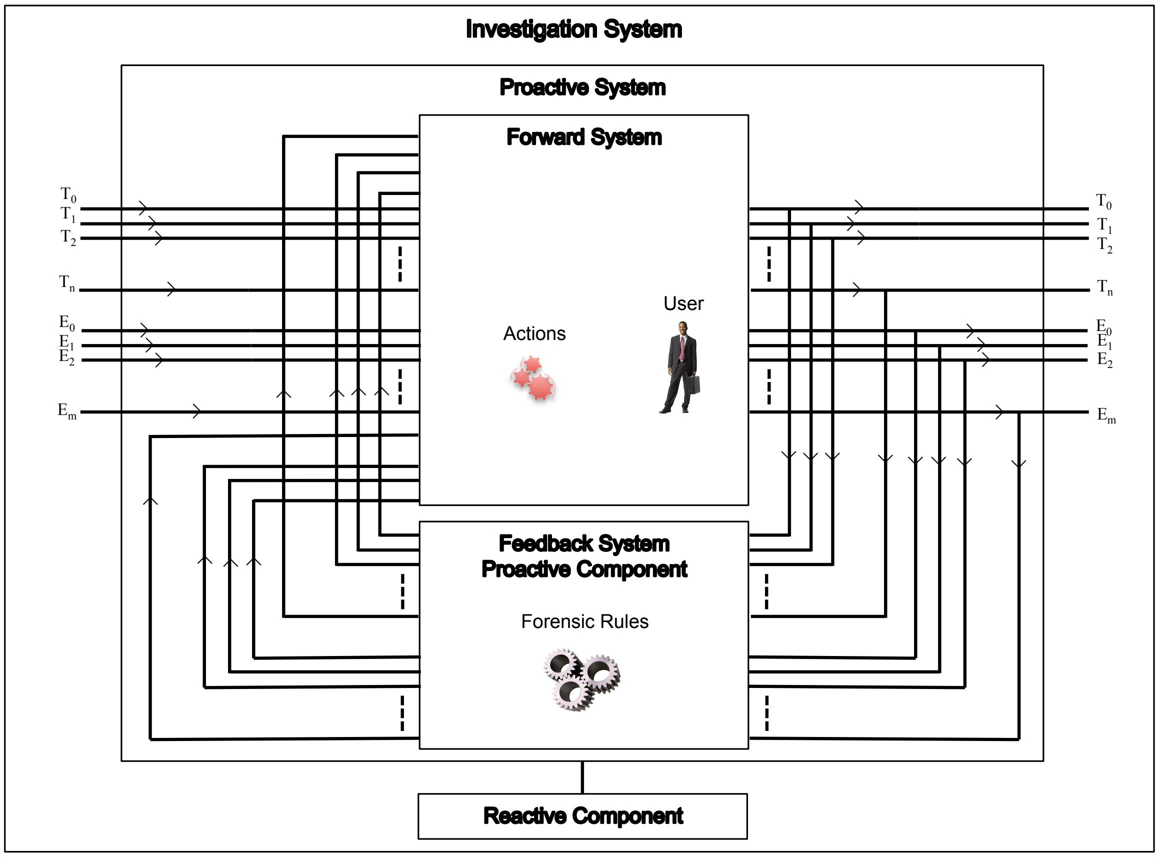


Figure 3.2: Investigation System.

Both systems (the forward and the feedback) can be modelled as a tuple (T, E, A), where T is a set of *target*s, E is a set of *event*s, and A is a set of possible *action*s each of which is viewed as a transfer function of targets and events. To clarify this, each target $f \in T$ is associated with a set $S(f)$ representing the possible states in which it can be. The Cartesian product of $S(f)$ for all targets f defines the state space of

[1]It is an approach to understand the behaviour of complex system over time. Also, it deals with feedback loops to adjust the system accordingly, as is the case in our proactive system.

[2]It is the proactive component that takes the output from the forward system (system under investigation) and takes proactive measures to feed it in again as input to the forward system as an adjustment.

the system's targets and we denote it by $\mathcal{T}$. We do the same for every event e but we consider $S(e)$ to contain two and only two elements, namely $\uparrow$ (triggered event) and $\downarrow$ (not triggered event). The Cartesian product of all the system's events ($S(e)$ for every event e) is denoted by $\mathcal{E}$ (status space). An action a is therefore a function from $\Gamma \times \mathcal{T} \times \mathcal{E}$ to $\mathcal{T} \times \mathcal{E}$, where Γ represents the time dimension. The *evolution function* ψ is defined from $\Gamma \times (\mathcal{T} \times \mathcal{E}) \times A$ to $\mathcal{T} \times \mathcal{E}$ by

$$\psi(t, (\vec{r}, \vec{e}), a) = a(t, \vec{r}, \vec{e})^3.$$

At a time $t \in \Gamma$, we say that an event e is *triggered* if its status at time t is $\uparrow$, and *not triggered* $\downarrow$ otherwise. The notation $\uparrow^t e$ will be used to denote that the event e is triggered at time t. We extend the set of targets and events with two special elements $\emptyset_T$ and $\emptyset_E$ representing empty target and empty event. The $\emptyset_A$ represents the empty action; that is, for every $(\vec{r}, \vec{e}) \in (\mathcal{T} \times \mathcal{E})$, we have

$$\emptyset_A(t, \vec{r}, \vec{e}) = (\vec{r}, \vec{e}).$$

In addition to the tuple (T, E, A) specifying a digital system, the proactive component is specified by an extra tuple $(\mathcal{D}, P, C, R)$, where $\mathcal{D}$ is a set of binary relations on $E \times T$, P is a computable bijection function from T to T, C is a set of logical expressions, and R is a set of rules called *Digital Forensic Rules* (DFRs). If, for each target f in the domain of P (denoted by $P^{-1}(T)$), there is a computable bijection from $S(f)$ to $S(P(f))$, which we denote by P_f, we say that P is a *preservation* function. In this case, the domain of P is called the *collected* targets and the image of P, denoted by $P(T)$, is called the *preserved* targets as shown in Figure 3.3. A DFR is a tuple of the form $(e, c, a, e') \in E \times C \times A \times E$ representing the execution of the action a and triggering of e' when e is triggered and the condition c is true. To ease the interpretation of the tuple, we denote such a DFR by $@e \xrightarrow{c} a, \uparrow e'$. This is the general form of the DFR and, therefore, we qualify it as *generalized.* If the event e' is $\emptyset_E$, then the DFR is a *conditional* DFR and is written as $@e \xrightarrow{c} a$. A *simple* DFR is the one for which c is true all the time and the event e' is $\emptyset_E$. The notation can therefore be simplified to $@e \rightarrow a$. In summary, we distinguish three kinds of forensic rules:

[3]In practice, we take finite targets $\{f_1, \ldots, f_n\}$ and events $\{e_1, \ldots, e_m\}$ and produce state vector targets of the form $\vec{r} = (r_1, r_2, ..., r_n)$, where r_i is a state of the target f_i, and status vector events of the form $\vec{e} = (u_1, u_2, ..., u_m)$, where u_i is a status of the event e_i.

- Simple forensic rule which has the form: $@e \rightarrow a$,
- Conditional forensic rule with the form: $@e \xrightarrow{c} a$,
- Generalized forensic rule, which has the form: $@e \xrightarrow{c} a, \uparrow e'$.

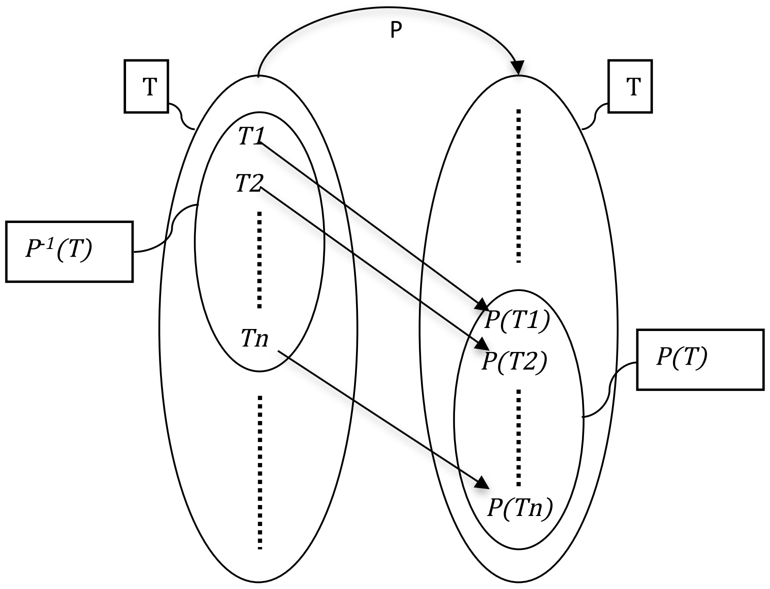


Figure 3.3: Targets Preservation as a Bijection Function.

Events (E) can be associated to targets (T) using a binary relation $D \in \mathcal{D}$ ($D \subseteq E \times T$) , which can be viewed as in Figure 3.4. An example of D would be the binary relation that associates each event to targets that need to be preserved when the event is triggered. Yet another example of D is the relation that associates events to the targets that trigger them. A target T triggers an event e when the change of the T's state causes e to fire.

To illustrate what is really happing in Figure 3.2 as well as the different notions introduced above, we give the following example associated with the botnet called Zeus, as it is the case study we used for the whole proactive system implementation (see Section 5.4). The forward system is the system under the investigation, the computer (operating system — Windows XP machine) susceptible to Zeus attacks. The proactive forensic component that we implemented is the feedback system and is responsible for continuously collecting and preserving important targets and events as well as doing the analysis and generating reports. More specifically, Zeus's important

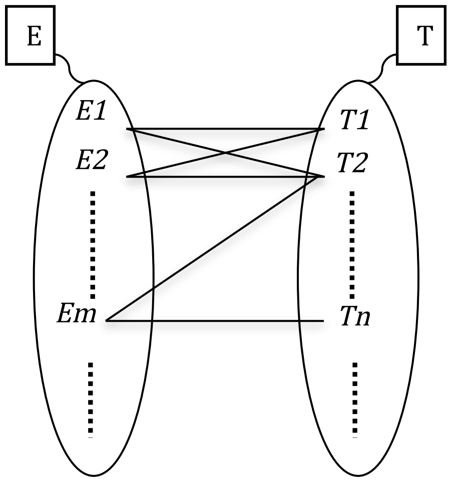


Figure 3.4: Events and Targets binary relation.

targets and events are the system32 folder and its status change. Therefore the target "system32 folder metadata" is collected and preserved at prespecified time intervals. When the system32 folder changes, this event is captured by the proactive component and is used by the forensic rule engine to trigger the right forensic rules, which are responsible for handling the analysis of this incident. In addition to adding extra targets and events to be collected and preserved, these forensic rules may take extra actions that can be sent as feedback to the forward system to do those adjustments and take proactive measures. The feedback system will generate a report (a target) and alerts (events) for the system administrators when it is done analysing and correlating the evidence.

With this we set the stage for the DF analysis and the specification of events and targets as discussed in the next sections. Before we move on, it is worth pointing out that the proactive component phases can be expressed using DFRs. For example, given an event e and a relation $D \in \mathcal{D}$, the preservation phase can be expressed as a forensic rule as follows:

$$@e{\rightarrow}A(P\{De\}),$$

where De is the set of targets associated with e via D and $A(P\{De\})$ is the action of preserving every target in De.

Collecting and preserving all targets as well as analysing the system in the full space of $F = (T, E, t)$ are infeasible as they would require unlimited resources to store the profiles and execute the analysis in real time. Therefore, we need ways to reduce

F to a better subspace requiring reasonable computational resources.

3.3 Theory of Events, Targets and Actions

Keeping track of all events and targets is expensive. To reduce them, we introduce a few classifications using preorder and equivalence relations. To illustrate the idea behind these classifications, imagine a botnet writing into a file. This event will trigger other events including checking the permission on the file, updating the access time of the file, changing the inode and writing the data to the actual disk. The idea behind our formalization is to be able to know which events are important (maximal) and which ones can be ignored. The same thing holds for the targets.

3.3.1 Short Theory on Events

Let e_1 and e_2 be two events in E. We defined the relation $\leq_E$ on E as follows: $e_1 \leq_E e_2$ if and only if ($\iff$) whenever the event e_1 happens at a time t, the event e_2 must also happen at a time t' greater than or equal to t. Formally, this can be expressed as:

$$e_1 \leq_E e_2 \iff (\forall t \uparrow^t e_1 \Rightarrow \exists t' \geq t \uparrow^{t'} e_2)$$

Lemma 3.1. *The relation $\leq_E$ is a preorder relation.*

Proof. To prove the lemma, we need to show that the relation $\leq_E$ is reflexive and transitive. That is what we do next.

1. Reflexivity: it is obvious that $e_1 \leq_E e_1$.

2. Transitivity: suppose that $e_1 \leq_E e_2$ and $e_2 \leq_E e_3$. This means that when e_1 happens at a time t, e_2 must happen at a $t' \geq t$. Because $e_2 \leq_E e_3$, e_3 must happen at $t'' \geq t' \geq t$. Therefore, $e_1 \leq_E e_3$.

$\square$

The events that are less than or equal to an event e are called the *subsequent* events of e. Note that if the subsequent events of any event e in E cannot trigger e, then the relation $\leq_E$ is an order relation.

The preorder $\leq_E$ can be turned into a partial order relation $\leq_{[E]}$ on the equivalence classes $[E]$ of the following equivalence relation

$$e_1 \cong_E e_2 \iff e_1 \leq_E e_2 \wedge e_2 \leq_E e_1$$

Lemma 3.2. *Let $\sim_E$ be the relation defined by*

$$e_1 \sim_E e_2 \iff e_1 \leq_E e_2 \vee e_2 \leq_E e_1$$

The relation $\sim_E$ is a dependency relation. This follows from the fact that $\sim_E$ is reflexive and symmetric. Informally, $e_1 \sim_E e_2$ means that when one of them happens, the other one has to happen. Two events are partially independent if they are not related by $\sim_E$. Two subsets of events are partially independent if any two events from both are partially independent. Based on $\sim_E$, we defined the following relation:

$e_1 \approx_E e_2 \iff e_1 \sim_E e_2$ *or there exist e_3 such that $e_1 \sim_E e_3$ and $e_3 \sim_E e_2$*

Theorem 3.1. *The relation $\approx_E$ is an equivalence relation.*

The proof is as follows.

Proof. The theorem holds iff $\approx_E$ is reflexive, symmetric and transitive. These are shown next.

- Reflexivity: since $e_1 \sim_E e_1$, it follows that $e_1 \approx_E e_1$.
- Symmetry: $e_1 \approx_E e_2$ implies that there exist e_3 such that $e_1 \sim_E e_3$ and $e_3 \sim_E e_2$. Given the symmetry of $\sim_E$, we have $e_2 \approx_E e_1$.
- Transitivity: $e_1 \approx_E e_2$ and $e_2 \approx_E e_3$ implies that $e_1 \sim_E e_2$ and $e_2 \sim_E e_3$. Therefore $e_1 \approx_E e_3$.

□

Two events are *totally independent* if they are not related by $\approx_E$. Two subsets of events are *independent* if any two events from both are independent.

Theorem 3.2. *There exists a family of pairwise totally independent subsets of events $\{E_s^i\}$ such that*

$$E = \bigcup_i E_s^i$$

Proof. The proof follows from the fact that $\approx_E$ is an equivalence relation on E. The family can be taken to be the equivalence classes related to $\approx_E$. □

From a graph theory perspective, The preorder relation $\leq_E$ induces a directed graph on the events (see Figure 3.5); the graph corresponds to the Hasse diagram (directed graph with no loops and arcs implied by the transitivity) of the preorder $\leq_E$. The dependency relation $\sim_E$ transforms the directed graph to an undirected one as in Figure 3.6. The connected components of the undirected graph are transformed into equivalence classes given by the equivalence relation $\approx_E$ in Figure 3.7. The equivalence class relation $\cong_E$ replaces the cycles from the undirected graph with a single vertex as in Figure 3.8. In addition, the family of the equivalence classes associated with $\approx_E$ can be seen in Figure 3.9.

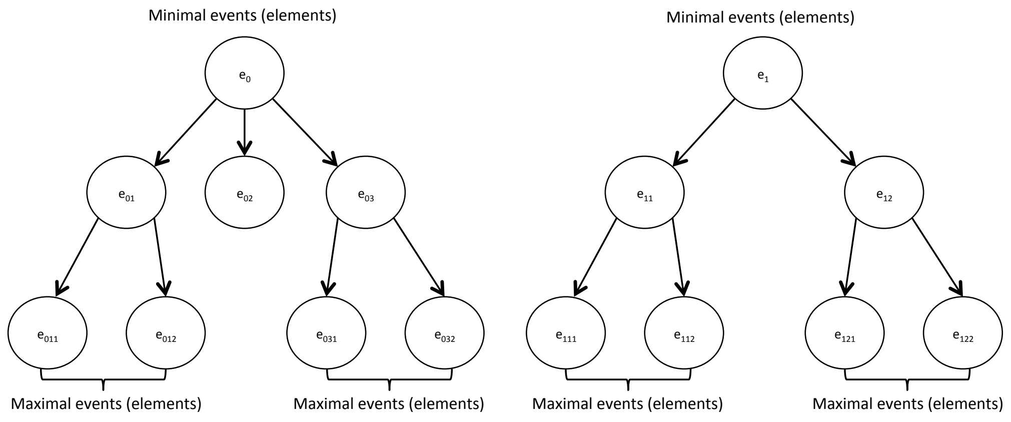

Minimal events (elements)
e0
e01
e02
e03
e011
e012
e031
e032
Maximal events (elements)
Maximal events (elements)
Minimal events (elements)
e1
e11
e12
e111
e112
e121
e122
Maximal events (elements)
Maximal events (elements)

Figure 3.5: Preorder relation on Events (grouping events based on subsequent events).

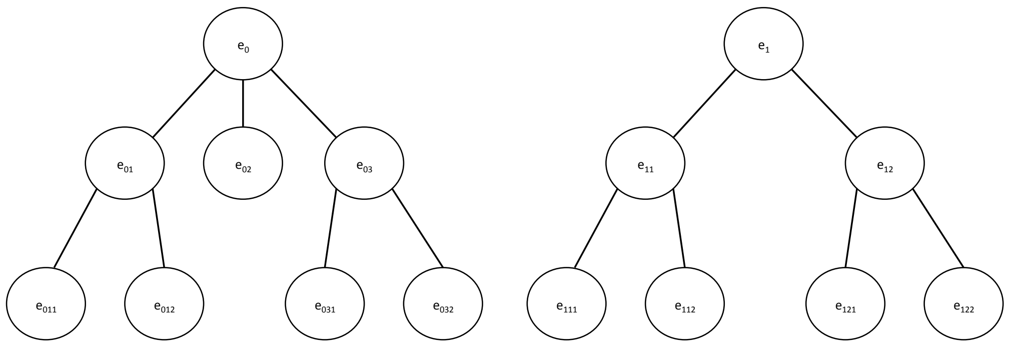


Figure 3.6: Dependency relation on Events (causal relationship between events).

