Hermes: A Targeted Fuzz Testing Framework

by

Caleb James Shortt
B.Sc., University of Victoria, 2012

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

MASTER OF SCIENCE

in the Department of Computer Science

Hermes: A Targeted Fuzz Testing Framework

by

Caleb James Shortt
B.Sc., University of Victoria, 2012

ABSTRACT

The use of security assurance cases (security cases) to provide evidence-based assurance of security properties in software is a young field in Software Engineering. A security case uses evidence to argue that a particular claim is true. For example, the highest-level claim may be that a given system is sufficiently secure, and it would include sub claims to break that *general* claim down into more granular, and tangible, items - such as evidence or other claims. Random negative testing (fuzz testing) is used as evidence to support security cases and the assurance they provide. Many current approaches apply fuzz testing to a target system for a given amount of time due to resource constraints. This may leave entire sections of code untouched [60]. These results may be used as evidence in a security case but their quality varies based on controllable variables, such as time, and uncontrollable variables, such as the random paths chosen by the fuzz testing engine.

This thesis presents Hermes, a proof-of-concept fuzz testing framework that provides improved evidence for security cases by automatically targeting problem sections in software and selectively fuzz tests them in a repeatable and timely manner. During our experiments Hermes produced results with comparable target code coverage to a full, exhaustive, fuzz test run while significantly reducing the test execution time that is associated with an exhaustive fuzz test. These results provide a targeted piece of evidence for security cases which can be audited and refined for further assurance. Hermes' design allows it to be easily attached to continuous integration frameworks where it can be executed in addition to other frameworks in a given test suite.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

my supervisor Jens Weber, for his guidance and patience.

my family, for their constant encouragement and support.

DEDICATION

For my wife Lisa.

And with this book, in noble pursuit,
I make my mark, in the most meagre way.
Year's expense, they cost me some,
Knowledge's hunt, sharpened mind come.
In the hope to push, and bend the bounds,
of what is known, and of what confounds.

I know one thing: that I know nothing
(Arguably) Socrates

Chapter 1

Introduction

1.1 Security Assurance

1.1.1 Assurance and Assurance Cases

“Assurance is confidence that an entity meets its requirements based on evidence provided by the application of assurance techniques” [4]. *Security assurance* simply narrows this scope to only include security claims or requirements. Security assurance cases (security cases) are used to argue the assurance of specific claims. They provide a series of evidence-argument-claim structures that, when combined, provide assurance on the original claim.

For example, the claim “the REST API is secure against attack” cannot be proven directly, but all claims require evidence to provide assurance that it is true, therefore the main claim will have a series of *subclaims* that must be assured to assure the main claim. Once the subclaims are assured, the main claim is assured. These subclaims would include statements such as “The REST API is resistant against attack from bots”. This subclaim would require either more subclaims that must be assured, or it must provide evidence that assures that the claim is true. Evidence provides the “hard facts” that support a claim. Each hierarchy of claims and subclaims must have evidence supporting it.

Security assurance, and security cases, provide a structured method of documenting the claims and evidence that support each claim. This simplifies the auditing process and helps both the developers and QA team with directing their efforts. Security cases provide the assurance, and thus the confidence, that the given entity meets its requirements.

1. **Claim:** The REST API is secure against attack.
   - (a) **Subclaim:** The REST API is resistant against bot (automated) attacks.
     - i. **Evidence:** Updating procedures implemented on a weekly basis for libraries. Software will be continually updated weekly. (Ref Document)
     - ii. **Evidence:** Fuzz tested API. (Ref Document)
     - iii. **...**
   - (b) **Subclaim:** The REST API is resistant against hacker (human) attack.
     - i. **Evidence:** Hired a pen-tester to attempt to compromise the API. (Ref Document)
     - ii. **Evidence:** Followed the principle of least privilege. (Ref Document)
     - iii. **...**
   - (c) **...**

Figure 1.1: Security Cases: An Incomplete List of Claims, Subclaims, and Evidence

1.1.2 Evidence

Assurance cases rely on evidence to support their claims, therefore it is important that evidence is gathered properly and expressed clearly. Kelly and Weaver have introduced "Goal Structured Notation" (GSN) [58] as a method to express assurance cases. It includes the relationships, claims, evidence, and the context required to interpret the evidence with respect to the associated claim. GSN is a graphical representation of the assurance case. This graphical method of claim-argument-evidence expresses the evidence in a clear and intuitive manner, and as such, it has gained a strong following in the assurance case community.