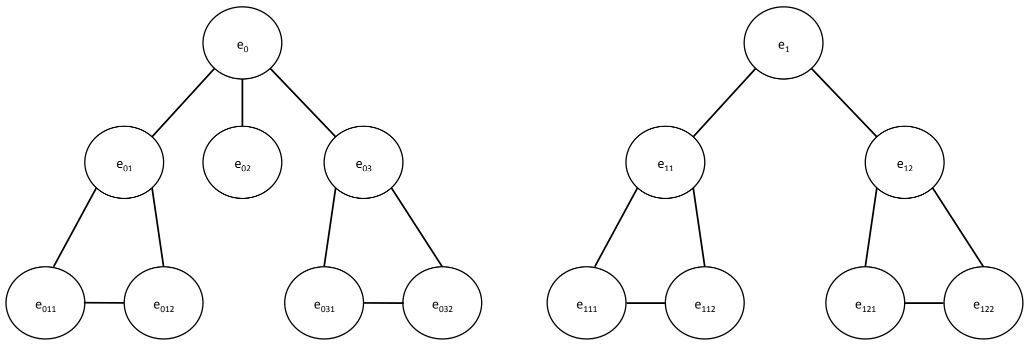


Figure 3.7: Equivalence relation on Events (connect graph).

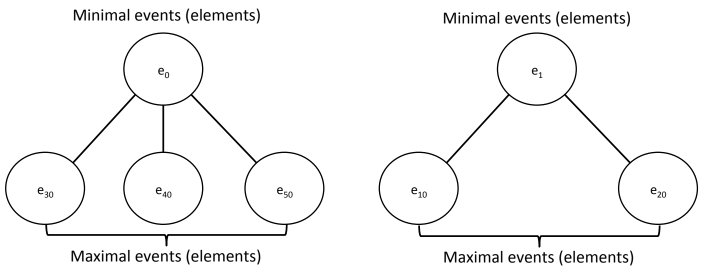

Minimal events (elements)
e0
e30
e40
e50
Maximal events (elements)
Minimal events (elements)
e1
e10
e20
Maximal events (elements)

Figure 3.8: Equivalence class relation on Events (replacing loops with a single vertex).

All the relations defined above can be stated for the equivalence classes $[E]$ given by the relation $\cong_E$. We will replace E by $[E]$ in all the definitions above.

Each equivalence class of $\approx_{[E]}$ is a tree under the relation $\sim_{[E]}$ and a directed tree under $\leq_{[E]}$. Therefore, the set $[E]$ is a forest.

Lemma 3.3. *Since each equivalence class is finite and ordered by $\leq_{[E]}$, each equivalence class of $\approx_{[E]}$ has a maximal and minimal element (see Figure 3.8).*

Definition 3.1. *A* stealth *attack is a digital attack that triggers events that the monitoring system in use cannot notice.*

Theorem 3.3. *A stealth attack that triggers an event in E cannot bypass a monitoring system that monitors an event in every maximal element of each equivalence class of $[E]$, given that the monitor system itself is not attacked. If these events are not watched, then a digital forensic attack can bypass the monitor system.*

The theorem basically states that when an attack triggers an event e in E, it must trigger an event in a maximal element of the equivalence class of $[e]$, where $[e]$ is the

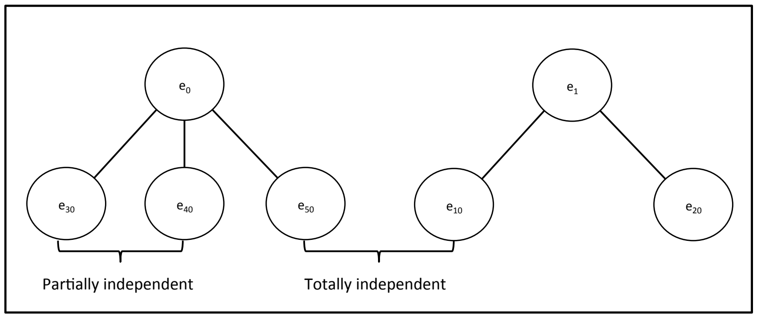


Figure 3.9: Family of pairwise totally independent subset of Events.

equivalence class of e under $\cong_E$. Therefore, if the greatest events are watched, the attack cannot go unnoticed unless the attack modified the monitoring system and caused it not to see events that are maximal elements.

If, however, the monitor system is watching an event that is not the maximal, then the attack can manage to only trigger bigger events than the ones watched by the monitoring system and, therefore, to bypass it.

Note that the theorem does not state that the system will detect that a stealth attack happened when a maximal event is triggered. It only states that the stealth property of the attack does not hold any more.

It is worth pointing out that the classification above is not effective in handling events that trigger other events when specific conditions are true and/or specific actions are carried out. To be precise, we distinguish at least three ways in which the triggering of two events e_1 and e_2 can be related:

1. $\forall t \uparrow^t e_1 \Rightarrow \exists t' \geq t \uparrow^{t'} e_2$
2. $\forall t \uparrow^t e_1 \Rightarrow (\exists t' \geq t$ and $\exists c \in C, c \Rightarrow \uparrow^{t'} e_2)$
3. $\forall t \exists c \in C, c \Rightarrow (\uparrow^t e_1 \Rightarrow \exists t' \geq t$ and $\exists c' \in C, c' \Rightarrow \uparrow^{t'} e_2)$

Once the conditions are added into the picture, the order property of the relation between events does not hold any more. In addition, the conditions, as well as the actions, may depend on time and, as such, the classification of events becomes an undecidable problem. In fact, there is no universal procedure to know when an event is supposed to trigger another.

3.3.2 Short Theory of Targets

Let Ψ be the mapping from T to E (Figure 3.10) that associates each target with its change of status event. The mapping Ψ and $\leq_E$ induces a preorder relation $\leq_T$ defined by

$$T_1 \leq_T T_2 \iff \Psi(T_1) \leq_E \Psi(T_2)$$

Informally, this means that whenever target T_1 changes at time t the target T_2 must change at $t' \geq t$. Similarly, the relations $\cong_E$, $\sim_E$ and $\approx_E$ induce the relations $\cong_T$, $\sim_T$ and $\approx_T$ on targets:

$$T_1 r_T T_2 \iff \Psi(T_1) r_E \Psi(T_2), \text{ where r is } \cong, \sim \text{ or } \approx$$

As we did with events, we can state properties of T and $[T]$ (the equivalence classes given by $\cong_T$). We can also define the partial and the total independence similar to events.

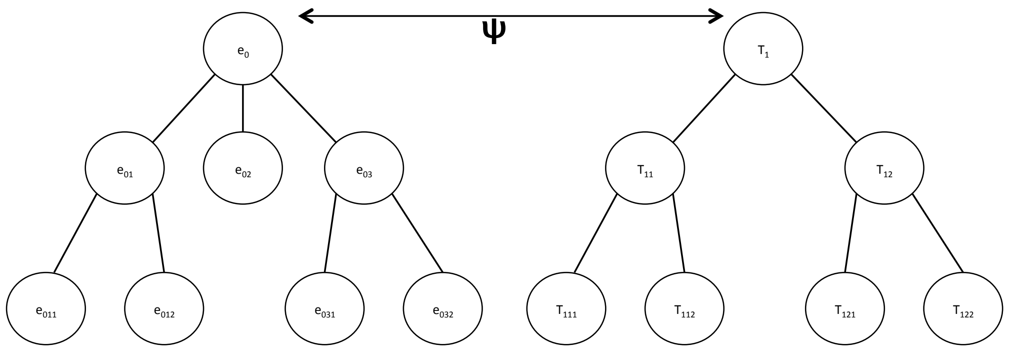


Figure 3.10: Mapping between Targets and Events.

Using the theorem stated above, the maximal targets are enough to detect whether a target has changed. They may not, however, be sufficient to gather all the evidence of an attack. As such, monitoring only the maximal targets may not be enough to gather the necessary evidence. As an example, when a file is written to a disk, it is done in the binary format. Considering only the disk may require extra information about the way the file was written in order to read, interpret and find the evidence in that file.

In what follows, we suppose that each target/event has a set of features that characterize it. These features include priority level, privilege levels, modification time, creation time, access time, etc.

3.3.3 Short Theory on Actions

The set of actions A is extended to $\bar{A}$ using the following operators:

- An associative binary operator called *sequential operator* and denoted by ;. Given two actions a_1 and a_2, the action $a_1; a_2$ is semantically equivalent to carrying out a_1 and then a_2 (the two transfer functions are in series). Note that $\emptyset_A$ is a neutral element of A with respect to ; (i.e., $a; \emptyset_A = \emptyset_A; a = a$ for every action a).

- A commutative binary operator called *parallel operator* and denoted by $||$. In this case $a_1||a_2$ is equivalent to carrying a_1 and a_2 simultaneously (the two transfer functions are in parallel). The action $\emptyset_A$ is also a neutral element of A with respect to $||$.

- A *conditional operator* defined as follows. Given two conditions c_i and c_e in C, and an action a, the operator $c_i a c_e$ represents the action of iteratively carrying out a only when c_i is true and stopping when c_e is false. That is:

$$c_i a c_e = \begin{cases} a; c_i a c_e & \text{if } c_i \text{ is true and } c_e \text{ is false} \\ \emptyset_A & \text{if } c_i \text{ is false} \\ a & \text{if } c_i \text{ is true and } c_e \text{ is true} \end{cases}$$

  If c_e is true, then $c_i a c_e$ is denoted by $c_i a$. If c_i is true, it is denoted by $a c_e$. Note that if both are true, then $c_i a c_e$ is a. Note that $c_i \emptyset_A c_e = \emptyset_A$ for any c_i and c_e in C. As far as we know, the conditional operator does not have an equivalent in dynamical system theory.

The elements of A will be called *primitive* actions and those of $\bar{A}$ *composite* actions.

Since, in practice, the evaluation of a condition and the execution of an action take time, we introduce the *running time* function d from $\Gamma \times (\mathcal{T} \times \mathcal{E}) \times (A \cup C)$ to Γ such that $d(t, x, \emptyset_A) = t$ and $d(t, x, y) \geq t$. The evolution function ψ satisfies the following formula for the sequential operator:

$$\psi(t, (\vec{r}, \vec{e}), a_1; a_2) = \psi(t + d(t, (\vec{r}, \vec{e}), a_1), a_1(t, \vec{r}, \vec{e}), a_2)$$

For the conditional operator, it should satisfy:

$$\psi(t,(\vec{r},\vec{e}),c_iac_e) = \begin{cases} \psi(t+t_{c_i}+t_a+t_{c_e}, a(t+t_{c_i},\vec{r},\vec{e}), c_iac_e), \text{ where } t_{c_i} = d(t,(\vec{r},\vec{e}),c_i), \\ \quad t_a = d(t+t_{c_i},(\vec{r},\vec{e}),a), t_{c_e} = d(t+t_{c_i}+t_a,(\vec{r},\vec{e}),c_e) \\ \quad \text{if } c_i \text{ is true and } c_e \text{ is false} \\ (\vec{r},\vec{e}) \quad \text{if } c_i \text{ is false} \\ a(t+d(t,(\vec{r},\vec{e}),c_i),\vec{r},\vec{e}) \quad \text{if } c_i \text{ is true and } c_e \text{ is true} \end{cases}$$

Given an action $a \in \bar{A}$ and an event $e \in E$, we say that a triggers e if e changes its status for some input event status and target state of a at some instant t. We denote this by $a \rightarrow \uparrow e$. We say that an action a changes a target f if the state of f changes for some input event status and target state and at some time t. Given an action a, we denote by $T(a)$ and $E(a)$ the set of targets and events changed and triggered by a respectively:

$$T(a) = \{f | f \text{ is a target that } a \text{ changes}\} \text{ and } E(a) = \{e | e \text{ is an event that } a \text{ triggers}\}$$

Definition 3.2. *The action support and the target support of an action a, denoted by $\sup(a)$ and $T_{\sup}(a)$ respectively, are defined recursively as follows:*

- *If a is primitive, then $\sup(a) = \{a\}$ and $T_{\sup}(a) = T(a)$.*
- *If a is of the form $a_1; a_2$ or $a_1 || a_2$, then $\sup(a) = \sup(a_1) \cup \sup(a_2)$ and $T_{\sup}(a) = T_{\sup}(a_1) \cup T_{\sup}(a_2)$.*
- *If a is of the form $c_1a'c_2$, then $\sup(a) = \sup(a')$ and $T_{\sup}(a) = T_{\sup}(a')$.*

Since the conditions may mask an action, we have the following lemma.

Lemma 3.4. *Given an action a in $\bar{A}$, we have*

$$T(a) \subseteq T_{\sup}(a).$$

Definition 3.3. *We assume that the state space of targets is partitioned into two regions: legitimate $\mathcal{L}$ and non-legitimate $\mathcal{L}^c$. An attack is an action k that causes the states of the targets to be in $\mathcal{L}^c$ at some instant of time. The target evidence of k is defined as $T(k)$ and its event evidence is $E(k)$.*

Proposition 3.1. *Let k be an attack and let $T_{\text{sup}}(k)$ be its target support. Let M be the set of targets, denoted by M, that the monitoring system is observing. If $M \cap T_{\text{sup}}(k) = \emptyset$, then the monitoring system will not detect any target evidence of k.*

Since $T(k) \subseteq T_{\text{sup}}(k)$, we have $M \cap T(k) = \emptyset$. As such the targets affected (changed) by the attack are not part of M.

In Definition 3.3, we assumed that an action can lead to either a legitimate region or not. And to be proactive, we are interested in computationally deciding in advance which region should avoid the damage. Unfortunately, such an aim is not possible as stated in the following theorem.

Theorem 3.4. *In general, there is no universal computational procedure to detect attacks.*

The proof follows from the following facts:

- The model used to represent the proactive system is general enough to include all possible Turing Machines (TM). In fact, every TM, including the universal ones, can be represented using the tuple (T, E, A), where T contains the tape of the machine, its states, current state register, the head position and its direction; E contains all the events corresponding to the changes of these targets; and A contains the actions corresponding to the transition function of the machine TM according to its targets and events.

- Deciding where an action is legitimate implies deciding whether a Turing Machine causes its tape to have a specific state. Since this is not always decidable, there is no universal computational procedure to generally decide that actions are legitimate. In particular, if we take non-legitimate to be the region characterizing viruses, deciding whether an action is an attack is equivalent to deciding it is a virus. This is, of course, undecidable [27, 39, 41].

Based on the same reasoning as above, detecting whether an action will modify the state of a target to a specific one or trigger a specific event are all undecidable in general. In fact, if the final state of a TM is taken to be the specific state we are interested in, then the decidability is exactly that of the halting problem [39, 41]. As such we have the following theorem.

Theorem 3.5 (Target and Event Undecidability)**.** *In general, there is no universal computational procedure that decides whether an action will trigger a specific event or modify a target to a specific state.*

It is worth emphasizing that these two theorems strongly support the seven principles we discussed previously and suggest subdividing the state space not only into two crisp regions, as done above, but into many regions as described next.

3.3.4 Zone-Based Classification and Forensic Space Reduction

To address the limitation of the classification described previously and address the undecidability issue above, we classify the event and target state into a set of priority zones. These zones can be represented with different colors: green, yellow, and red; starting from a lower priority to a higher one as shown in Figure 3.11. The different sequence of symbols ⋆ (profile 1), ∘ (profile 2), and ■ (profile 3) shown in the figure specify different profiles of the system affecting different targets and events as it evolves in time. When important events/targets with high-priority levels are triggered, a more thorough analysis is expected. Moreover, the zones can be used as a quantifying matrix that provides numbers reflecting the certainty level for the occurrence of an incident. In our case, this number is an important piece of information in the final report.

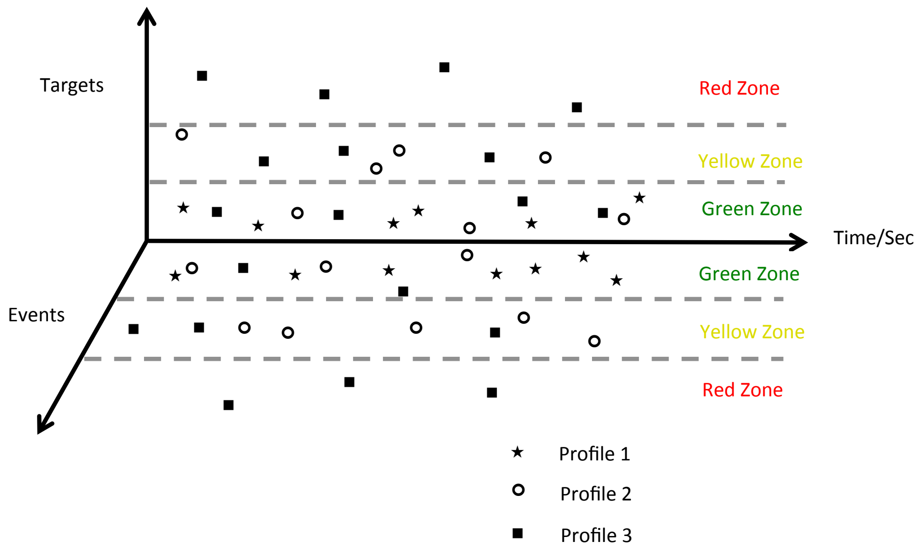


Figure 3.11: Classifications of Events and Targets according to their priority levels.

In addition, if the events/targets are within the high-priority zone, then certain proactive actions can be taken, such as providing a fake identity in the case of the

Zeus botnet, which is discussed in Section 5.4. Those proactive actions can come from a tabular forensic rule engine that has both targets and events listed according to their priority zones (see Table 3.1).

For example, in the case of Zeus, described in Section 5.4, the high-priority events can involve one of the following: IDS, Antivirus, Firewall off and changing the windows system32 folder. On the other hand, the high-priority targets are the system32 folder, registry, network traffic and memory dump.

	Events		
Targets	Green Zone	Yellow Zone	Red Zone
Green Zone	R11	R12	R13
Yellow Zone	R21	R22	R23
Red Zone	R31	R32	R33

Table 3.1: Tabular forensic rule engine.

Given that the number of targets and events are large, this classification is not enough, especially during the analysis phase. As such, we need to reduce the forensic space. Similar to the principal component analysis technique [59], we suggest restricting the analysis to "important" targets and events based on a specific organization policy. This can be seen as projecting the full forensic space F onto a sub-space F' in which the evidence is most probably located. In the case of the Zeus botnet attack (please refer to Section 5.4), if the projection is done on the Zeus's targets and events, then the analysis and the detection of the Zeus attack can be easily done.

As the reader may have guessed, the full space is used to state our theoretical facts while the restricted (projected) spaces will be used to implement the theoretical framework.

3.4 Towards Universal Analysis of Forensic Crimes

Based on the previous sections, we propose analyzing the "digital forensic scene" based on the following observation. The usual non-digital forensic crimes occur in a space such as a room or a subway, during a time interval and using objects such as a knife and a key. By analogy, digital forensic crimes can be analyzed as a profile in a space F. As Section 3.2.2 suggests, this space is given by $\mathcal{T} \times \mathcal{E} \times \Gamma$ (space dictated by targets, events and time); see Figure 3.11.

Definition 3.4. *The* profile *of a system is the evolution of all target and event states in time.*

The profile can visually be viewed as a sequence of points (symbols) in the digital forensic space $F = \mathcal{T} \times \mathcal{E} \times \Gamma$ (see Figure 3.11). Armed with this information, the activity of analyzing a crime can be done in one of the following ways (as done in Intrusion Detection Systems):

- Signature-based proactive analysis: each digital forensic incident that occurred has a special profile in the F space similar to a serial killer signature/profile. During signature-based analysis, we try to correlate the profile of the system gathered so far with the existing signatures to determine any similarities. As the actual profile is being built, we can use it to find the most similar known digital forensic attack scenario. Based on that guidance, we can predict the next move in F and take the necessary proactive actions accordingly. As an example, imagine that the profile gathered so far is similar to that of the Zeus botnet. Imagine also that, following the Zeus profile, the Zeus's next move is to copy our bank information from a fake website page. As a proactive action and given that the profile of the system so far is similar to the Zeus profile, it makes sense to provide fake information and see whether that information is transferred to the wrong location, named the control master server.

  Although this proactive analysis is prone to unknown digital forensic crimes, it provides:

  - The signature of the crime (since we are building the profile of the system as we go and we are adding it to the knowledge base of known digital forensic crimes). In other words, we are building the timeline of the crime as it occurs as opposed to the reactive systems that try to build it after the crime has occurred. This is an effective way of handling crimes that try to hide or erase their traces/history as is the case for anti-forensics methods.
  - The evidence of the crime (since we are keeping track of the necessary records).

- Anomaly-based proactive analysis: our objective here is to analyze the behaviour of the system and see whether it follows its normal profile. If we notice that the system is not behaving as usual, we can take proactive measures as needed. Our focus in this thesis is based on this type of analyzing a digital crime since we do not know in advance what to look for proactively. A new metric based on outlier analysis [12] will be extended and a novel information theory approach will be introduced.

- Protocol-based proactive analysis: in the case, the system profile must conform to a certain protocol [69, 40]. If this is breached, then proactive actions must be taken.

All three analyses can be expressed in the same framework as follows. As the system S proceeds in time, we build a profile for it in the space F. This profile can be denoted by $P(S, t_c)$, where t_c is the current time. The profile is incomplete in the sense that it is only known for time t in the interval $[t_i, t_c]$, where t_i is the initial time. We consider a family of profiles $P_\alpha(S)$ for the system, where α is an index in any set. In this context, analyzing the system means to correlate the profile of the system as specified by $P(S, t_c)$, and the family of profiles given by $P_\alpha(S)$. Based on the kind of the analysis intended, the family $P_\alpha(S)$ can be specified as follows.

- Signature-based proactive analysis: in this case $P_\alpha(S)$ corresponds to the signatures of different digital forensic crimes.

- Anomaly-based proactive analysis: $P_\alpha(S)$ corresponds to the normal profiles of the system.

- Protocol-based proactive analysis: $P_\alpha(S)$ is the family of profiles that matches the protocol.

Finding the similarities between two profiles boils down to computing the correlation between the two. Let $\mathrm{Bu}(E_i)$ be the profile of the system up to time t, where t is the time where the event E_i is triggered. Let $\mathrm{Ex}(E_i)$ be the set of "interesting" profiles with respect to the event E_i, and $C(h, g)$ be the correlation between the two profiles h and g. Computing the correlation between profiles is an action that could be carried out each time a new event is triggered. As such, we see the advantage of using DFRs. In fact, the full analysis in all three kinds of proactive analysis can be modelled in a general single framework and expressed easily using the following DFRs:

1. $@E_i{\rightarrow}P(S,t) = \mathrm{Bu}(E_i)$: This rule builds the profile of the system until time t, where t is the time where the event E_i happened.

2. $@E_i{\rightarrow}\{P_\alpha(S)\} = \mathrm{Ex}(E_i)$: This rule extracts a set of "interesting" profiles.

3. $@E_i{\rightarrow}\{C_\alpha\} = C(P(S,t), \{P_\alpha(S)\})$: This rule computes the correlation between the profiles. This rule should be executed for each α and as soon as $P_\alpha(S)$ and $P(S,t)$ are available.

In the previous section, we introduced the idea of reduced forensic space to handle the complexity of the digital crimes. Instead of carrying out the analysis of profiles in the full space $F = \mathcal{T} \times \mathcal{E} \times \Gamma$ associated with the full targets T, events E and time Γ, we use a sub-space $F' = \mathcal{T}' \times \mathcal{E}' \times \Gamma'$ associated with targets in $T' \subseteq T$, events $E' \subseteq E$ and time $\Gamma' \subseteq \Gamma$. For this, we introduce the projection function $\pi_{(T',E',\Gamma')}$ that takes a profile $\mathcal{P}_{(T,E,\Gamma)}$ in the space F and projects it onto F' to produce a reduced profile $\mathcal{P}_{(T',E',\Gamma')}$. If we denote the set of profiles in F and F' by $\mathbb{P}_{(T,E,\Gamma)}$ and $\mathbb{P}_{(T',E',\Gamma')}$ respectively, we have

$$\pi_{(T',E',\Gamma')} : \; \mathbb{P}_{(T,E,\Gamma)} \longrightarrow \mathbb{P}_{(T',E',\Gamma')}$$

$$\mathcal{P}_{(T,E,\Gamma)} \longmapsto \mathcal{P}_{(T',E',\Gamma')} \text{ such that}$$

$$\forall(\vec{r'}, \vec{e'}, t) \in \mathcal{T}' \times \mathcal{E}' \times \Gamma', (\vec{r'}, \vec{e'}, t) \in \mathcal{P}_{(T',E',\Gamma')} \Rightarrow \exists(\vec{r}, \vec{e}) \in \mathcal{T} \times \mathcal{E} \text{ such that}$$

$(\vec{r}, \vec{e}, t) \in \mathcal{P}_{(T,E,\Gamma)}$, and $\vec{r'}$ and $\vec{e'}$ are the projections of $\vec{r}$ and $\vec{e}$ onto $\mathcal{T}'$ and $\mathcal{E}'$, respectively.

The analysis framework introduced above can take advantage of this concept and use the reduced space F' instead of F to efficiently process the profiles. Since our proactive system is modelling a feedback system, it is natural for the reduced space F' to not be enough to discern legitimate and non-legitimate attacks. Therefore, the subspace on which the projection is done should be dynamic in the sense that it should be adjusted as the analysis proceeds in time.

Chapter 4

Towards Automated Analysis for PDF

Forensic analysis is the most demanding phase in both reactive and proactive digital forensics. Investigators usually carry out this step manually as there is no automated tool to do so. In this chapter, following our previous ideas on universal analysis (see Section 3.4) for digital forensics, we propose a framework to execute and automate the analysis step [1]. The framework makes use of two important concepts: information theory and outlier detection [12] techniques, which we introduce next.

4.1 Information Theory Background

The following definitions in information theory are borrowed from [22]. The entropy (self-information) $H(X)$ of a discrete random variable X is defined in $[H(X) = -\sum_x p(x) \log p(x)]$. This definition evaluates the uncertainty of X. The value of $H(X)$ decreases as the certainty about the event represented by X increases.

The Conditional entropy $H(Y|X)$ is defined by

$$H(Y|X) = -\sum_y \sum_x p(x,y) \log p(x|y),$$

This will give us the amount of information uncertainty about X when Y is known.

Mutual information is given by

$$I(X;Y) = \sum_x \sum_y p(x,y) \log \left(\frac{p(x,y)}{p(x)\,p(y)} \right),$$

This shows the amount of reduction of uncertainty in X if Y is known. Moreover, it gives us the amount of information shared between X and Y.

The main idea behind introducing information theory in our work is to quantify the uncertainty of a profile being legitimate based on the nearby profiles and/or on other elements of the profile itself. For example, if we consider the files in a directory d and their sizes for the analysis, the uncertainty associated with a file f can be taken to be the information of f being legitimate based on the sizes of the files in d. The entropy is therefore the average of the information across all the files.

4.2 Overview on Outlier Detection

To handle the large number of elements of investigation and to perform the PDF analysis in real time, as the seven principles dictate [1], one has to automate the analysis. A promising technique for such an automation is the spatial outlier detection introduced in [12]. It is an anomaly-based technique that detects suspicious elements by examining one or many of its attributes using the iterative z algorithm [36]. This algorithm finds the outlier in a set of spatial features, referred to as *points*, by calculating the differences between their attributes and a *summary function* of their neighbours. These differences are then normalized with respect to their mean and standard deviation. When the point with the maximum normalized difference is greater than a prefixed threshold, it is marked as an outlier and its attribute is taken to be the summary function of its neighbours. The process is then repeated until no outliers are found. A flowchart of the algorithm is shown in Figure 4.1 and the algorithm itself is presented in Algorithm 1 [12].

4.3 Outlier Detection for Automated DF Analysis

Carrier et al. [12] adopted the iterative z algorithm to automate the DF analysis. Algorithm 1 shows how it is expressed when the elements of DF investigation are regular files and the attribute functions are the sizes of the file. In this algorithm, a regular file is denoted by ϕ_i, its size by $|\phi_i|$ and its directory by $D(\phi_i)$. The set of all files is denoted by $\mathcal{N}$ and its cardinality by $|\mathcal{N}|$.

The outlier detection algorithm considers all points to have the same importance and, as such, the summary function is the arithmetic mean of the attribute functions. In practice, however, some attribute functions may be weighted more than others. In

Algorithm 1 The iterative z algorithm using files as the elements of DF investigation and `size` as the attribute function.

```
1: Let $\theta$ be a threshold
2: Let $n$ be $|\mathcal{N}|$
3: outlier=true
4: $\sigma = 0; \mu = 0$
5: **for** $i = 1 \to n$ **do**
6:     Let $N(\phi_i)$ be the set of files in $D(\phi_i)$ and $|N(\phi_i)|$ its cardinality
7:     Set the *attribute function* $f(\phi_i)$ to be $|\phi_i|$
8:     Compute the *summary function* $g(\phi_i) = \frac{1}{|N(\phi_i)|} \sum_{\phi \in N(\phi_i)} f(\phi)$
9:     Compute the *comparison function* $h(\phi_i) = f(\phi_i) - g(\phi_i)$
10:    $\mu = \mu + h(\phi_i)$
11:    $\sigma = \sigma + h(\phi_i)^2$
12: **end for**
13: $\sigma = \sqrt{\frac{\sigma}{n} - (\frac{\mu}{n})^2}$                    ▷ The standard deviation
14: **while** outlier==true **do**
15:     outlier=false
16:     $\phi_q = \arg\max_\phi |\frac{h(\phi)-\mu}{\sigma}|$
17:     **if** $|\frac{h(\phi_q)-\mu}{\sigma}| \geq \theta$ **then**                ▷ $\phi_q$ is an outlier
18:         Mark $\phi_q$ as an outlier
19:         $f(\phi_q) = g(\phi_q)$
20:         Update $g(\phi)$ and $h(\phi)$ for every $\phi$ in $N(\phi_q)$
21:         Update $\mu$ and $\sigma$
22:         outlier=true
23:     **end if**
24: **end while**
```

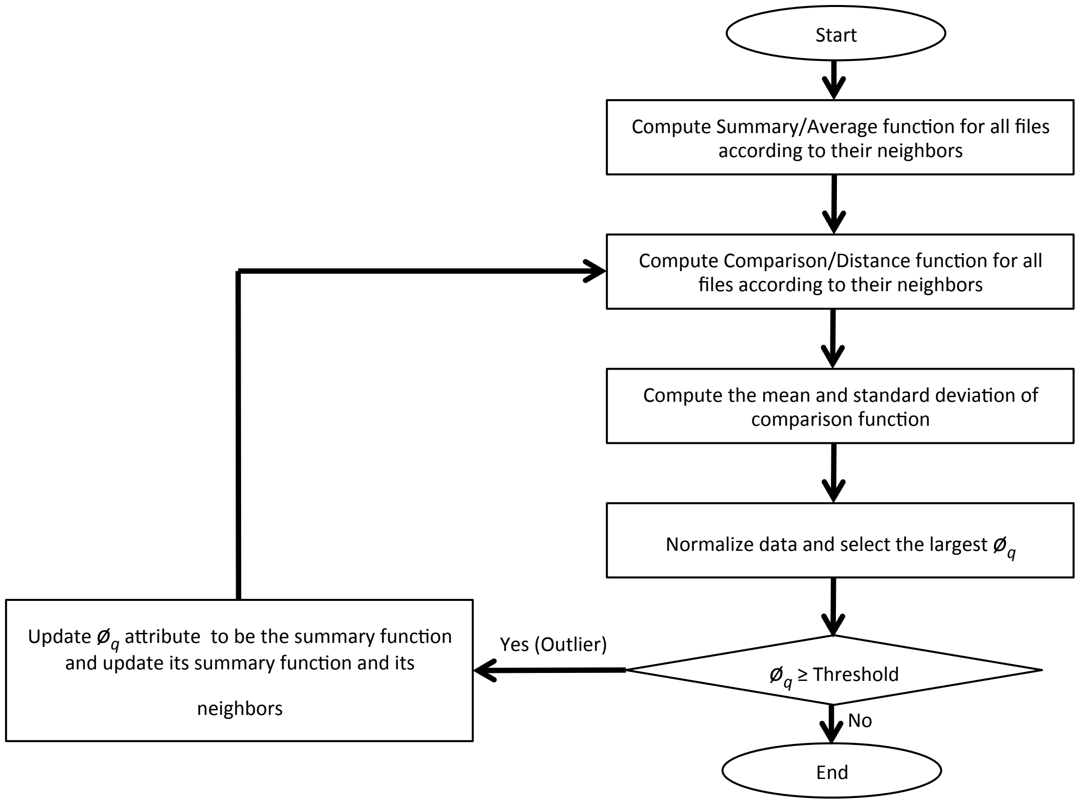

Start
Compute Summary/Average function for all files according to their neighbors
Compute Comparison/Distance function for all files according to their neighbors
Compute the mean and standard deviation of comparison function
Normalize data and select the largest $\emptyset_q$
$\emptyset_q$ ≥ Threshold
Yes (Outlier)
Update $\emptyset_q$ attribute to be the summary function and update its summary function and its neighbors
No
End

Figure 4.1: Flowchart of Iterative z Algorithm.

fact, when a system becomes unstable or malfunctions, the first step towards fixing it is to check the latest changes that were made to it. In the case of files, this translates to weighting old files more than the new ones.

As such, we introduce two kind of probability distributions: *local* and *global competitive* distributions. The latter gives the relative weights of a file with respect to all the files and the former gives its relative weights with respect to its neighbours. Algorithm 2 [1] is the new algorithm and the flowchart of the algorithm is in Figure 4.2.

If the local and the global competitive probabilities are `uniform` (i.e., $p^l(\phi) = 1/|N(\phi_i)|$ and $p^g(\phi_i) = 1/|\mathcal{N}|$), then we obtain Algorithm 1 [12]. But if, for example, old accessed files weigh more than the new ones, then non-uniform local and the global competitive probabilities should be used as illustrated below.

Let $t_a(\phi)$ be the `atime` (access time) of a file ϕ and let t_{aref} be a reference time, which we usually take to be the latest `atime` of all the files (global `atime`) or of the files in the directory $D(\phi)$ (local `atime`). The local and the global competitive probabilities can be then taken to be

Algorithm 2 The probabilistic iterative z algorithm.

```
1: Let $\theta$ be a threshold
2: Let $n$ be $|\mathcal{N}|$
3: outlier=true
4: $\sigma = 0; \mu = 0$
5: for $i = 1 \to n$ do
6:     Let $N(\phi_i)$ be the set of files in $D(\phi_i)$ and $|N(\phi_i)|$ its cardinality
7:     Let $p^l_{N(\phi_i)}(\phi_i)$ be the local competitive probability of $\phi_i$
8:     Let $p^g(\phi_i)$ be the global competitive probability of $\phi_i$
9:     Set the attribute function $f(\phi_i)$ to be $|\phi_i|$
10:    Compute the summary function $g(\phi_i) = \sum_{\phi \in N(\phi_i)} p^l_{N(\phi_i)}(\phi) f(\phi)$
11:    Compute the comparison function $h(\phi_i) = f(\phi_i) - g(\phi_i)$
12:    $\mu = \mu + p^g(\phi_i) h(\phi_i)$
13:    $\sigma = \sigma + p^g(\phi_i) h(\phi_i)^2$
14: end for
15: $\sigma = \sqrt{\sigma - \mu^2}$                                ▷ The standard deviation
16: while outlier==true do
17:    outlier=false
18:    $\phi_q = \arg\max_\phi |\frac{h(\phi)-\mu}{\sigma}|$
19:    if $|\frac{h(\phi_q)-\mu}{\sigma}| \geq \theta$ then                    ▷ $\phi_q$ is an outlier
20:        Mark $\phi_q$ as an outlier
21:        $f(\phi_q) = g(\phi_q)$
22:        Update $g(\phi)$ and $h(\phi)$ for every $\phi$ in $N(\phi_q)$
23:        Update $\mu$ and $\sigma$
24:        outlier=true
25:    end if
26: end while
```

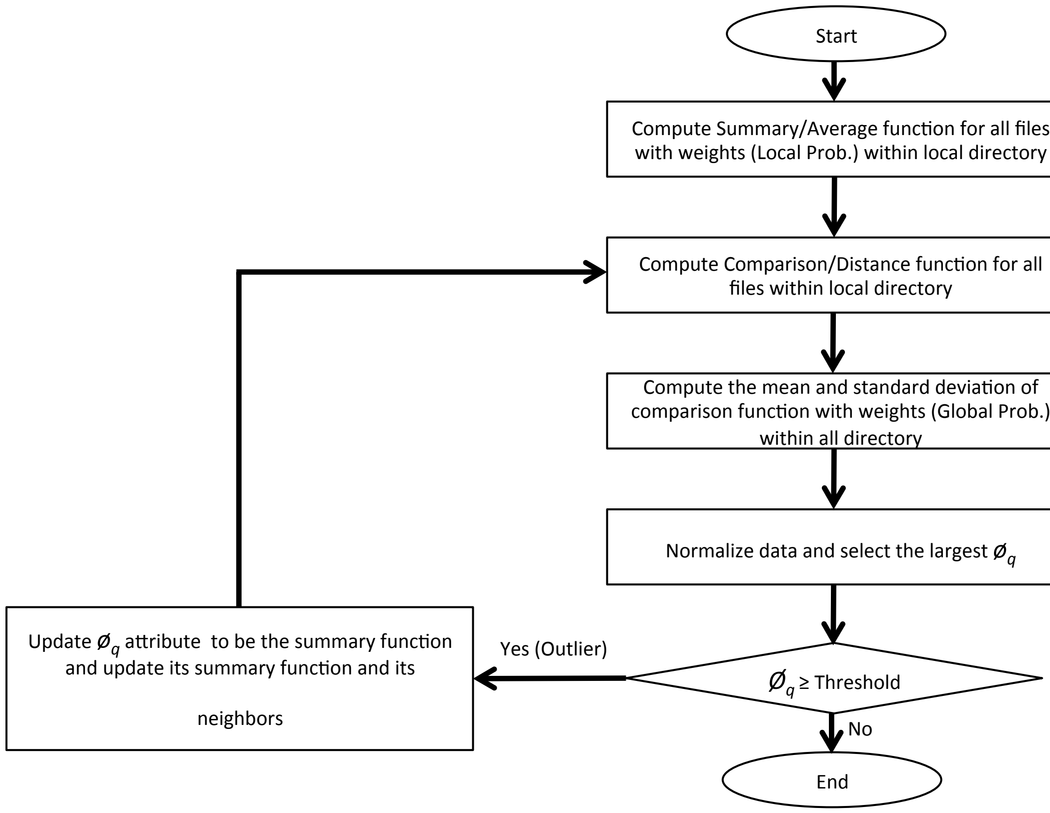

Start
Compute Summary/Average function for all files with weights (Local Prob.) within local directory
Compute Comparison/Distance function for all files within local directory
Compute the mean and standard deviation of comparison function with weights (Global Prob.) within all directory
Normalize data and select the largest Ø_q
Ø_q ≥ Threshold
Yes (Outlier)
No
Update Ø_q attribute to be the summary function and update its summary function and its neighbors
End

Figure 4.2: Flowchart of the Probabilistic Iterative z Algorithm.

$$\begin{aligned} p^l(\phi_i) &= \frac{(t_{aref} - t_a(\phi_i) + 1)^2}{\sum_{\phi \in N(\phi_i)} (t_{aref} - t_a(\phi) + 1)^2} \\ p^g(\phi_i) &= \frac{(t_{aref} - t_a(\phi_i) + 1)^2}{\sum_{\phi \in \mathcal{N}} (t_{aref} - t_a(\phi) + 1)^2} \end{aligned}$$

Depending on the value of t_{aref}, we may have p^l with either local or global `atime` and p^g with either local or global `atime`. In the case, for example, where p^l uses local `atime` and p^g uses global `atime`, the two distributions will be referred to as *global/local `atime`* distributions. The other three cases are easy to infer.