The types of evidence vary widely from company to company, however some common types of evidence for security cases include black-box testing results, white-box testing results, state machines (to prove that certain paths are impossible), standards compliance check lists, fuzz test results, and penetration-test reports. Evidence is produced using the testing techniques implemented and executed. The associated evidence and gathering techniques are displayed in Table 1.1

The evidence gathering techniques *produce* the evidence. They may not be evidence themselves.

Technique	**Evidence**
Black-Box Testing (General)	Results from black-box testing
White-Box Testing (General)	Results from white-box testing
Standard Compliance Audit	Whitepaper or standards compliance check list
Fuzz Testing	Fuzz testing results
Penetration Testing	Penetration test report

Table 1.1: Gathering Techniques and their Associated Evidence

1.2 Problem

Software assurance cases provide “a basis for justifiable confidence in a required property of software” [39]. If weak evidence is used for assurance the confidence is not justifiable. Therefore, depending on the evidence provided, a security case can reduce uncertainty and provide justifiable confidence, or it can provide a lack of confidence in the system [39].

Due to the nature of argumentation in a security case, the evidence may be either qualitative or quantitative. This means that there exists a measure of subjectivity in their evaluation and interpretation [58]. The goal is to move from an “inductive argumentation” approach, where “the conclusion follows from the premises not with necessity but only with probability”, to the far stronger “deductive argumentative” approach, where “if premises are true, then the conclusion must also be true” [58].

“Traditional fuzz testing tools” encounter difficulties, and become ineffective, if most generated inputs are rejected early in the execution of the target program [91]. This is common in undirected fuzz testing and can lead to a significant lack of overall code coverage. In fact, it is “well-known that random testing usually provides low code coverage” [37]. In addition to poor code coverage, undirected fuzz testing performs poorly, overall, in application [15]. Therefore the undirected fuzz testing approach can hardly be considered sufficient for the assurance of a security case.

Directed fuzz testing approaches exist and include solutions that rely on taint analysis to provide a certain level of introspection [33, 91]. Taint analysis relies on a “clean” run to provide a baseline execution pattern for the target application. It then compares all subsequent executions to the baseline in an attempt to find discrepancies. Further approaches include the addition of symbolic execution and constraint solvers to take full advantage of the introspective properties of the taint analysis approach

[15]. These “whitebox” approaches are complex and costly in time [94]. Additionally, it is an open question if symbolic execution fuzz testing can consistently achieve high code coverage on “real world” applications [15], and the symbolic execution “is limited in practice by the imprecision of static analysis and theorem provers” [37]. Finally, fuzz testing is executed for a certain amount of time to be considered “good enough”. However “good enough” is a subjective term and lacks the quantitative properties required to be reviewed as evidence. These approaches are certainly an improvement over undirected fuzz testing in the quality of evidence provided, but the issues of performance, complexity, and uncertainty of application in “real world” systems leave much to be desired.

1.3 Desired Features for a Solution

Figure 1.2 states the requirements that our solution must meet to address the problem stated in section 1.2.

1. Must be effective in pinpointing specific types of defects in the target software
2. Must provide a repeatable and reviewable approach to fuzz testing for the purposes of enhancing security case evidence.
3. Must achieve code coverage parity, with existing methods, on target areas when executed under timing constraints.
4. Must integrate into test framework easily.

Figure 1.2: Goals for a Desired Solution

1.4 Thesis Statement

We have shown that current fuzz testing methods can be improved to provide more quantitative and reviewable evidence for security assurance cases. Current white-box fuzz testing approaches perform poorly in practice and are complex. We have identified some aspects of static analysis, genetic algorithms, and dynamic protocol generation, that may provide a more targeted fuzz testing platform for an improved security assurance case.