The competitive aspect of the current perspective can be easily seen from the two algorithms: the files are mutually exclusive entities and their probabilities add up to one. As such, the files are treated as if they are mutually independent. Of course, this is a limited perspective that we will improve next.

In the probabilistic iterative z algorithm, if we have

$$f(\phi_i) = -\log\left(\frac{|\phi_i|}{\sum_{\phi\in N(\phi_i)}|\phi|}\right),$$

then the attribute function f can be viewed as the information associated with each file. In addition, by taking

$$p^l_{N(\phi_i)}(\phi_i) = \frac{|\phi_i|}{\sum_{\phi\in N(\phi_i)}|\phi|},$$

we transform the probabilistic iterative z algorithm to the *information iterative z algorithm* shown in Algorithm 3 [1].

Algorithm 3 The information iterative z algorithm.

```
1: Let $\theta$ be a threshold
2: Let $n$ be $|\mathcal{N}|$
3: outlier=true
4: $\sigma = 0; \mu = 0$
5: for $i = 1 \to n$ do
6:     Let $N(\phi_i)$ be the set of files in $D(\phi_i)$ and $|N(\phi_i)|$ its cardinality
7:     Let $p^l_{N(\phi_i)}(\phi_i)$ be the local competitive probability of $\phi_i$
8:     Let $p^g(\phi_i)$ be the global competitive probability of $\phi_i$
9:     Set the attribute function $f(\phi_i)$ to be the information $I(\phi_i) = -\log p^l_{N(\phi_i)}(\phi_i)$
10:    Set the summary function $g(\phi_i)$ to be the entropy $H(\phi_i) = \sum_{\phi\in N(\phi_i)} p^l_{N(\phi_i)}(\phi) f(\phi)$
11:    Compute the comparison function $h(\phi_i) = f(\phi_i) - g(\phi_i)$
12:    $\mu = \mu + p^g(\phi_i)h(\phi_i)$
13:    $\sigma = \sigma + p^g(\phi_i)h(\phi_i)^2$
14: end for
15: $\sigma = \sqrt{\sigma - \mu^2}$                          ▷ The standard deviation
16: while outlier==true do
17:     outlier=false
18:     $\phi_q = \arg\max_\phi |\frac{h(\phi)-\mu}{\sigma}|$
19:     if $|\frac{h(\phi_q)-\mu}{\sigma}| \geq \theta$ then          ▷ $\phi_q$ is an outlier
20:         Mark $\phi_q$ as an outlier
21:         $f(\phi_q) = g(\phi_q)$
22:         Update $g(\phi)$ and $h(\phi)$ for every $\phi$ in $N(\phi_q)$
23:         Update $\mu$ and $\sigma$
24:         outlier=true
25:     end if
26: end while
```

In all of the previous algorithms, the comparison function $h(\phi)$ is expressed as the difference between the attribute function $f(\phi)$ and the summary function $g(\phi)$. This difference is supposed to quantify to what extent $f(\phi)$ is independent from $g(\phi)$. From the information theory perspective, it is more convenient to use mutual information for that. The idea behind mutual information is to compute the dependency between files instead of the difference as done in the outlier analysis. In fact, files created under a directory are expected to be somehow related (dependent) on each other. Therefore, mutual information is the right tool that will allow us to quantify such a dependency. For this to be done, the files should support each other and be treated as cooperative entities instead of being competitive entities only. This, however, entails introducing the *cooperative* probability $(p_r(\phi_i))$[1] distribution of files in addition to their *competitive* distributions as done in Algorithms 2 and 3.

The cooperative aspect between files leads to different Algorithms 4 [1] and 5. Notice how the comparison function $h(\phi)$ is expressed as the mutual information between ϕ and a virtual file with size $g(\phi)$. In addition, the inequality in line 21 of Algorithm 4 has $\leq$ instead of $\geq$ used in the previous Algorithms 1, 2 and 3. The reason is that $h(\phi)$ expresses the dependency between files instead of their independence.

In some cases, we may want to consider the directories as independent entities as done in [12]. In these cases, one should use the local means and the local standard deviations instead of the global ones (i.e., μ and σ).

Algorithms 1, 2, 3, 4 and 5 can be generalized to treat any elements of investigation (instead of files only) as well as multiple attributes (instead of a single attribute, namely the `size`). They can also be adapted to PDF by running them as needed, for example, when a new element of investigation is created or an attribute of an existing one changes. In addition, they can be run at regular intervals of time. As a remark, the previous runs of the algorithms should be used to reduce the processing by only computing the necessary functions. For example, the summary functions of investigation elements that are not neighbors of the updated elements should not be

[1]It is used in computing mutual information $I(\phi_i, \tilde{\phi}_i)$ between files. An example of computing $p_r(\phi_i)$ in practice is as follows. Let $p_r(\phi_i)$ be the probability that the file ϕ_i is legitimate based on size relative to other files within a directory. Let ϵ be a non-negative real number and let $N_\epsilon(\phi_i)$ be the number of files with size in interval $[|\phi_i| - \epsilon, |\phi_i| + \epsilon]$ within a directory. Also, let $n(d)$ be the total number of files within the same directory. That is,

$$p_r(\phi_i) = \frac{N_\epsilon(\phi_i)}{n(d)}.$$

Algorithm 4 The centered mutual information iterative z algorithm.

```
 1: Let $\theta$ be a threshold
 2: Let $n$ be $|\mathcal{N}|$
 3: outlier=true
 4: $\sigma = 0; \mu = 0$
 5: for $i = 1 \to n$ do
 6:     Let $N(\phi_i)$ be the set of files in $D(\phi_i)$ and $|N(\phi_i)|$ its cardinality
 7:     Let $p^l_{N(\phi_i)}(\phi_i)$ be the local competitive probability of $\phi_i$
 8:     Let $p^g(\phi_i)$ be the global competitive probability of $\phi_i$
 9:     Set the *attribute function* $f(\phi_i)$ to be $|\phi_i|$
10:     Set the *summary function* $g(\phi_i) = \sum_{\phi \in N(\phi_i)} p^l_{N(\phi_i)}(\phi) f(\phi)$
11:     Let $p_r(\phi_i)$ be the cooperative probability of $\phi_i$
12:     Let $\tilde{\phi}_i$ be a virtual file with size $g(\phi_i)$
13:     Compute the *dependency function* $h(\phi_i) = I(\phi_i, \tilde{\phi}_i)$
14:     $\mu = \mu + p^g(\phi_i)h(\phi_i)$
15:     $\sigma = \sigma + p^g(\phi_i)h(\phi_i)^2$
16: end for
17: $\sigma = \sqrt{\sigma - \mu^2}$                                    ▷ The standard deviation
18: while outlier==true do
19:     outlier=false
20:     $\phi_q = \arg\min_\phi |\frac{h(\phi)-\mu}{\sigma}|$
21:     if $|\frac{h(\phi_q)-\mu}{\sigma}| \leq \theta$ then                  ▷ $\phi_q$ is an outlier
22:         Mark $\phi_q$ as an outlier
23:         $f(\phi_q) = g(\phi_q)$
24:         Update $g(\phi)$ and $h(\phi)$ for every $\phi$ in $N(\phi_q)$
25:         Update $\mu$ and $\sigma$
26:         outlier=true
27:     end if
28: end while
```

recomputed.

Algorithm 5 The averaged mutual information iterative z algorithm.

```
1: Let $\theta$ be a threshold
2: Let $n$ be the total number of files
3: outlier=true
4: $\sigma = 0; \mu = 0$
5: for $i = 1 \to n$ do
6:     Let $N(\phi_i)$ be the set of files in $D(\phi_i)$ and $|N(\phi_i)|$ its cardinality
7:     Let $p_r(\phi_i)$ be the cooperative probability of the file $\phi_i$ being legitimate
8:     Let $p^l_{N(\phi_i)}(\phi_i)$ be the local competitive probability of selecting $\phi_i$ as the legitimate file amongst the files in $N(\phi_i)$
9:     Let $p^g(\phi_i)$ be the global competitive probability of selecting $\phi_i$ as the legitimate file amongst all files
10:    Set the *attribute function* be $f(\phi_i) = \sum_{\phi \in N(\phi_i)} p_r(\phi|\phi_i) I(\phi_i, \phi)$
11:    Set the *summary function* $g(\phi_i) = \sum_{\phi \in N(\phi_i)} p^l_{N(\phi_i)}(\phi) f(\phi)$
12:    Compute the *comparison function* $h(\phi_i) = f(\phi_i) - g(\phi_i)$
13:    $\mu = \mu + p^g(\phi_i) h(\phi_i)$
14:    $\sigma = \sigma + p^g(\phi_i) h(\phi_i)^2$
15: end for
16: $\sigma = \sqrt{\sigma - \mu^2}$                                ▷ The standard deviation
17: while outlier==true do
18:    outlier=false
19:    $\phi_q = \arg\max_\phi |\frac{h(\phi)-\mu}{\sigma}|$
20:    if $|\frac{h(\phi_q)-\mu}{\sigma}| \geq \theta$ then                       ▷ $\phi_q$ is an outlier
21:        Mark $\phi_q$ as an outlier
22:        $f(\phi_q) = g(\phi_q)$
23:        Update $g(\phi)$ and $h(\phi)$ for every $\phi$ in $N(\phi_i)$
24:        Update $\mu$ and $\sigma$
25:        outlier=true
26:    end if
27: end while
```

A more computationally expensive mutual information-based algorithm is given in 5.

An important aspect that should be emphasized is that the spatial outlier analysis as done in [12] uses the difference between the file sizes, as opposed to our case in which we are measuring the dependency between the files based on their sizes. So, our approach can be easily generalized by only changing the way the probabilities are computed; instead of basing the calculations on the sizes only, we can extend it to other attributes such as MAC (modification, access and creation) times, application type, and so on, as done next.

4.4 Information-Based Temporal Outlier Analysis

In the previous sections, especially Section 4.2, we discussed the spatial outlier analysis by focusing on particular element attributes fixed in time. This has the drawback of not considering the whole history of a target attribute and of possibly missing the time at which suspicious activities might happen. Looking at the whole history is a practical way to detect and trace most digital crimes. Such a requirement is even more stringent for a proactive system such as ours.

To make our generalization easy to follow, we will first focus on a single attribute, namely file size, as we did in Section 4.2.

Let ϕ be a file and let $(|\phi|(t_i))_{i\in\mathbb{N}}$ be the time series of its size at times $(t_i)_{i\in\mathbb{N}}$. Let $[t_b, t_e]$ be the time interval in which the analysis is desired. For every $|\phi|(t_i)$ for which $t_i \in [t_b, t_e]$ we associate a virtual file ϕ_{t_i} with size $|\phi_{t_i}| = |\phi|(t_i)$.

The probabilistic and information-based Algorithms 2, 3, 4, 5 can be used to do the outlier analysis of the (virtual) files ϕ_{t_i}. Any outlier file ϕ_{t_j} detected during the analysis represents a suspicious activity on the file ϕ at time t_j.

The interval $[t_b, t_e]$ could be taken to be the whole timeline of the file but, for efficiency purposes, it needs to be restricted, especially when the sampling frequency of recording the size is high. In what follows we propose three approaches to deal with a large number of samples.

4.4.1 Moving Window Temporal Analysis

This way of restricting the number of virtual files is based on the intuition that the suspicious size is an outlier compared to its neighbours in time. As such, the moving window outlier analysis has a Markov-like (memoryless) property as it does not take into consideration the past, far-distant virtual files (see Figure 4.3).

A novel criterion to detect strong outliers is to find files that stay outliers for all the windows they belong to. This could be enhanced with a voting system by comparing the number of windows for which the file is an outlier against those for which it is not.

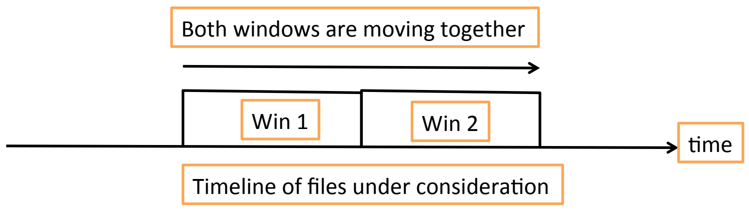

Both windows are moving together
Win 1
Win 2
time
Timeline of files under consideration

Figure 4.3: Moving Windows Temporal Analysis.

4.4.2 Fixed Window Temporal Analysis

The intuition supporting this kind of analysis is based on the observation that the old file sizes tend to be more legitimate. As such, the file sizes are compared against some past file sizes. The most forward way to implement this is to consider a time interval (fixed window) in the past $[t_b^p, t_e^p]$ and carry out the outlier analysis for the virtual files in that interval and the file under consideration (see Figure 4.4).

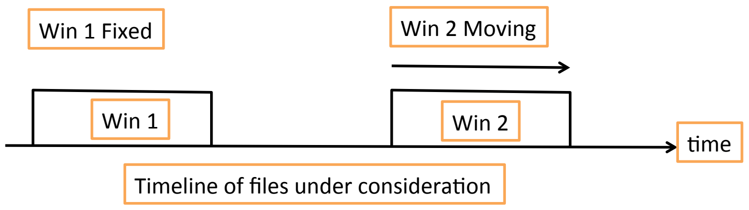


Figure 4.4: Fixed Windows Temporal Analysis.

If we allow the fixed window to move according to its distance to the virtual file under consideration, then we obtain what we call *Moving Fixed-window Temporal Analysis* (see Figure 4.5).

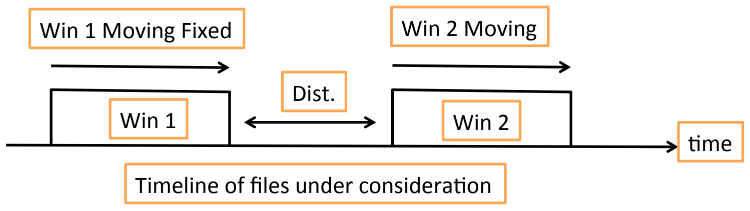


Figure 4.5: Moving Fixed Windows Temporal Analysis.

4.4.3 Hierarchical Temporal Analysis

In this case, we partition the timeline into a sequence of slots, then we carry out the outlier analysis on each. The outliers detected are then partitioned and the outlier analysis is again performed on them. The process is repeated until we reach a predetermined number of outliers (see Figure 4.6). The advantage of this multi-resolution approach is that it is parallel by nature (see Chapter 5).

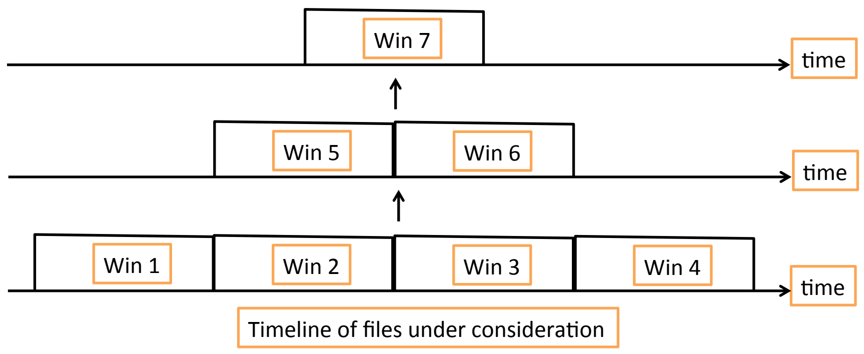


Figure 4.6: Hierarchical Temporal Analysis.

4.4.4 Modulo Temporal Analysis

A legitimate question that we did not address yet in this section is: "Why not use the time as an attribute and use the outlier detection algorithm to find the suspicious times at which the files were changed instead of all these variant temporal analysis algorithms?" The main reason is that time, by default, is changing (increasing). As such, the comparison function, which gives a sort of a distance, is not reliable to compare two changes in time. To avoid this natural time progression, the time could be used as an attribute but with a modulo transformation. By doing this, the time is represented within an interval, and any activity in time is mapped to this interval using the right modulo transformation. The choice of the interval depends on the kind of activity pattern the investigator is interested in. For example, a 24-hour interval should be used if daily patterns are sought. Just to be more precise, the modulo transformation we are dealing with is defined as follows.

Definition 4.1 (Modulo Transformation)**.** *Given two positive real numbers a and b such that $a < b$, the modulo transformation M with respect to the time interval $[a, b]$*

is defined as:

$$M(t) = t - (b - a)\lfloor (t - a)/(b - a) \rfloor.$$

For a given time t (a real number), the value $M(t)$ will be denoted $t[a, b]$.

4.5 Event-Based Outlier Detection and Analysis

As opposed to the above sections in which we consider the targets (with and without time), we briefly present how events can be used in the outlier analysis. The fact that most activities of a system are recorded in logs in the form of events makes this kind of analysis an excellent candidate for investigation. As we previously defined it, an event is a change in the target. This change is usually augmented with attributes such as when and who.

To simplify our discussion, we focus on file-change events, specifying that a file has changed, and, especially, on the file size change attribute. The size change and the time at which the event occurred will constitute the attributes $(\Delta s, t)$ of a file-change event e (e_f is used to denote the file changed). This implies combining the usual outlier detection algorithm with a temporal-based one to produce a multi-attribute outlier detection algorithm as shown in Algorithm 6. This algorithm uses the modulo transformation to map the time change of files to an interval I and the multi-attribute function $f(e) = (\Delta s, t[I])$. In the algorithm, we used the covariance matrix, as well as its inverse, to compute the so called *Mahalanobis* distance. This distance is better than the Euclidean distance as it takes into account any dependencies between the attributes [35].

Note that, in addition to its special temporal aspect, our algorithm is different from the one by Lu et al. [35] in the sense that it is an extension of the single attribute and it only marks the element with the maximum distance that is greater than or equal to the threshold θ as the outlier. The algorithm by Lu et al. does not have this iterative aspect that makes outlier detection algorithms more reliable. Our algorithm is also different from that of Carrier et al. ([12]) as it, in addition to the above, uses the right definition of Mahalanobis distance; in fact, Carrier et al. used the actual covariance matrix instead of its inverse.

Algorithm 6 The iterative z algorithm on file-change events using the multi-attributes $(\Delta s, t)$.

```
1: Let $\theta$ be a threshold
2: Let $\mathcal{N}$ be the set of file-changed events triggered in an interval of time
3: Let $n$ be $|\mathcal{N}|$
4: Let $I$ be a time interval
5: outlier=true
6: **for** $i = 1 \rightarrow n$ **do**
7:     Let $N(e_i)$ be the set of events for which the files are in $D(e_{if})$ and let $|N(e_i)|$ its cardinality
8:     Take the *attribute function* of $e_i$ to be $f(e_i) = (\Delta s_i, t_i[I])$
9:     Compute the *summary function* $g(e_i) = \frac{1}{|N(e_i)|} \sum_{e \in N(e_i)} f(e)$
10:    Compute the *comparison function* $h(e_i) = f(e_i) - g(e_i)$
11: **end for**
12: $\mu = \frac{1}{n} \sum_{e \in \mathcal{N}} h(e)$                                ▷ the mean vector
13: $\sigma = \frac{1}{n-1} (h(e) - \mu)^T (h(e) - \mu)$                     ▷ the covariance matrix
14: **while** outlier==true **do**
15:    outlier=false
16:    $e_q = \arg\max_e (h(e) - \mu)\sigma^{-1}(h(e) - \mu)^T$
17:    **if** $(h(e) - \mu)\sigma^{-1}(h(e) - \mu)^T \geq \theta$ **then**        ▷ $e_q$ is an outlier
18:        Mark $e_q$ as an outlier
19:        $f(e_q) = g(e_q)$
20:        Update $g(e)$ and $h(e)$ for every $e$ in $N(e_q)$
21:        Update $\mu$ and $\sigma$
22:        outlier=true
23:    **end if**
24: **end while**
```

4.6 Information-Based Spatial, Temporal, and Event Outlier Analysis

Instead of dealing with targets and events separately, we generalize the outlier analysis presented in the previous sections to all elements of investigation. Following the framework presented in the previous chapter, we suppose that we have a set of profiles $\mathbb{P}_{(T,E,\Gamma)}$, which we take to be the functional space $F = (\mathcal{T} \times \mathcal{E})^{\Gamma}$. This space not only includes the profiles that can happen on a real system (*actual* profiles) but also all kinds of imaginary profiles (*virtual* profiles). By doing so, a projection of an actual profile is treated as a (virtual) profile. The corresponding information iterative z algorithm is given by Algorithm 7.

Considering the profiles as points hides the dependency of the profiles on time and makes Algorithm 7 time-independent. In a proactive system, the profile is partially available and unfolds as the time progresses. For this reason, we propose looking at the profiles as a sequence of points (symbols) evolving in time and deal with time in a special manner. As done in Section 4.4, different algorithms could be given and we opt for Algorithm 8 to illustrate the idea. Moreover, the algorithms discussed so far do not take known profiles into consideration. It is very common to have some profiles that are known to be malicious and others that are not. Therefore we propose making use of such information to improve our algorithms, as discussed next.

Using known malicious profiles: Let $\mathbb{P}^m_{(T,E,\Gamma)}$ be the set of known malicious profiles and let $\mathcal{P}$ be a given profile. If an analysis technique, for example one of the outlier detection algorithms, is applied to the set of profiles $\mathbb{P}^m_{(T,E,\Gamma)} \cup \{\mathcal{P}\}$ and detects that $\mathcal{P}$ is not that different from the malicious profiles (e.g., $\mathcal{P}$ is not an outlier when an outlier detection algorithm is applied), then, most likely, $\mathcal{P}$ is a malicious profile. If that is not the case, then it is probably not a malicious profile similar to the known ones. To make sure it is not a new malicious profile, it still needs to be checked against the other usual profiles. To take this into account and to simplify the algorithm, we denote by $\mathcal{O}_m(\mathcal{P}, \mathbb{P}^m_{(T,E,\Gamma)})$ the function that returns `true` if $\mathcal{P}$ is not similar to the malicious profiles known so far $\mathbb{P}^m_{(T,E,\Gamma)}$ and `false` otherwise.

Using known safe profiles: Running an outlier detection algorithm on the set of safe profiles can be used to compute the minimum threshold for which no outlier is to be found. This threshold could be used to detect outliers in the unclassified profiles. From a practical perspective, however, this may yield a large value for the threshold, which can cause high false positives. In this dissertation, safe profiles will

Algorithm 7 The information iterative z algorithm on profiles.

```
 1: Let $\theta$ be a threshold
 2: Let $\mathcal{N}$ be the set of profiles $\mathbb{P}_{(T,E,\Gamma)}$, which we assume is finite
 3: Let $n$ be $|\mathcal{N}|$
 4: Let $.*$ denote the element-wise multiplication of vectors
 5: outlier=true
 6: **for** $i = 1 \to n$ **do**
 7:     Compute the neighborhood $N(\mathcal{P}_i)$ of the profile $\mathcal{P}_i$
 8:     Let $\vec{p}^{l}_{N(\mathcal{P}_i)}(\mathcal{P}_i)$ be the local competitive probabilities of $\mathcal{P}_i$
 9:     Let $\vec{p}^{g}(\mathcal{P}_i)$ be the global competitive probabilities of $\mathcal{P}_i$
10:     Set the *attribute function* $f(\mathcal{P}_i)$ to be the information vector $I(\mathcal{P}_i) = -\log \vec{p}^{l}_{N(\mathcal{P}_i)}(\mathcal{P}_i)$
11:     Set the *summary function* $g(\mathcal{P}_i)$ to be the entropy vector $H(\mathcal{P}_i) = \sum_{\mathcal{P} \in N(\mathcal{P}_i)} p^{l}_{N(\mathcal{P}_i)}(\mathcal{P}) f(\mathcal{P})$
12:     Compute the *comparison function* $h(\mathcal{P}_i) = f(\mathcal{P}_i) - g(\mathcal{P}_i)$
13: **end for**
14: $\mu = \sum_{\mathcal{P} \in \mathcal{N}} \vec{p}^{g}(\mathcal{P}) .* h(\mathcal{P})$    ▷ the mean vector
15: $\sigma = \sum_{\mathcal{P} \in \mathcal{N}} (\vec{p}^{g}(\mathcal{P}) .* (h(\mathcal{P}) - \mu))^T (h(\mathcal{P}) - \mu)$    ▷ the covariance matrix
16: **while** outlier==true **do**
17:     outlier=false
18:     $\mathcal{P}_q = \arg\max_{\mathcal{P}} (h(\mathcal{P}) - \mu)\sigma^{-1}(h(\mathcal{P}) - \mu)^T$
19:     **if** $(h(\mathcal{P}) - \mu)\sigma^{-1}(h(\mathcal{P}) - \mu)^T \geq \theta$ **then**    ▷ $\mathcal{P}_q$ is an outlier
20:         Mark $\mathcal{P}_q$ as an outlier
21:         $f(\mathcal{P}_q) = g(\mathcal{P}_q)$
22:         Update $g(\mathcal{P})$ and $h(\mathcal{P})$ for every $\mathcal{P}$ in $N(\mathcal{P}_q)$
23:         Update $\mu$ and $\sigma$
24:         outlier=true
25:     **end if**
26: **end while**
```

be used in a rudimentary way: the outlier profiles should not include the known safe profiles. More precisely, if the set of profiles used in the outlier detection algorithm is $\mathbb{P}_{(T,E,\Gamma)}$ and the set of known safe profiles is denoted by $\mathbb{P}^s_{(T,E,\Gamma)}$, then the maximum of the Mahalanobis distances is taken over $\mathbb{P}_{(T,E,\Gamma)} \setminus \mathbb{P}^s_{(T,E,\Gamma)}$. This ensures that the outliers found are not part of the safe profiles.

To each profile $\mathcal{P}$, we associate a threat level $\mathcal{P}^{th}$ that we initialize at zero when the profile is created. As the time progresses, the threat level is adjusted according to the outlier detection outcome. When the threat level exceeds a specified threshold γ, the proactive component system should generate a notification and a report to the designated personnel. Proactive actions could also be taken as required.

Algorithm 8 The generalized information iterative z algorithm on profiles.

```
1: Let $\theta$ and $\gamma$ be thresholds
2: Let $\mathcal{N}$ be the set of profiles $\mathbb{P}_{(T,E,\Gamma)}$, which we assume is finite
3: Let $n$ be $|\mathcal{N}|$
4: Let $.*$ denote the element-wise multiplication of vectors
5: Initialize all threat levels to 0
6: outlier=true
7: for $i = 1 \to n$ do
8:     Update the set of malicious profiles $\mathbb{P}^m_{(T,E,\Gamma)}$
9:     if $\mathcal{O}_m(\mathcal{P}_i, \mathbb{P}^m_{(T,E,\Gamma)})$ ==false then
10:        Mark $\mathcal{P}_i$ as an outlier and remove it from the set of profiles
11:        continue
12:    end if
13:    Update the set of safe profiles $\mathbb{P}^s_{(T,E,\Gamma)}$
14:    Compute the neighborhood $N(\mathcal{P}_i)$ of the profile $\mathcal{P}_i$ at time $t$
15:    Let $\vec{p}^{\,l}_{N(\mathcal{P}_i)}(\mathcal{P}_i, t)$ be the local competitive probabilities of $\mathcal{P}_i$ at time $t$
16:    Let $\vec{w}(\mathcal{P}_i, t)$ be probability distributions of $\mathcal{P}_q$ in time
17:    Let $\vec{p}^{\,g}(\mathcal{P}_i, t)$ be the global competitive probabilities of $\mathcal{P}_i$ at time $t$
18:    Set the *attribute function* $f(\mathcal{P}_i, t)$ to be the information vector $I(\mathcal{P}_i, t) = -\int_{-\infty}^{t} \vec{w}(\mathcal{P}_i, x) \log \vec{p}^{\,l}_{N(\mathcal{P}_i)}(\mathcal{P}_i, x) dx$
19:    Set the *summary function* $g(\mathcal{P}_i, t)$ to be the entropy vector $H(\mathcal{P}_i, t) = \sum_{\mathcal{P} \in N(\mathcal{P}_i)} p^l_{N(\mathcal{P}_i)}(\mathcal{P}, t) f(\mathcal{P}, t)$
20:    Compute the *comparison function* $h(\mathcal{P}_i, t) = f(\mathcal{P}_i, t) - g(\mathcal{P}_i, t)$
21: end for
22: $\mu(t) = \sum_{\mathcal{P} \in \mathcal{N}} \vec{p}^{\,g}(\mathcal{P}, t) .* h(\mathcal{P}, t)$    ▷ The mean vector
23: $\sigma(t) = \sum_{\mathcal{P} \in \mathcal{N}} (\vec{p}^{\,g}(\mathcal{P}, t) .* (h(\mathcal{P}, t) - \mu(t)))^T (h(\mathcal{P}, t) - \mu(t))$    ▷ The covariance matrix
24: while outlier==true do
25:    outlier=false
26:    $\mathcal{P}_q = \arg\max_{\mathcal{P} \in \mathbb{P}_{(T,E,\Gamma)} \setminus \mathbb{P}^s_{(T,E,\Gamma)}} (h(\mathcal{P}, t) - \mu(t)) \sigma(t)^{-1} (h(\mathcal{P}, t) - \mu(t))^T$
27:    if $h(\mathcal{P}_q, t) - \mu(t)) \sigma(t)^{-1} (h(\mathcal{P}_q, t) - \mu(t))^T \geq \theta$ then
28:        $\mathcal{P}_q^{th} += h(\mathcal{P}_q, t) - \mu(t)) \sigma(t)^{-1} (h(\mathcal{P}_q, t) - \mu(t))^T$
29:        $f(\mathcal{P}_q, t) = g(\mathcal{P}_q, t)$
30:        Update $g(\mathcal{P}, t)$ and $h(\mathcal{P}, t)$ for every $\mathcal{P}$ in $N(\mathcal{P}_q)$
31:        Update $\mu(t)$ and $\sigma(t)$
32:        outlier=true
33:    end if
34: end while
35: Report every profile $\mathcal{P}$ with $\mathcal{P}^{th} \geq \gamma$ if any
```

Chapter 5

Implementation of Proactive Digital Forensics

For a successful implementation of the proactive digital forensics, one needs to keep the design and the implementation as close as possible to the phases and the workflow of the proactive system depicted in Figure 2.1. Moreover, the design should satisfy other requirements such as modularity, extensibility, reliability and efficiency. As such, we opted for an object-oriented design and modelled our proactive system accordingly, as discussed in the next section.

Moreover, for testing, validating and verifying our proactive system implementation, we used two main cases:

- The Honeypot test case to evaluate and validate only the analysis phase of the proactive system. Algorithms 1 [12], 2 [1] and 3 [1] were used for this.
- The other case is Zeus, which allows us to validate all phases of the proactive system. In the analysis phase, Algorithm 8 was used.

In addition, the implementation done in this chapter is based on the theory developed in Chapter 3 from the following perspectives:

- Projecting the profiles of the system onto important targets and events (this related to Sections 3.3.2, 3.3.1, 3.3.4, 3.4 and 5.4).
- Developing the forensic rules and the forensic rule engine to take actions when events are triggered (this related to Sections 3.2.2, 3.4 and 5.4).

- Analyzing the profiles of the system in the Zeus case via an analysis forensic rule implementing Algorithm 8 (this related to Sections 3.4, 4.6 and 5.4).

5.1 Proactive System Components

The UML component architecture diagram of our proactive system is shown in Figure 5.1 and matches the phases of the proactive system in Figures 2.1 and 3.2.

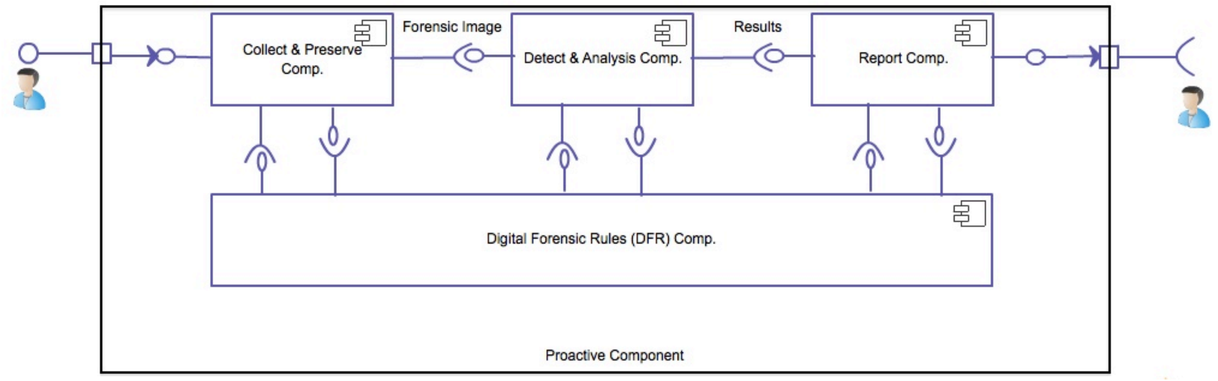


Figure 5.1: Proactive System Architecture.

The first component is the proactive collection and preservation component, which collects and preserves specific elements of the targeted system based on their criticality. More elements can be handled by this component as needed.

The second component is the proactive detection and analysis component. It is responsible for detecting any suspicious event by relying on external applications, such as an IDS, and/or internal implementation of event triggering functions (e.g., outlier detection analysis). When a suspicious event is detected, a more in-depth analysis is carried out accordingly by calling the appropriate digital forensic rules, as dictated by the digital forensic rules component. In addition, the analysis ensures that more elements of investigation are collected and preserved as required.

The third component is the preliminary/proactive report component where the relevant evidence is gathered, logged and sent to the investigator. The report is nothing more than a summary of the findings of the second component.

The last component is the Digital Forensic Rules (DFR) component. DFR is the brain of the proactive digital forensic system; it controls and executes forensic rules depending on the events triggered by different components. A forensic rule, introduced in Section 3.2.2, is implemented as an event-driven rule, similar to a statement in the usual languages such as Java or C/C++ or to an expert system rule. DFR is an event-driven model of computation composed of forensic rules. By that we mean, when specific events happen, DFR activates specific forensic rules accordingly to a model of computation. Each activated rule carries out a sequence of actions and triggers a sequence of events.

To fulfil the requirements of the design mentioned above, we adhered to a few design patterns including those of the following types: *strategy* (used in targets, actions and forensics rules), *observer* (used in triggering function and forensic rule engine), *composition* (used in collector) and *decoration* (used in targets bijection). A class diagram for the whole system is provided in Appendix A.

5.2 Work-Flow of the Proactive System Implementation

As mentioned in Section 3.3.4, the implementation of the system is done in a projected space, which the user should initially specify. This space should include all targets and events that the user or the organization has classified as important. The collection and preservation are restricted to the projected space but the analysis can be done on any space including the projected one.

More precisely, the work-flow of the proactive system is as follows:

1. The user specifies a set of targets $T = \{T_1, T_2, T_3, \ldots, T_n\}$ that she/he is interested in collecting, preserving and/or analyzing. These targets can be customized to match the needs of any individual organization. Examples of such targets are network traffic, firewall, processes, file system and registry.