The goal of this thesis is to answer the question: **Is it possible to use targeted fuzz testing as evidence for security assurance cases to reduce the computation time required while achieving the same code coverage as a full fuzz test run?**

1.5 Thesis Outline

In this chapter, we briefly introduced security assurance, assurance cases, and how evidence is gathered to provide proof of assurance. We explained the problem with the current evidence provided by fuzz testing and listed the desired features for a solution. Chapter 2 introduces security assurance cases in a more complete manner and also reviews the models and metrics, used for software quality and security, that provide the basis for security assurance cases. We continue by reviewing static and dynamic analysis, and conclude by looking at possible optimization approaches for fuzz testing. Chapter 3 describes our solution from a high-level perspective. Chapter 4 outlines the reasoning for our implementation decisions and provides a more in-depth view of our solution. Chapter 5 describes the evaluation approach for our solution and analyses the results produced. Chapter 6 introduces future research avenues to pursue based on our solution and Chapter 7 summarizes our findings.

Chapter 2

Related Work

2.1 Security Assurance Cases

The use of security assurance cases is a relatively new development in Software Engineering. Previously, the tendency was to argue that a given piece of software was secure by saying that it followed certain guidelines, best practises, or standards, or by executing a battery of tests that exercise the software within certain limits, or, in some cases, hiring professionals to attack the system and ensure that it can reasonably withstand an intelligent adversary. These methods are effective in their respective modes and produce valuable information on the target system, however there is no structured way to gather the "evidence" from the variety of tests executed and organize them in a way that could provide a compelling, and coherent, argument for the security of the system –not just to internal parties but also to external auditors and review entities. This problem is solved by the security assurance case.

"An assurance case is a body of evidence organized into an argument demonstrating that some claim about the system holds, i.e., is assured" [42]. In the context of a security assurance case, or security case, the "claim" would be a statement such as "the system is adequately secure". This claim would then be supported by subclaims, or evidence, that are linked by arguments and that argue that the specified claim, or evidence, supports the higher level claim –much like a legal case. Security cases evolved from safety assurance cases in the automotive, aerospace, medical, defence, and nuclear industries. These industries include safety-critical systems and, in light of "several significant catastrophes", safety standards were introduced to regulate and verify safety properties such as reliability and fault tolerance. Safety cases were created to address these new standards in a structured way [39]. In light of the apparent success of safety cases, security practitioners began to incorporate these methodologies into their own work and modified the safety cases to include security properties for assurance.

2.1.1 Security Case Components

There are three main components to a security case: claims, arguments, and evidence. Security practitioners can construct elaborate and extremely complex security cases using these components as building blocks [3].

Claims are true or false statements that are to be proven by the connected arguments or evidence [3]. The highest-level claim is usually accompanied by a justification for choosing that claim. This allows readers to follow the thought process of the creator of the security case more thoroughly. Claims can be used as evidence (referred to as an assumption) or as avenues to further argue the security case [3].

Arguments are used to connect claims to higher-level claims, or evidence to claims, in a coherent manner. Arguments provide the reasoning as to why the claim has been adequately met and why the reader should believe the claim [42].

Evidence is either an assumption (a claim with no further support), some piece of data, test results, or even an additional assurance case [3]. It is the "material" that, given the argumentative context, supports a given claim. Evidence is the lowest level of an assurance case, and in proper security cases most, if not all, leaves of the security case "tree" are evidence. There are varying degrees of quantity, quality, context and confidence in evidence. It is left to the practitioner to argue convincingly that the evidence provided adequately supports the given claim [67, 20, 3, 4, 90, 81, 98, 14, 39, 68, 42, 61].

2.1.2 Goal Structured Notation

Assurance cases can become extremely complex and include vast amounts of data, arguments, and references, and it can lead to an overwhelming "mountain" of data [45, 67, 59, 20, 39, 61]. It became pertinent to create a standardized method and notation to express assurance cases and help alleviate the risk of becoming bogged down with data. Goal Structured Notation (GSN) [58] is an assurance case notation that is able to express complex cases with simple building blocks that represent claims, arguments, and evidence.