2. The user also specifies a set of events $E = \{E_1, E_2, E_3, \ldots, E_m\}$ that she/he is interested in watching for. These events can be customized according to the user's requirements. Examples of events are suspicious network traffic, firewall status, process state change, file system modification, and registry modification.

3. Initially, a set of targets is proactively collected and preserved in a forensically sound manner. When an event, which can be associated with a target from DFR or an outside source such as IDS, is trigged, specific targets including the ones associated with the event triggered can proactively be collected and preserved as well.

4. When one or more events are detected, via a proactive event detection system such as an IDS/outlier detection analysis, the forensic rules are executed.

5. When an event E_i is trigged, the preservation mapping, which plays the role of a hashing function, proactively preserves all the predefined targets (i.e., targets associated with the event E_i). Moreover, extra targets T', which have not been included in the predefined ones, can be proactively collected and preserved. The advantage of having forensic rules is that the preservation phase can be expressed as a forensic rule as follows:

$$@E_i \rightarrow P\{DE_i \cup T'\},$$

where DE_i is the set of targets associated with E_i via a binary relation D.

6. The proactive analysis initiates the analysis forensic rules, such as the outlier analysis, and correlates the evidence from the data collected.

7. The preliminary report is then generated based on the evidence gathered as well as its timeline. In addition, a confidence number that quantifies the uncertainty of an incident is also reported. For the outlier analysis this number represents the normalized distance from the mean for single attribute or the Mahalanobis distance for the generalized case.

5.3 Distributed Proactive System

To be proactive and to handle the cases where many systems are involved (as is the case for networks or clouds), one has to consider not only one system but the ensemble. For that we propose a simple framework in which each system is equipped with a Proactive Sub-Agent (PSA) that may carry out PDF on the system it is running on and relies on a Proactive Agent (PA) to do the full PDF for the ensemble. As an illustration, PSA could deal with known safe and malicious profiles while PA runs the full outlier detection analysis.

Although PA can do most of the processing, PA in turn may rely on a High Performance Computing (HPC) cluster to store data and do the analysis in real time. Figure 5.2 depicts this framework. In addition, a multi-resolution approach is introduced to cope with the increasing size of data when an investigation is needed.

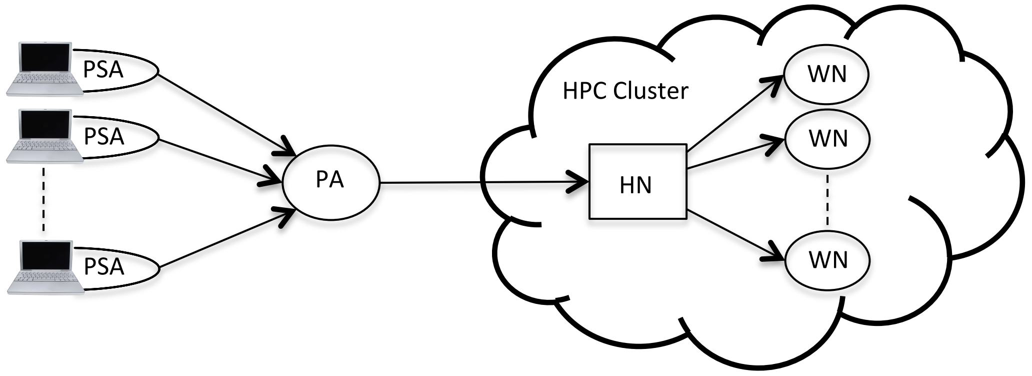


Figure 5.2: General Architecture of Distributed Proactive System (PSA: Proactive Sub-agent; PA: Proactive Agent; HN: Head Node; WN: Worker Node).

In this section, we are only considering the analysis phase, which is the most expensive phase of the proactive system. We have used our dataset from the Honeypot Challenge test case [26] to evaluate and validate some of the outlier algorithms in Chapter 4 which are also published in [12] and [1].

5.3.1 A Multi-Resolution Framework for Digital Forensics

The projection framework presented in Section 3.3.4 can be extended by iteratively applying it to an initial space. This leads to what we call *the multi-resolution framework for digital forensics*.

The multi-resolution approach allows us to cope with the large number of elements of DF investigation and the high dimensional space of the system by reducing the set of initial elements of DF investigation S_0 to a smaller one S_1. This in turn is reduced to S_2 and so on. The sequence $(S_n)_{n\in\mathbb{N}}$ obtained satisfies

$$S_0 \supseteq S_1 \supseteq \ldots \supseteq S_n \supseteq \ldots$$

A set S_{n+1} is usually the outcome of applying an operator Θ_p to S_n, that is

$$S_{n+1} = \Theta_p(S_n)$$

Definition 5.1. *A* DF reduction operator Θ *is an operator that maps a set S of elements of investigation to $S' = \Theta(S)$ such that $S' \subseteq S$.*

Although the DF reduction operator is a new concept, it is commonly used in practice. For example, taking a snapshot of the system at a specific time t can be viewed as a DF reduction operator that reduces the elements of investigation to the files at t. Another example is generating the timeline of the system in the interval $[t_i, t_e]$. It can be seen as reducing all the events and activities on the system to the ones reported by the operating system in the interval $[t_i, t_e]$. It is worth noting that a DF reduction operator can also be viewed as a projection operator that projects a DF space to a smaller one.

In our present work, the reduction operators are constructed from the spatial outlier detection algorithms. Given a set of investigation elements $\mathcal{E}$, the spatial outlier analysis reduces $\mathcal{E}$ to suspicious investigation elements using single and/or multiple attributes.

Lemma 5.1. *Given a sequence of DF reduction operators* $\Theta_1, \ldots, \Theta_n$*, then the composition* $\Theta_1 \circ \ldots \circ \Theta_n$ *is a DF reduction operator. In particular, if* $\Theta_1 = \Theta_2 = \ldots = \Theta_n = \Theta$*, the composition* $\Theta^n = \Theta_1 \circ \ldots \circ \Theta_n$ *is a reduction operator for every* $n \in \mathbb{N}$*.*

Definition 5.2. *A DF reduction operator* Θ *is said to* preserve V *iff* $V \subseteq \Theta(V \cup S)$ *for every DF state set* S*.*

In the definition above, if we take $S = \emptyset$, then we have the following lemma.

Lemma 5.2. *If a DF reduction operator* Θ *preserves* V*, then* V *is a fixed point of* Θ*, i.e.,*

$$\Theta(V) = V.$$

Definition 5.3. *A DF reduction operator is* safe *if it preserves the evidence.*

Since it is undecidable to find the evidence of all kind of DF attacks, it is impossible to build a safe computational DF reduction operator.

Definition 5.4. *Given a DF reduction operator* Θ *and a set* S *of elements of investigation, we define the* reduction amplitude at level n *of* Θ *on* S *with respect to another set* H *of elements of investigation as*

$$|\Theta|_H^n(S) = 1 - \frac{|H \cap \Theta^n(S)|}{|H \cap S|}$$

In what follows, $|\Theta|^n(S)$ will be used to denote $|\Theta|_{S_0}^n(S)$, where S_0 is the initial set of investigation elements. Note that the amplitude of reduction at level n of Θ is the same as the amplitude of reduction at level 1 of Θ^n. In other words, $|\Theta^n|_H(S) = |\Theta|_H^n(S)$.

The reason for introducing the amplitude of reduction is to quantify the effectiveness of a reduction operator. For example in the Honeypot Challenge test case (see Section 5.3.3), when using the `mtime` attribute under Algorithm 1, the amplitude of reduction at level 1 with respect to all possible set of states S_0 is

$$|\Theta_m|(S_0) = 1 - \frac{|\Theta_m(S_0)|}{|S_0|} = 1 - \frac{8127}{20861} = 0.6$$

The `size` attribute, however, gives

$$|\Theta_s|(S_0) = 1 - \frac{|\Theta_s(S_0)|}{|S_0|} = 1 - \frac{20067}{20861} = 0.04$$

From these results we can see using the `mtime` attribute give us better reduction than using the `size` attribute (see Tables 5.1 and 5.2). With respect to the evidence E, we have

$$|\Theta_m|_E(S_0) = 1 - \frac{|E \cap \Theta_m(S_0)|}{|E \cap S_0|} = 1 - \frac{18}{60} = 0.7$$

and

$$|\Theta_s|_E(S_0) = 1 - \frac{|E \cap \Theta_s(S_0)|}{|E \cap S_0|} = 1 - \frac{58}{60} = 0.03$$

The goal is to have $|\Theta|$ as big as possible for the initial set S_0 but as small as possible for the evidence E. As such, we introduce the *evidence amplitude at level n* to be

$$[\Theta]^n(S) = \frac{|E \cap \Theta^n(S)|}{|E \cap S|} = 1 - |\Theta|_E^n(S)$$

For example, we have

$$[\Theta_m]_E(S_0) = 1 - |\Theta_m|_E(S_0) = 1 - 0.7 = 0.3$$

and

$$[\Theta_s]_E(S_0) = 1 - |\Theta_s|_E(S_0) = 1 - 0.03 = 0.97$$

This implies that using the `size` attribute, the evidence was reduced compared to `mtime` attribute (see Tables 5.1 and 5.2). Therefore `mtime` attribute is better when the evidence is considered.

It is clear that a good reduction operator should have $|\Theta|^n_{S_0}(S)$ and $[\Theta]^n(S)$ as large as possible. In other words, it should reduce S to the greatest possible extent while leaving E intact. Therefore, the perfect reduction operator is the one for which $\Theta^n(S)$ converges to $E \cap S$ as n increases.

5.3.2 Implementation and Results

The requirement of analysing data in real time, as stated in the seven principles, led us to implement parallel versions of the spatial outlier analysis Algorithms 1 [12], 2 [1] and 3 [1] based on Message Passing Interface (MPI). MPI is a standard for writing parallel libraries that allows processes on different machines to communicate and share a common address space through Remote Direct Memory Access (RDMA). Several implementations of MPI exist and support the InfiniBand (IB) standard for low-latency and high throughput network communications. Since WestGrid has the IB hardware as well as the MPI implementations (openMPI and Intel MPI), we opted to using WestGrid as the HPC cluster for our distributed proactive system.

The parallel implementation is based on the following steps:

1. Group regular files according to their folders.

2. Distribute the groups across the processes.

3. Initialize the attribute functions and the probability distributions.

4. Compute the summary/average function $g(\phi)$ for each file ϕ.

5. Compute the comparison/distance function $h(\phi) = f(\phi) - g(\phi)$ for each file ϕ.

6. Compute the partial mean μ_l of the comparison functions and the partial mean of their squares ζ_l locally.

7. Compute the sum μ of μ_l and the sum ζ of ζ_l collectively using `MPI_Allreduce` with `MPI_SUM` as its operation.

8. Compute the standard deviation $\sigma = \sqrt{\zeta - \mu^2}$.

9. Compute the normalized value $y(\phi) = |(h(\phi) - \mu)/\sigma|$ for each file ϕ and select its largest value y_l locally.

10. Select the largest value y_m amongst y_l across all the processes and its location s_r using `MPI_Allreduce` with `MPI_MAXLOC` as its operation.

11. If y_m is greater than or equal to a threshold θ, then it is an outlier.

    (a) If my rank is equal to s_r,

        i. Update the attribute of the outlier to be the summary function.

        ii. Update its summary function and that of its neighbours.

        iii. Update the partial means μ_l and ζ_l.

    (b) go to step 7.

12. Stop if y_m is less than θ.

Since most forensic images have many directories, and to reduce the amount of communications involved in computing the summary functions, we first partition the files into groups based on the directories. This partition is done using a Perl script that uses GNU Parallel [66] to create the groups on multiple processes. To ensure a reasonable workload balance among, say, P processes, we sort the groups ascendingly according to their number of files and distribute them in a slightly modified round-robin fashion: if W is the number of directories, then at each round r the process with rank $r\%P$ (remainder of r by P) is chosen to be the initial round-robin element for the first W/P directories. The rest ($W\%P$ directories) is distributed to the processes with less workload. Then for each process, the name of the files it handles are written to a formatted file.

Each process then reads its file and initializes the local and global probability distributions. Different probability distributions were implemented to compare their effect on the outlier analysis as discussed below. The summary functions, the comparison functions and its partial mean and squares are then computed locally. Using `MPI_Allreduce` the mean and the standard deviation are computed. These are then used to normalized the comparison functions and compute the maximum value of the local normalized values. To get the global maximum value y_m and the rank r of the process from which it originated, we use the operation `MPI_MAXLOC` in `MPI_Allreduce` instead of `MPI_MAX`. The outlier detection is then executed by comparing y_m with a threshold θ. If y_m is greater than or equal to θ, then the process of rank r updates the necessary functions and recomputes the partial means and mean squares. All the processes then compute the global mean and the global standard deviation as outlined above. These steps are repeated until no outliers are detected (i.e., y_m is less than θ).

Since the group of files that each process handles are specified in a formatted file F, implementing the multi-resolution approach is straightforward: each time an outlier is detected, it is recorded in a formatted file similar to F. This file is read instead of F when the next multi-resolution level is desired.

Note that when the directories are assumed to be independent entities, the two MPI calls above are ignored and the code becomes a pure, embarrassingly parallel one. This of course an oversimplification and it is not the case in practice as the directories are usually correlated.

5.3.3 Honeypot Challenge Test Case: Implementation Verification and Validation

Although our aim is to run the outlier detection algorithms in a PDF context, we needed a benchmark dataset to evaluate the effectiveness of our algorithms in [1] and to test and compare our implementation with the work in [12]. The comparison showed promising results which we discuss next.

The Honeypot Challenge benchmark [26] is set of images of a computer that was attacked just after it was brought online. The images are of six partitions and the largest is the `/usr` partition (`honeypot.hda5.dd`) This partition contains a total number of **20,861** files. The attacker used the Linux root kit version 4 (`lrk4`) to compromise the system and introduced around **60** infected files in `/usr`.

To handle the I/O bound aspect of DF, we first mounted the images on ramdisk on a cluster worker node with 24 GB of memory and 8 cores with read-only access. We then run the serial and parallel versions of different algorithms to make sure that our parallel implementation results match the serial ones. All the parallel algorithms, including the ones using the multi-resolution technique, completed in less than 2 minutes. On a Dual core MacBook Pro with 4 GB of memory, it took almost 12 minutes. Although the dataset used is small, we obtained a speedup of 6.

The analysis was done for many attributes including `size` , `mtime` , `atime` , `ctime` and `inode` number, for different global and local probability distributions such as `uniform` (represents Algorithm 1 by Carrier et al. in [12]), `mtime` [1], `atime` [1] and `ctime` [1] distributions, and for the information iterative z algorithm [1]. Some of the results obtained are presented in Tables 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8.

For more clarification of the tables listed below, we have two types of tables. The first table is for displaying the number of files left over after applying the outlier algorithms at each reduction level (up to 5 levels). The second table is for showing the percentage of reductions in both reduction and evidence amplitudes which is generated from the first table (see Section 5.3.1). Terms, used in tables, are defined as follows:

- Set Reduction: the reduction in the number of files in a set after applying the outlier detection algorithms.
- Evidence Reduction: the reduction in the number of files that were actually involved in the incident after applying the outlier detection algorithms.
- Reduction Amplitude: the percentage of reduction in the reduction amplitude generated after applying the outlier detection algorithms on a set (see Section 5.3.1). This number is a reflection of what is in the first table (set reduction column). For the best results, this percentage has to be as big as possible.
- Evidence Amplitude: the percentage of reduction in the evidence amplitude (actual files involved in the incident) generated after applying the outlier detection algorithms (see Section 5.3.1). In addition, this number is a reflection of what is in the first table (evidence reduction column). For the best results, this percentage has to be as small as possible to keep evidence intact.

$\theta = 2$, Prob. Dist. = Uniform						
Multi-resolution Level	`size`		`mtime`		`inode`	
	Set Red.	Evidence Red.	Set Red.	Evidence Red.	Set Red.	Evidence Red.
1	20067	58	8127	18	19964	50
2	20002	58	7198	17	19428	41
3	19814	54	1883	6	19251	39
4	19653	50	1511	5	19251	39
5	19652	50	1450	5	19251	39

Table 5.1: Number of files left from running iterative z algorithm (Algorithm 1) for `size`, `mtime` and `inode` attributes under uniform distributions.

A brief analysis of the results is as follows:

$\theta = 2$, Prob. Dist. = Uniform						
Multi-resolution Level	`size`		`mtime`		`inode`	
	Reduction Amp.(%)	Evidence Amp.(%)	Reduction Amp.(%)	Evidence Amp.(%)	Reduction Amp.(%)	Evidence Amp.(%)
1	4	97	60	30	4	83
2	4	97	65	28	7	68
3	5	90	91	10	8	65
4	6	80	93	8	8	65
5	6	80	93	8	8	65

Table 5.2: Comparing the percentage of `size`, `mtime` and `inode` attributes under uniform distributions for iterative z algorithm (Algorithm 1).

$\theta = 2$, Prob. Dist. = Global/Local `mtime`						
Multi-resolution Level	`size`		`mtime`		`inode`	
	Set Red.	Evidence Red.	Set Red.	Evidence Red.	Set Red.	Evidence Red.
1	20221	53	9312	16	20403	53
2	19531	48	6843	11	20202	48
3	11904	43	5389	9	20131	47
4	4942	31	2506	6	20091	47
5	4668	30	2386	6	20065	47

Table 5.3: Number of files left from running probabilistic iterative z algorithm (Algorithm 2) for `size`, `mtime` and `inode` attributes under global/local `mtime` distributions.

- The results we obtained in Tables 5.3 and 5.4 for the `size` attribute under Algorithm 2 [1] were better than the results obtained via implementing Algorithm 1 [12] in Tables 5.1 and 5.2. In a sense, we have got a bigger reduction for the reduction amplitude and less evidence reduction for the evidence amplitude. For example, at the multi-resolution level 5 (under `size` attribute) in Table 5.4, we got 70% difference (bigger) for the reduction amplitude and 30% less (smaller) for the evidence amplitude than what is in Table 5.2 (under `size` attribute). For the other attributes (`mtime` and `inode`) in Tables 5.2 and 5.4, we got similar results. More comparison can be seen in Tables 5.5 and 5.6.

- The reduction amplitude for `mtime` attribute is higher than that of `size` and `inode` attributes (see Tables 5.2 and 5.4). In addition, the evidence amplitude for `mtime` is less than that of `size` and `inode` for the same tables.

$\theta = 2$, Prob. Dist. = Global/Local `mtime`						
Multi-resolution level	`size`		`mtime`		`inode`	
	Reduction Amp.(%)	Evidence Amp.(%)	Reduction Amp.(%)	Evidence Amp.(%)	Reduction Amp.(%)	Evidence Amp.(%)
1	3	88	55	27	2	88
2	6	80	67	18	3	80
3	43	72	74	15	3	78
4	76	52	88	10	4	78
5	78	50	89	10	4	78

Table 5.4: Comparing the percentage of `size`, `mtime` and `inode` attributes under global/local `mtime` distributions for probabilistic iterative z algorithm (Algorithm 2).