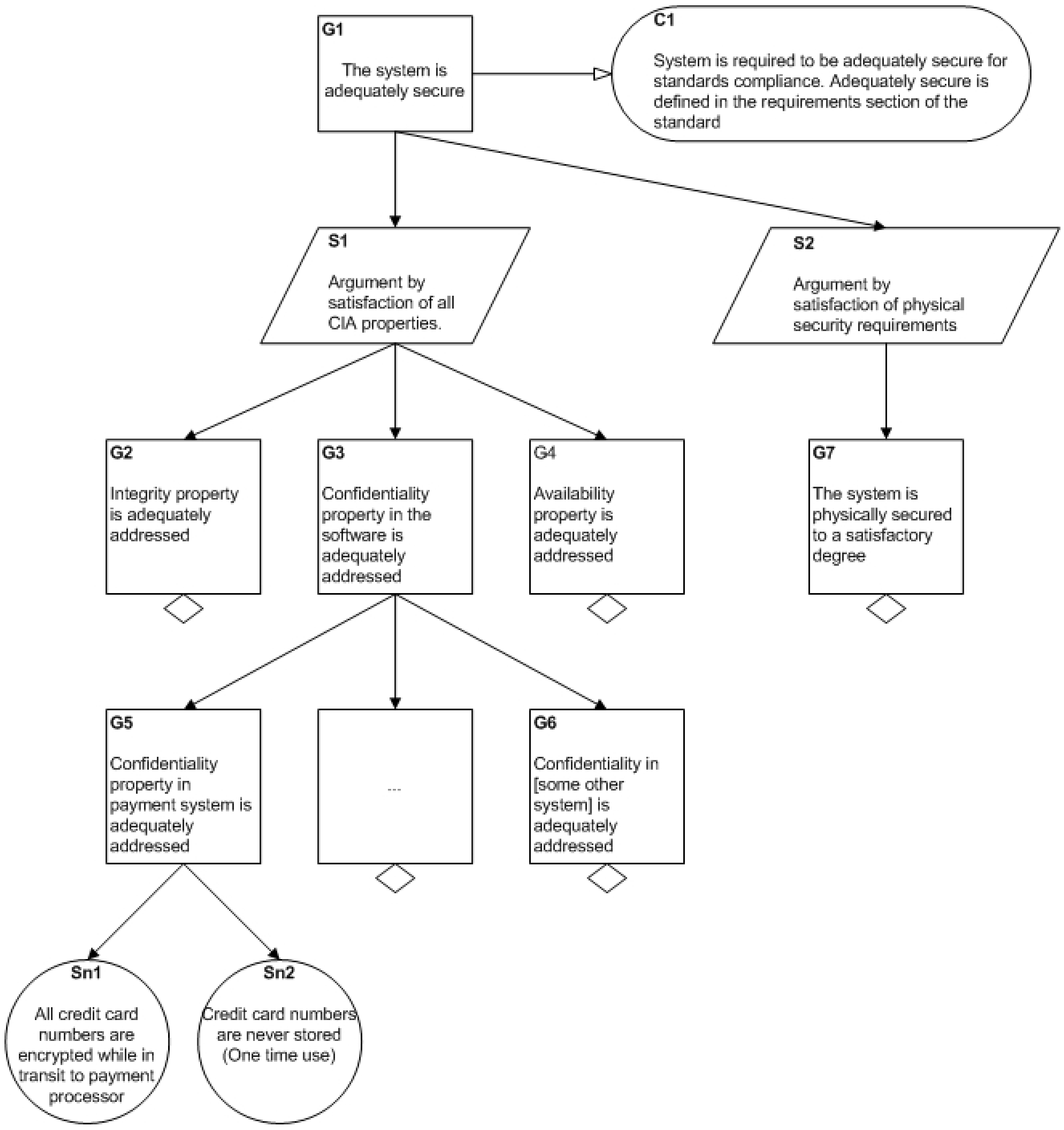
G1
The system is adequately secure
C1
System is required to be adequately secure for standards compliance. Adequately secure is defined in the requirements section of the standard
S1
Argument by satisfaction of all CIA properties.
S2
Argument by satisfaction of physical security requirements
G2
Integrity property is adequately addressed
G3
Confidentiality property in the software is adequately addressed
G4
Availability property is adequately addressed
G7
The system is physically secured to a satisfactory degree
G5
Confidentiality property in payment system is adequately addressed
...
G6
Confidentiality in [some other system] is adequately addressed
Sn1
All credit card numbers are encrypted while in transit to payment processor
Sn2
Credit card numbers are never stored (One time use)

Figure 2.1: A partial security case in Goal Structured Notation (GSN)

Figure 2.1 details an incomplete security case in GSN. Claims are represented by rectangles labelled G1– Gn and are considered "goals" in GSN. Claims, or goals, that are underdeveloped and require further elaboration have a small diamond below them. This can be seen in Figure 2.1 with G2, G4, G6, and G7. Sometimes context is required to justify top-level claims or to simply provide additional information on the given claim. There is a context object labelled C1 in this example. Arguments, or strategies in GSN, are represented with parallelograms. The reader can see arguments in the provided example. They are labelled S1 and S2. Evidence is represented as circles and are labelled Sn1 and Sn3. These building blocks allow for complex and extensive assurance cases to be created and communicated in a standardized manner.

In the example shown in Figure 2.1, we are attempting to show that a system is "adequately" secure. Although much of the security case is incomplete, there is a path of evidence to support claim "G5". If we work from the evidence and move up we see that the two pieces of gathered evidence, Sn1 and Sn2, directly support the claim that confidentiality in the payment system is "adequately" addressed via encryption of credit card numbers while in transit and in the policy to never store credit card numbers. Now this evidence may not support the claim G5 satisfactorily, and if an auditor was to review this security case they may agree, but it was laid out in a manner that was easily understood by the auditor and they were able to make a decision as to whether the claim was supported to their satisfaction.

If we follow the security case up from claim G5 we see that it is part of a larger claim – the claim that confidentiality is addressed in the software. This is how security cases work: they follow a tree-like structure where the objects of one level support the objects of a higher level until there is only a single object supported by a host of claims and evidence that is easily traversed. GSN facilitates the display of the security case "tree".

2.1.3 Limitations, Direction & Future Work

Limitations in security cases, and assurance cases as well, have been exposed in various published works. Concerns over methodologies and proper treatment of information have been raised. These limitations in themselves may suggest future work in the field.

The frameworks used for constructing and evaluating security cases have been found to focus too heavily on the final structure of the assurance case and too little on "how to identify, collect, merge, and analyze technical evidence" [8]. Additionally, security cases produce a vast amount of evidence. This evidence varies in quantity, quality, subjectivity and confidence [67, 20, 3, 4], and there is a risk of key pieces of evidence being overlooked due to the sheer amount of data and analysis required. This results in "squandered diagnostic resources" as the approach thus far seems to be to "cast a wide net" [8, 67].

Security case creation can be extremely time consuming, expensive and can consume vast amounts of resources if implemented incorrectly. This is mainly due to the increased complexity of software, evidence generation, and organization [45, 67, 59, 20, 61]. There is a call for security cases to be "designed in" at an early stage in the SDLC and for practitioners to proactively maintain the security case throughout the SDLC. Otherwise evidence is lost and the security case becomes brittle and maintainable [45, 76, 67, 59, 3, 4, 90, 81, 78, 39, 42, 61].

There is a strong demand for further tool development for security case support and implementation [8, 90, 98, 78, 39, 68, 42, 61]. This is due to the vast amount of data and resources required to properly construct a security case and maintain it.

Additionally, metrics pertaining to the confidence of evidence in security cases are greatly needed [59, 77, 90]. With proper confidence metrics, an accurate measure of the support each claim holds could be expressed. This would further enhance the ability to analyze security case strength externally and internally.

Further development in methodologies is needed, in general, for security cases to gain traction in larger systems [30, 77, 98, 78, 68, 61]. This is apparent in that there is little research on security cases for large and complex systems [59, 90].

2.2 Fuzz Testing

Fuzz testing is an automated, or semi-automated, type of random, or semi-random, negative testing method that attempts to cause a target system to crash, hang, or otherwise fail in an inelegant manner [38, 88, 87, 69]. It takes a dynamic analysis approach and tracks the attempted input and the resulting response from the system – whether or not it fails and, in some cases, includes the type of failure. In essence, it is a black-box scattergun approach where the accuracy of the "scattergun" is determined by the fuzzer utilized.