$\theta = 2$, `mtime`				
Multi-resolution level	Prob. Dist. Uniform		Prob. Dist. Global/Global `atime`	
	Set Red.	Evidence Red.	Set Red.	Evidence Red.
1	8127	18	8136	16
2	7198	17	7332	13
3	1883	6	7036	13
4	1511	5	4541	10
5	1450	5	4413	10

Table 5.5: Number of files left from running uniform (Algorithm 1) and global/global `atime` (Algorithm 2) distributions under `mtime` attribute.

- The information-based outlier detection algorithm [1] using the `mtime` attribute outperformed the rest of the algorithms including the one reported by Carrier et al. [12] (see Table 5.2) in most cases as in Table 5.8. This can be seen from the fact that we got a bigger reduction in the reduction amplitude and a smaller one for the evidence amplitude. As an example, at the multi-resolution level 1 (under Prob. Dist. global/global `mtime`) in Table 5.8, we got a 55% difference (bigger) for the reduction amplitude and 54% less for the evidence amplitude than what is in Table 5.2 (under `size` attribute). Moreover, at the multi-resolution level 1 (under Prob. Dist. global/global mtime) in Table 5.8, we got a 56% difference (bigger) for the reduction amplitude and 45% less for the evidence amplitude than what is in Table 5.4 (under `size` attribute).

- Global/local distribution can help in increasing the reduction amplitude as the

$\theta = 2$, `mtime`				
Multi-resolution level	Prob. Dist. Uniform		Prob. Dist. Global/Global `atime`	
	Reduction Amp.(%)	Evidence Amp.(%)	Reduction Amp.(%)	Evidence Amp.(%)
1	61	30	61	27
2	65	28	65	22
3	91	10	66	22
4	93	8	78	17
5	93	8	79	17

Table 5.6: Comparing the percentage of uniform (Algorithm 1) and global/global `atime` (Algorithm 2) distributions under `mtime` attribute.

$\theta = 2$, Information-based detection using `mtime`				
Multi-resolution level	Prob. Dist. Global/Global `mtime`		Prob. Dist. Local/Local `mtime`	
	Set Red.	Evidence Red.	Set Red.	Evidence Red.
1	8612	26	11209	21
2	7018	23	9277	13
3	1805	7	8582	12
4	1603	6	8242	12
5	1473	6	7979	12

Table 5.7: Number of files left from running global/global and local/local `mtime` distributions under information iterative z algorithm (Algorithm 3) using `mtime` attribute.

multi-resolution level increases as shown for the example in Table 5.4.

5.4 Zeus Use Case

5.4.1 Why Zeus?

The Zeus toolkit has become the hacker's best toolkit due to its user-friendly interface and its stealth capabilities. According to Symantec Corporation [65], it is considered to be the "King of the Underground Crimeware Toolkits." On Damballa [15], Guntter Ollmann ranked it as the number-one threat which infected 3.6 million hosts in the US alone (this accounts for 19% of the Internet-connected PCs in the US). Moreover, about 44% of banking malware infections were attributed to Zeus [37]. As such, we

$\theta = 2$, Information-based detection using `mtime`				
Multi-resolution level	Prob. Dist. Global/Global `mtime`		Prob. Dist. Local/Local `mtime`	
	Reduction Amp.(%)	Evidence Amp.(%)	Reduction Amp.(%)	Evidence Amp.(%)
1	59	43	46	35
2	66	38	56	22
3	91	12	59	20
4	92	10	60	20
5	93	10	62	20

Table 5.8: Comparing the percentage of global/global and local/local `mtime` distributions under information iterative z algorithm (Algorithm 3) using `mtime` attribute.

took Zeus as an excellent use case to test and validate our implementation.

5.4.2 Description of Zeus Crimeware Toolkit

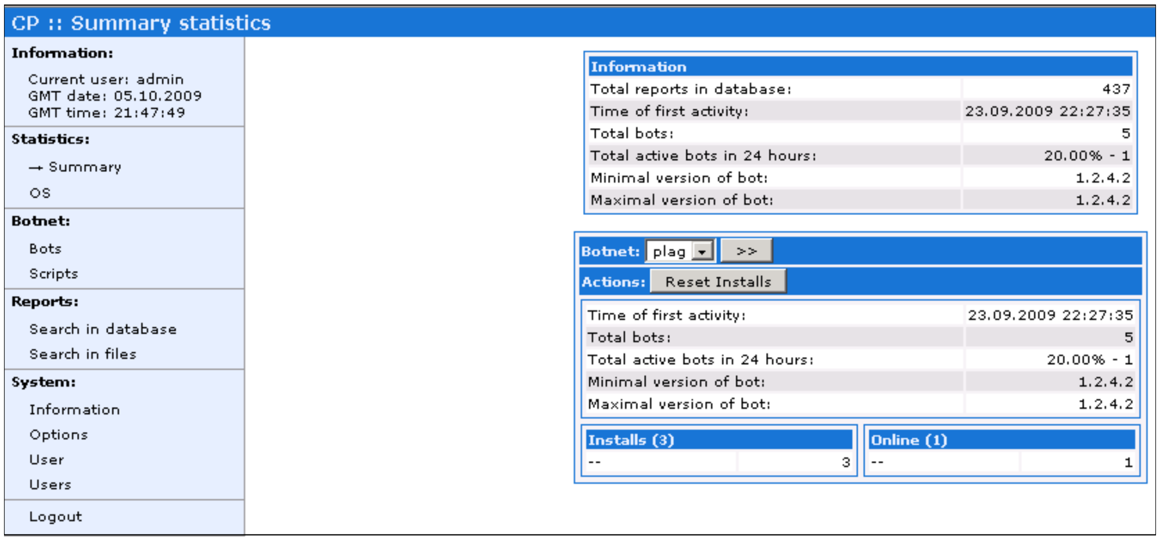
The Zeus botnet acts as a spy on the infected machine and sends credential and financial information to the attacker. The affected user is tricked into entering his banking information in fake or hijacked websites configured by the attacker. The information gathered is then sold on the black market [25].

The toolkit can be sold within the range of $800 to $4000 [51] and sometimes more depending on its capabilities.

The Zeus crimeware toolkit that we used is version 1.2.4.2, and has the following components:

1. A Control Panel (CP) (see Figure 5.3), which is installed on the attacker machine and acts as Command and Control (C& C) server for the Zeus botnet.
2. A builder program shown in Figure 5.4, which generates two files: a binary file for the botnet (**bot.exe**) and an encrypted configuration file (**config.bin**).

Two configuration files are included with the builder program and are used to configure the botnet parameters. Those files are **config.txt** and **webinjects.txt**; the former has the basic configuration information and the latter lists the targeted websites and some injection rules. The CP component is developed using PHP scripting language and a MySQL database and it is used to manage, control and collect the information from the botnets. The MySQL database is used to store the stolen information as



Current user: admin
GMT date: 05.10.2009
GMT time: 21:47:49

Information	
Total reports in database:	437
Time of first activity:	23.09.2009 22:27:35
Total bots:	5
Total active bots in 24 hours:	20.00% - 1
Minimal version of bot:	1.2.4.2
Maximal version of bot:	1.2.4.2

Time of first activity:	23.09.2009 22:27:35
Total bots:	5
Total active bots in 24 hours:	20.00% - 1
Minimal version of bot:	1.2.4.2
Maximal version of bot:	1.2.4.2

Installs (3)	
--	3

Online (1)	
--	1

Figure 5.3: Zeus Command and Control server.

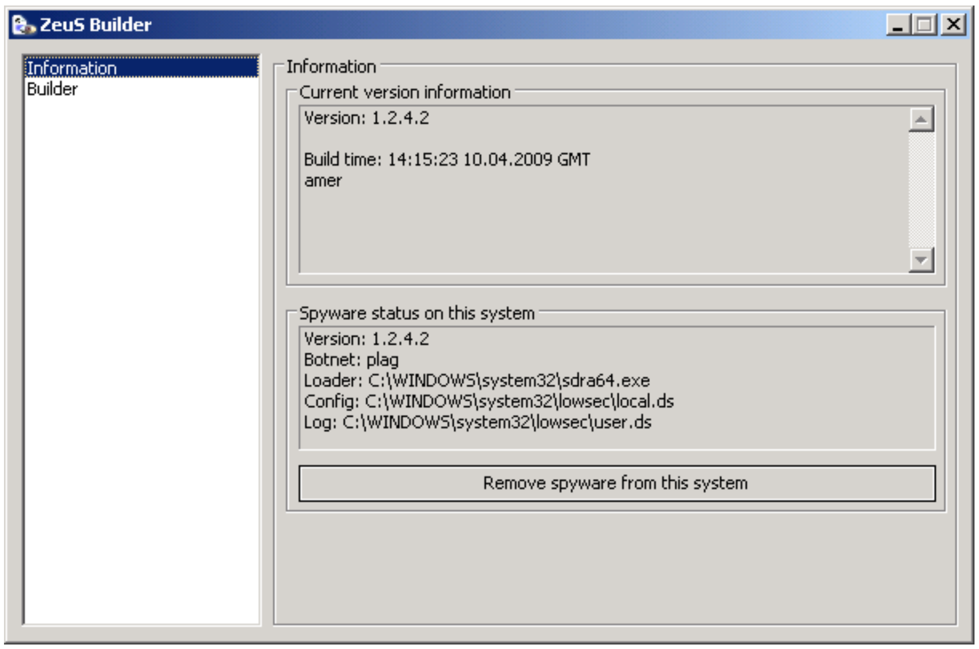
illustrated in Figure 5.5. Another important function of the CP is to communicate with the botnets (the infected machines) and to display all related information for each botnet on the MySQL database. The configuration file (**webinjects.txt**) contains a list of URLs that can be hijacked and the corresponding HTML that will be injected instead. The injection file **webinjects.txt** is customizable and the attacker can adjust it to define new rules for web injection as needed. When a machine is infected, the hijacked page is shown and it is used to steal information entered by the end user. The stolen information is encrypted with the RC4 algorithm and transferred to the C& C server where it gets decrypted and stored in the MySQL database.

5.4.3 Zeus Proactive System

Testbed System

Two machines are used as a testbed for our proactive system. The two machines are connected through a wireless switch as shown in Figure 5.6. One of the machines has a couple of Virtual Machines (VMs) running. Windows XP (service pack 2) VM was chosen to host the PSA and used for Zeus infection. The PSA does the collection and preservation as well as event detection and triggering.

The most critical system assets, such as the system32 folder, are collected and preserved. When a critical event is detected or triggered by the event triggering


ZeuS Builder
Information
Builder
Information
Current version information
Spyware status on this system
Remove spyware from this system

```
Version: 1.2.4.2

Build time: 14:15:23 10.04.2009 GMT
amer
```

```
Version: 1.2.4.2
Botnet: plag
Loader: C:\WINDOWS\system32\sdra64.exe
Config: C:\WINDOWS\system32\lowsec\local.ds
Log: C:\WINDOWS\system32\lowsec\user.ds
```

Figure 5.4: Zeus Bot Builder.

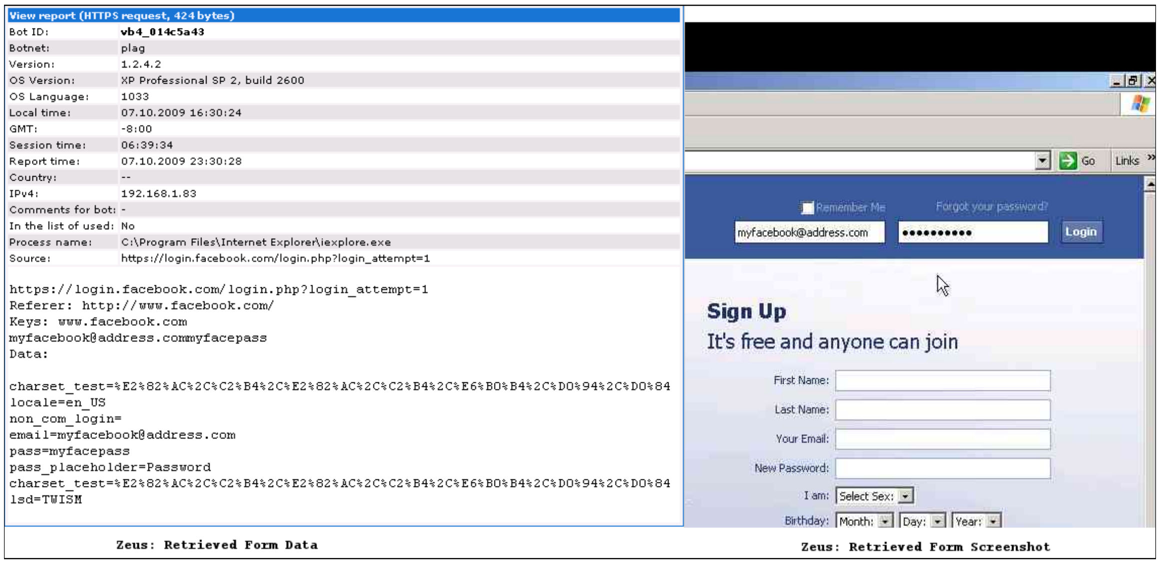
function, the remote PA is notified to carry out the analysis and generate the report. The report is sent to the investigator, via SMS and email, as well as to the PSA.

PA is a Mac OS X machine equipped with the necessary software to run the parallel implementation of the outlier detection algorithms.

5.4.4 Zeus Installation

Zeus components, the Control Panel and the Builder, were downloaded from the underground community (such as opensc.ws). They both came as separated zipped folders or sometimes under one folder depending on where it was downloaded from. The CP was installed on a Linux VM. After many trials and errors, we managed to get the proper installation work-flow by first installing the XAMPP-linux-1.7.4 package which includes Apache 2.2.17, MySQL 5.5.8, PHP 5.3.5 and other components. We then installed CP and configured its web interface to work properly. CP web navigation was intuitive and did not require considerable time.

The Builder is provided as an already made executable and was copied to the Windows XP VM. A few mouse clicks were enough to launch the Builder and generate the botnet binary, which we then used to manually infect the Windows XP machine. It was challenging to get the systems to work together, as the underground community



View report (HTTPS request, 424 bytes)

Bot ID:	**vb4_014c5a43**
Botnet:	plag
Version:	1.2.4.2
OS Version:	XP Professional SP 2, build 2600
OS Language:	1033
Local time:	07.10.2009 16:30:24
GMT:	-8:00
Session time:	06:39:34
Report time:	07.10.2009 23:30:28
Country:	--
IPv4:	192.168.1.83
Comments for bot:	-
In the list of used:	No
Process name:	C:\Program Files\Internet Explorer\iexplore.exe
Source:	https://login.facebook.com/login.php?login_attempt=1

```
https://login.facebook.com/login.php?login_attempt=1
Referer: http://www.facebook.com/
Keys: www.facebook.com
myfacebook@address.commyfacepass
Data:

charset_test=%E2%82%AC%2C%C2%B4%2C%E2%82%AC%2C%C2%B4%2C%E6%B0%B4%2C%D0%94%2C%D0%84
locale=en_US
non_com_login=
email=myfacebook@address.com
pass=myfacepass
pass_placeholder=Password
charset_test=%E2%82%AC%2C%C2%B4%2C%E2%82%AC%2C%C2%B4%2C%E6%B0%B4%2C%D0%94%2C%D0%84
lsd=TWISM
```

Figure 5.5: Zeus Stolen information with victim screen-shot from infected machine.

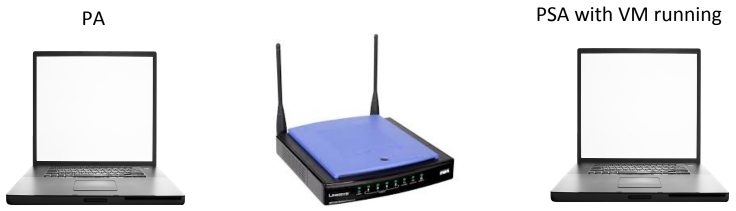


Figure 5.6: Testbed System for Zeus.

was not of much help and the manuals found were not easy to follow. Since then, more straightforward documentation on Zeus and its installation are available online; examples include [68, 72].

5.4.5 Proactive System Implementation

To test and validate our theoretical framework, we implemented a distributed system, which we described generally in Section 5.2 and Section 5.3. The implementation details are as follows:

- On the Proactive Sub-Agent (PSA) component:
  - We run the PSA component on the target system, which is connected remotely to the Proactive Agent (PA) through a secure connection (SSH).

  - PSA starts collecting and preserving important targets and events such as the System32 folder and network status. The important targets are those specified by the user. Since Zeus mainly infects the System32 folder, we used the System32 folder as the main target for our analysis.
  - Important events, which the user also specifies, are captured and used to trigger forensic rules. The main event we have for Zeus is System32 folder change.
  - When Zeus attacks the system, the System32 folder is changed. This event is associated with the forensic rule that carries out the analysis step. This forensic rule, as well as others, is added to the forensic rule engine and is ready to be triggered when System32 changes. When Zeus attacks the system, the System32 directory is changed, and the forensic rule engine starts the analysis as follows: 1) checks for any inconsistencies or known attack patterns on the hosts (PSA), 2) sends a copy of the forensic image of System32 folder to the PA, and 3) requests the PA to carry out the rest of the analysis.

- On the Proactive Agent (PA) component:
  - The PA component is up and running and waiting for the PSA component to request analysis. The PA is designed to listen to multiple PSAs but can only carry out one analysis at a time.
  - The forensic image of the important targets and events is moved to PA for further analysis. For the Zeus case, the System32 folder is the important target and it is copied to the PA when the "System32 folder change" event is triggered.
  - Outlier analysis based on Algorithm 8 with known attacks (as described in Section 4.6) embedded in the multi-resolution framework described in Section 5.3.1 is conducted on the image.
  - A final report that shows all outlier-detected files and their respected extracted and computed values is generated. It is then copied to the PSA for reference and sent to the investigator.

5.5 Summary

In this chapter, we presented a distributed and HPC proactive system as well as its implementation. Parallel versions of the outlier detection algorithms were also described and implemented. These algorithms were then incorporated, as DF reduction operators, into a multi-resolution framework that permitted us to iteratively reduce the number of false positives.

Although the performance of our implementation was encouraging (the results of the analysis were obtained in less than two minutes), more testing on large datasets is recommended. Moreover, using our algorithms [1], we were able to get better results in comparison to Carrier et al.'s work in [12]. This can be seen from the fact that we got a bigger reduction for the reduction amplitude and less evidence reduction for the evidence amplitude.

Lastly, we discussed Zeus and chose it as the perfect use case to verify and validate our proactive system.