Fuzz testing has proved to be a valuable addition to current software security techniques and has caught the attention of industry leaders such as Miscrosoft who have incorporated it into the Security Development Lifecycle (SDL) [47, 63]. It is particularly well-suited to discover finite-state machine edge cases via semi-malformed inputs [87, 22]. The partially-correct inputs are able to penetrate the initial layers of verification in a system and test the bounds of areas that may have not been considered by the developers or design team. These partially-correct inputs can be generated from inputs provided to the fuzzer at runtime where it uses it as a template, or they can be "mutated" from capturing input information that is known to be correct. These two methods define the two categories of fuzzers: "Mutation-based" and "generation-based" [87, 22].

Fuzzing is able to discover a variety of security vulnerabilities and defects including crashes, denial of service vulnerabilities, security weaknesses (buffer and integer overflows, parse errors, etc), and performance issues [88, 22, 70, 31]. If is important to note that fuzzing's primary purpose is to find bugs, and not all bugs will result in security vulnerabilities [19].

2.2.1 Types of Fuzzers

There are two large categories of fuzz testers: mutation-based and generation-based fuzzers. These categories define the process in which the inputs are created for a particular target.

Generation-Based Fuzzers

The generation-based fuzzer uses random or brute-force input creation. For this reason, generation-based fuzzers are more specific to a particular protocol or application as the inputs have to be tailored to each specific use, but once it is set up it is relatively simple to execute. Once the fuzzer is connected to the target it can generate its inputs and track the responses returned [22].

The naive brute-force generation fuzzer, which contains no prior knowledge of the target, must generate the entirety of the attack space to be effective. This is an extremely inefficient method and would require immense amounts of time and processing power to complete. [18, 87, 22].

The random generation method involves randomly generating inputs to be given to the target. This may seem like an improvement over the naive brute-force method however attack surface coverage is sacrificed [18]. For example, if there are 2^n possible inputs, and only one input has a failure case, the random generator has to run a worst-case of 2^n times (if duplicates are forbidden) – the same as if we were to execute the brute-force method.

With these limitations in mind, generation-based fuzzers are inefficient and ineffective for larger software systems where the attack space is significant. However, they provide a valuable "baseline" that can be used as a crude metric of robustness. The question of how long the fuzzer is allowed to run still remains however.

Mutation-Based Fuzzers

The mutation-based fuzzer attempts to improve upon the basis of the generation-based fuzzer. Sometimes referred to "intelligent" fuzzers [75], a mutation-based fuzzer takes into account the limitations of its predecessor by capturing valid input, or being provided an input template, that it can use to generate semi-valid inputs. These may even include checksum calculation and other more advanced methods to circumvent the primary level of verification and follow unintended attack paths [75]. Mutation-based fuzzers are considered "generic fuzzers" as they are capable of fuzzing multiple protocols or applications [18]. This can be achieved by utilizing a "block-based approach" which uses "blocks" of information to construct protocols or data structures [19]. Each block can either be fuzzed or left in its original state. This maintains the shape of the datastructure but allows for fuzzing. With a block-based approach, additional information blocks can be created and reused to construct various protocol definitions, file formats, or validation techniques such as checksums [6].

Mutation-based fuzzers will provide inputs that are capable of passing the basic validation checks of the target. This increases efficiency as the fuzzer is not generating "frivolous" inputs known to be rejected by the most basic validation. Additionally, this improves code coverage and the chance of detecting catastrophic defects in the system.

Specialized Fuzzers Specialized fuzzers can be created for a particular target such as web applications, files, protocols, APIs, or networks. These specialized fuzzers can be in the form of a stand-alone fuzzer application or a fuzzing framework [57]. The benefit of specialized fuzzers is that they bring intimate knowledge of the target protocol, file format, or API. A fuzzer is only as good as the person who wrote it, and if an expert in a particular format wrote the fuzzer it is more likely to detect defects. In the end, however, the effectiveness of the fuzzer still falls on the skills of the person who wrote it [57].