Chapter 6

Conclusion and Future Work

Nowadays, digital data is becoming more critical than ever. Although it can take many forms and reside on many different platforms, it has to be forensically protected against sophisticated attacks. As we argued in the introduction as well as in other chapters, being proactive is the necessary step toward ensuring sound forensic investigations, especially against anti-forensic crimes, which affect the reactive investigation process in many ways: preventing evidence collection, increasing investigation time and cost, providing misleading evidence that can lead to unsuccessful or wrong prosecution, and/or preventing detection of the attack.

In this thesis, we have addressed proactive digital forensics at different levels, which we summarize below.

6.1 Proactive Digital Forensics at the Literature Level

In Chapter 2, we presented a proactive investigative system framework based on a thoroughly systematic literature review of the existing digital forensic processes. The SLR approach was followed for a couple of reasons. Firstly, SLR results are reproducible. Secondly, since all resources (databases) are queried systematically, there is less chance of missing important references. The framework presented combines two main components, proactive and reactive components, to form a self-contained digital forensic process to investigate anti-forensic attacks and promote an automation in semi-real time. Phases of the proposed process were defined and mapped to the existing investigation processes.

6.2 Proactive Digital Forensics from the First Principles

In Chapter 3, we reviewed and extended the existing five principles to proactive digital forensics. We also introduced two extra principles based on practical observations. From the seven principles, we deduced the inherent complexity and the large number of dimensions of the space in which the proactive digital forensic system resides. A theoretical foundation was proposed to formalize the implementation of the proactive system.

6.3 Proactive Digital Forensics at the Theory Level

Our theory behind the proactive digital forensics began mainly from Chapter 3 and was used to lay down the foundation of a reliable proactive system. The system was modelled as a feedback dynamical system in which the forward system is the system under investigation and the feedback system is the proactive component. An event-driven model of computation, called digital forensic rules, was used as the essential engine for the proactive component.

The targets, events and actions were found as the main elements of investigation and were reduced, classified and structured via relations, a zoning approach and space projection.

Given that the analysis is the most demanding phase, we not only classified it into three categories (signature-based, anomaly-based, and protocol-based) as done in intrusion detection systems, but we also unified the three categories using correlation on the space of profiles.

As the automated analysis is important in a proactive system, we dedicated Chapter 4 for it and formalized it based on the theoretical framework presented in Chapter 3. Two main ingredients were used: outlier analysis and information theory. From an outlier analysis perspective, we extended the iterative z algorithm to different elements of DF investigation, including events and targets. We also added local and global probability distributions to weight elements of DF investigation as required. We then generalized our probability-based algorithm to information-based iterative z algorithms. These information-based algorithms were introduced as a novel approach to express the outlier detection from an information theory perspective. Temporal and event information-based iterative z algorithms were proposed to deal with the time aspects of a proactive system as well as its events. Lastly, we combined the spatial, temporal and event information-based algorithms into a single information-based outlier detection algorithm involving the space of profiles. This algorithm was slightly modified to carry out both signature-based and anomaly-based analyses.

6.4 Proactive Digital Forensics at the Implementation Level

As the size of the investigation space is becoming large, a successful proactive tool must be able to reduce it in a systematic way without focusing on finding a specific piece of evidence and ignoring others. This investigation-oriented aspect is the main purpose behind the theory presented in Chapter 3 and Chapter 4, and the major drive for implementing different outlier detection algorithms (including the mutli-resolution one presented in Chapter 5).

In addition to being a theoretically well-founded system, the proactive system should be designed with other software engineering requirements and best practices in mind such as modularity, extensibility, reliability and decent performance. As such, we opted for an object-orientated design. The UML diagrams for the system were showing in Chapter 5. Furthermore, we discussed a distributed and HPC proactive system as well as its parallel implementation. Both serial and parallel versions of the outlier detection and analysis algorithms were verified and validated based on the dataset from the Honeypot challenge test case and compared with the work of Carrier et al. in [12]. These algorithms were then incorporated, as DF reduction operators, into a multi-resolution framework that permitted us to iteratively reduce the number of false positives. Finally, our full proactive system was verified and validated using the Zeus use case.

6.5 Future Work on Proactive Digital Forensics

In this thesis we restricted our implementation to three algorithms (1, 2 and 3) and left the rest for immediate and major future work on proactive digital forensics. Implementing and comparing the rest of the algorithms, especially the multi-attribute one, are of great importance not only to our work but also to the digital forensic community. This community will be eager to know how outlier detection algorithms are performing and how they could be enhanced using other data mining and analysis techniques.

Although the overall performance of our algorithm's implementation was promising compared to the work in [12], and our proactive outlier analysis algorithms weigh the recent elements of investigation less than the older ones, it is challenging to find the proper weighting of the probability distributions. It is even more challenging to find the right order in which to compose the outlier analysis algorithms and use them as DF reduction operators in the multi-resolution algorithm.

Moreover, in our final outlier detection analysis Algorithm 8 and in our implementation, we dealt with the unknown attacks (anomaly-based analysis) and the known ones (signature-based analysis) only, and we did not have any protocol-based analysis.

Lastly, the implementation of our proactive system lacks a Graphical User Interface (GUI) to help the users navigate, add and choose the right elements of investigation, specify the triggering events to watch for, add new forensic rules, and so on. Given that false positives were observed with the outlier detection algorithms implemented so far, enhancing our system with other searching techniques may be required. The enhancement we plan to do is to join our system with the Digital Forensics Framework software [38], a promising open-source initiative for reactive digital forensics. This will bring the GUI as well as many existing tools to the fingertips of the users of our proactive system.

Appendix A

UML Class Diagram for the Proactive Digital Forensics

The diagram shown next constitutes the full UML class diagram of the proposed Proactive Digital Forensics.

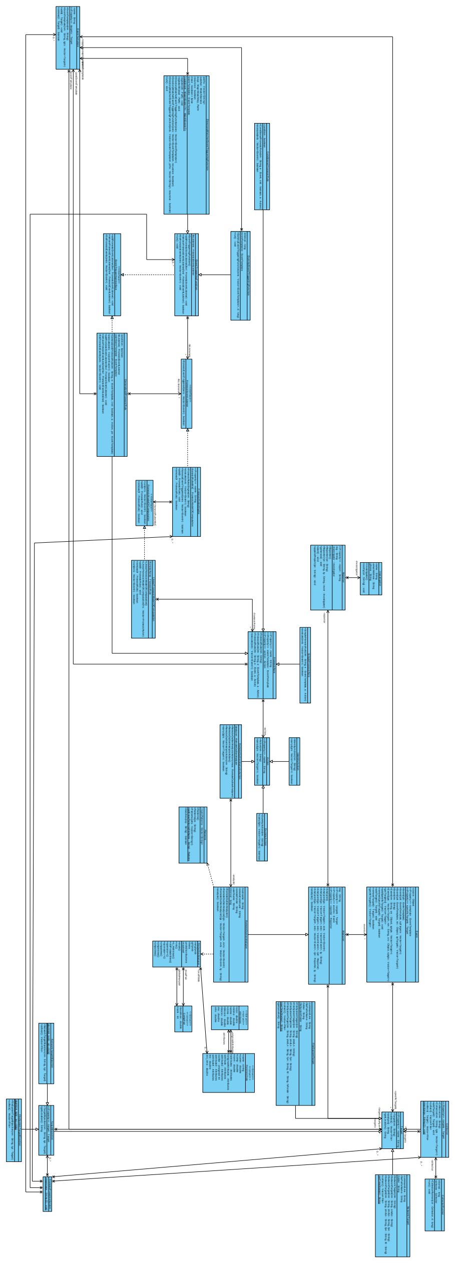


Figure A.1: Class diagram of the Proactive Digital Forensics System.

Bibliography

[1] Soltan Alharbi, Belaid Moa, Jens Weber-Jahnke, and Issa Traore. High performance proactive digital forensics. In *Journal of Physics: Conference Series*, volume 385, pages 01–15. IOP Publishing, 2012.

[2] Soltan Alharbi, Jens Weber-Jahnke, and Issa Traore. The proactive and reactive digital forensics investigation process: A systematic literature review. In *Information Security and Assurance*, pages 87–100. Springer, 2011.

[3] A.R. Arasteh, M. Debbabi, A. Sakha, and M. Saleh. Analyzing multiple logs for forensic evidence. *digital investigation*, 4:82–91, 2007.

[4] V. Baryamureeba and F. Tushabe. The enhanced digital investigation process model. In *Proceedings of the 4th Annual Digital Forensic Research Workshop, Baltimore, MD*. Citeseer, 2004.

[5] N.L. Beebe and J.G. Clark. A hierarchical, objectives-based framework for the digital investigations process. *Digital Investigation*, 2(2):147–167, 2005.

[6] D. Billard. An extended model for e-discovery operations. *Advances in Digital Forensics V*, pages 277–287, 2009.

[7] P.G. Bradford, M. Brown, J. Perdue, and B. Self. Towards proactive computer-system forensics. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, volume 2, pages 648–652. IEEE, 2004.

[8] P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571–583, 2007.

[9] B. Carrier and E.H. Spafford. Getting physical with the digital investigation process. *International Journal of Digital Evidence*, 2(2):1–20, 2003.

[10] B. Carrier and E.H. Spafford. An event-based digital forensic investigation framework. In *Digital forensic research workshop*, 2004.

[11] B.D. Carrier. *A hypothesis-based approach to digital forensic investigations*. ProQuest, 2006.

[12] B.D. Carrier and E.H. Spafford. Automated digital evidence target definition using outlier analysis and existing evidence. In *Proceedings of the 2005 Digital Forensic Research Workshop (DFRWS)*. Citeseer, 2005.

[13] B.D. Carrier and E.H. Spafford. Categories of digital investigation analysis techniques based on the computer history model. *digital investigation*, 3:121–130, 2006.

[14] S. Ciardhuáin. An extended model of cybercrime investigations. *International Journal of Digital Evidence*, 3(1):1–22, 2004.

[15] Damballa. Top-10 botnet outbreaks in 2009. `https://blog.damballa.com/archives/569/`, 2009.

[16] Federal Bureau of Investigation. Program annual report for fiscal year 2010. `http://www.rcfl.gov/downloads/documents/RCFL\_Nat\_Annual10.pdf`, 2010.

[17] F.C. Freiling and B. Schwittay. A common process model for incident response and computer forensics. In *Proceedings of Conference on IT Incident Management and IT Forensics*, 2007.

[18] S. Garfinkel. Anti-forensics: Techniques, detection and countermeasures. In *Proceedings of 2nd International Conference on Information Warfare and Security*, page 77, 2007.

[19] S.L. Garfinkel. Digital forensics research: The next 10 years. *digital investigation*, 7:S64–S73, 2010.

[20] P. Gladyshev and A. PATEL. Finite state machine analysis of a blackmail investigation. *International Journal of Digital Evidence*, 4(1):1–13, 2005.

[21] CP Grobler, CP Louwrens, and SH Von Solms. A multi-component view of digital forensics. In *Availability, Reliability, and Security, 2010. ARES'10 International Conference on*, pages 647–652. Ieee, 2010.

[22] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skoric. Towards an information-theoretic framework for analyzing intrusion detection systems. *Computer Security–ESORICS 2006*, pages 527–546, 2006.

[23] R. Hankins, T. Uehara, and J. Liu. A turing machine-based model for computer forensic reconstruction. In *Secure Software Integration and Reliability Improvement, 2009. SSIRI 2009. Third IEEE International Conference on*, pages 289–290. IEEE, 2009.

[24] W. Harrison. The digital detective: An introduction to digital forensics. *Advances in Computers*, 60:75–119, 2004.

[25] Thorsten Holz, Markus Engelberth, and Felix Freiling. *Learning more about the underground economy: A case-study of keyloggers and dropzones*. Springer, 2009.

[26] Honeynet Project. Forensic challenge. `http://www.honeynet.org/`, 2001.

[27] Philip K Hooper. The undecidability of the turing machine immortality problem. *The Journal of Symbolic Logic*, 31(2):219–234, 1966.

[28] R.S.C. Ieong. Forza–digital forensics investigation framework that incorporate legal issues. *digital investigation*, 3:29–36, 2006.

[29] J. James, P. Gladyshev, M.T. Abdullah, and Y. Zhu. Analysis of evidence using formal event reconstruction. *Digital Forensics and Cyber Crime*, pages 85–98, 2010.

[30] K. Kent, S. Chevalier, T. Grance, and H. Dang. Guide to integrating forensic techniques into incident response. *NIST Special Publication*, pages 800–86, 2006.

[31] M. Khatir, S.M. Hejazi, and E. Sneiders. Two-dimensional evidence reliability amplification process model for digital forensics. In *Digital Forensics and Incident Analysis, 2008. WDFIA'08. Third International Annual Workshop on*, pages 21–29. IEEE, 2008.

[32] J.M. Kizza. Computer crime investigations–computer forensics. *Ethical and Social Issues in the Information Age*, pages 263–276, 2010.

[33] M. Kohn, JHP Eloff, and MS Olivier. Framework for a digital forensic investigation. In *Proceedings of Information Security South Africa (ISSA) 2006 from Insight to Foresight Conference*, 2006.

[34] R. Leigland and A.W. Krings. A formalization of digital forensics. *International Journal of Digital Evidence*, 3(2):1–32, 2004.

[35] Chang-Tien Lu, Dechang Chen, and Yufeng Kou. Detecting spatial outliers with multiple attributes. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, pages 122–128. IEEE, 2003.

[36] C.T. Lu, D. Chen, and Y. Kou. Algorithms for spatial outlier detection. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 597–600. IEEE, 2003.

[37] Online. Banking malware zeus sucessfully bypasses anti-virus detection. `http://www.ecommerce-journal.com/news/18221zeusincreasinglyavoidspcsdetection`, 2010.

[38] Online. Digital forensics framework. `http://www.digital-forensic.org/`, 2013.

[39] Online. Halting problem. `http://en.wikipedia.org/wiki/Halting_problem`, 2013.

[40] Online. Intrusion prevention system. `http://en.wikipedia.org/wiki/Intrusion_prevention_system#cite_note-WhitmanMattord2009-3`, 2013.

[41] Online. Undecidable problem. `http://en.wikipedia.org/wiki/Undecidable_problem`, 2013.

[42] A. Orebaugh. Proactive forensics. *Journal of Digital Forensic Practice*, 1(1):37–41, 2006.

[43] G. Palmer. A road map for digital forensics research-report from the first digital forensics research workshop (dfrws). *Utica, New York*, 2001.

[44] S. Peisert, S. Karin, M. Bishop, and K. Marzullo. Principles-driven forensic analysis. In *Proceedings of the 2005 workshop on New security paradigms*, pages 85–93. ACM, 2005.

[45] S. Perumal. Digital forensic model based on malaysian investigation process. *IJCSNS*, 9(8):38, 2009.

[46] M.M. Pollitt. An ad hoc review of digital forensic models. In *Systematic Approaches to Digital Forensic Engineering, 2007. SADFE 2007. Second International Workshop on*, pages 43–54. IEEE, 2007.

[47] M. Reith, C. Carr, and G. Gunsch. An examination of digital forensic models. *International Journal of Digital Evidence*, 1(3):1–12, 2002.

[48] S. Rekhis and N. Boudriga. A formal approach for the reconstruction of potential attack scenarios. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1–6. IEEE, 2008.

[49] S. Rekhis and N.A. Boudriga. Visibility: a novel concept for characterising provable network digital evidences. *International journal of security and networks*, 4(4):234–245, 2009.

[50] Slim Rekhis and Noureddine Boudriga. Formal digital investigation of anti-forensic attacks. In *Systematic Approaches to Digital Forensic Engineering (SADFE), 2010 Fifth IEEE International Workshop on*, pages 33–44. IEEE, 2010.

[51] Marco Riccardi, Roberto Di Pietro, Marta Palanques, and Jorge Aguila Vila. TitansâĂŹ revenge: detecting zeus via its own flaws. *Computer Networks*, 2012.

[52] G.G. Richard III and V. Roussev. Next-generation digital forensics. *Communications of the ACM*, 49(2):76–80, 2006.

[53] M.K. Rogers, J. Goldman, R. Mislan, T. Wedge, and S. Debrota. Computer forensics field triage process model. In *Proceeding of the Conference on Digital Forensics Security and Law*, pages 27–40, 2006.

[54] R. Rowlingson. A ten step process for forensic readiness. *International Journal of Digital Evidence*, 2(3):1–28, 2004.

[55] C. Ruan and E. Huebner. Formalizing computer forensics process with uml. *Information Systems: Modeling, Development, and Integration*, pages 184–189, 2009.

[56] S.R. Selamat, R. Yusof, and S. Sahib. Mapping process of digital forensic investigation framework. *International Journal of Computer Science and Network Security*, 8(10):163–169, 2008.

[57] Y.D. Shin. New digital forensics investigation procedure model. In *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, volume 1, pages 528–531. Ieee, 2008.

[58] J. Slay, Y.C. Lin, B. Turnbull, J. Beckett, and P. Lin. Towards a formalization of digital forensics. *Advances in Digital Forensics V*, pages 37–47, 2009.

[59] Lindsay I Smith. A tutorial on principal components analysis. *Cornell University, USA*, 51:52, 2002.

[60] S.A. Soltan Alharbi, J.W.J. Jens Weber-Jahnke, and I.T. Issa Traore. The proactive and reactive digital forensics investigation process: A systematic literature review. *International Journal of Security and Its Applications*, 5(4):59–72, 2011.

[61] T. Stallard and K. Levitt. Automated analysis for digital forensic science: Semantic integrity checking. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 160–167. IEEE, 2003.

[62] P. Stephenson. Completing the post mortem investigation. *Computer Fraud & Security*, 2003(10):17–20, 2003.

[63] P. Stephenson. A comprehensive approach to digital incident investigation. *Information Security Technical Report*, 8(2):42–54, 2003.

[64] P. Stephenson. Modeling of post-incident root cause analysis. *International Journal of Digital Evidence*, 2(2):1–16, 2003.

[65] Symantec Corporation. Zeus, king of the underground crimeware toolkits. `http://www.symantec.com/connect/blogs/zeus-king-underground-crimeware-toolkits`, 2010.

[66] Ole Tange. Gnu parallel-the command-line power tool. *login: The USENIX Magazine*, pages 42–47, 2011.

[67] A. Tanner and D. Dampier. Concept mapping for digital forensic investigations. *Advances in Digital Forensics V*, pages 291–300, 2009.

[68] Shahzad Waheed. *Implementation and evaluation of a botnet analysis and detection method in a virtual environment.* PhD thesis, Edinburgh Napier University, 2012.

[69] Michael E Whitman and Herbert J Mattord. *Principles of information security.* Cengage Learning, 2010.

[70] S. Willassen. Hypothesis-based investigation of digital timestamps. *Advances in Digital Forensics IV*, pages 75–86, 2008.

[71] S.Y. Willassen. Timestamp evidence correlation by model based clock hypothesis testing. In *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop*, page 15. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[72] James Wyke. What is zeus? *Sophos, May*, 2011.