2.2.2 A Fuzzer Example: Sulley

Sulley is an open-source, block-based protocol fuzzing framework written in the Python language [7, 88, 87]. It is a framework and as such it does not provide immediate "plug and play" functionality. It requires the user to develop the protocol definition themselves or create a session for the framework to listen to. Sulley provides a variety of tools for the developer to create their own fuzzer such as target health monitoring, target reboot functionality, logging, and fault identification. It comes with a variety of protocol definitions that can be used by the developer once they have written the fuzzer.

```
 1  s_initialize("HTTP Requests")
 2  s_group("verbs", values=["GET", "HEAD", "POST", "PUT"])
 3  if s_block_start("body", group="verbs"):
 4          s_delim(" ")
 5          s_delim("/")
 6          s_string("index.html")
 7          s_delim(" ")
 8          s_string("HTTP")
 9          s_delim("/")
10          s_string("1")
11          s_delim(".")
12          s_string("1")
13          s_static("\r\n\r\n")
14          s_block_end()
```

Figure 2.2: An example HTTP protocol definition in Sulley

Figure 2.2 displays a basic representation of HTTP in Sulley's protocol definition language and includes only the GET, HEAD, POST, and PUT commands. It begins with the initialization of the protocol. The "s_initialize" command takes the label of the protocol you are to create as a parameter – this will be referenced in the fuzzer logic. In Sulley, groups can be created. Line 2 of Figure 2.2 is defining a set of HTTP verbs to be used in the protocol. Once the verbs are defined the block that will define the HTTP protocol can begin. It takes the verbs as a parameter and will prepend a single value from the set of verbs to the block. In the block is the structure definition of HTTP. The function "s_delim" is used to specify a delimiter and tells Sulley to leave it as it maintains the correct structure of the protocol. The "s_string" function tells Sulley that this is an area that can be safely fuzzed without compromising the structure of the protocol. The "s_static" function tells Sulley that the given string is required for proper protocol structure.

Once the protocol definition has been created the developer must write the fuzzer itself. For an HTTP fuzzer to execute it must create and configure a session object for monitoring both processes and the network, add the target to the session, connect to the target, specify the protocol to fuzz, and finally start the fuzzing process.

```
 1  from sulley import *
 2  import HTTPProtocolDefinition
 3
 4  sess = sessions.session(session_filename="http_test.session")
 5
 6  myip = "localhost"
 7  target = sessions.target(myip, 8080)
 8  target.netmon = pedrpc.client(myip, 26001)
 9  target.procmon = pedrpc.client(myip, 26002)
10
11  sess.add_target(target)
12  sess.connect(s_get("HTTP Requests"))
13
14  sess.fuzz()
```

Figure 2.3: An example HTTP fuzzer using the Sulley framework

Figure 2.3 uses the HTTP protocol definition that was specified in Figure 2.2. Line 2 imports that definition, now named “HTTPProtocolDefinition”, to be referenced on line 9. In this case, the fuzzer is targeting the localhost port 8080 – usually used by Apache Tomcat.

The result of each attempt is stored as a PCAP file and saved to disk. A large number of PCAP files will fill the destination folder due to the large amount of attempts. Each PCAP file stores the given input to the target and the response returned to the framework. This allows the developers to trace exactly what was given to the target to cause a given failure. In addition to the PCAP files stored, Sulley tracks a session with the target so that, if the fuzzing process is stopped for any reason, it can be restarted at the exact point where it was stopped. Logs for the process monitor, the network monitor, and the fuzzer itself are also generated to further refine the results generated.

2.2.3 Limitations, Direction & Future Work

The limitations of fuzzing include high computational loads and its time-intensive nature. It is inefficient and not possible to attempt every entry in the attack space as this would require an inordinate amount of time. Therefore, it is imperative that fuzz testers develop and identify an "effective input space rather than the absolute input space" [18]. Fuzzing requires knowledge of the protocol or target that is to be fuzzed. This means that the developer must have a working knowledge of the target protocol to be effective [75]. Additionally, fuzzing is limited in its capacity to test encrypted formats. This may be mitigated by giving the fuzzer the ability to decrypt the data before fuzzing it, but it severely limits its black-box testing ability. Anything more than bit flipping will be detected by the parser [75].

Open questions for fuzzing include concerns about metrics that are able to accurately measure the "quality" of a fuzzer for comparison of other fuzzers, and the question of how long should a fuzzer be run [22]. These questions point to the issue of a general lack of metrics for fuzzing.

Fuzzing has been incorporated into many software security methodologies including Microsoft's Security Development Lifecycle (SDLC) [47, 63]. These methodologies are combining fuzz testing with additional tools such as coverage tools and white-box testing tools to further improve the effectiveness of the test results. Additional research has been focused on reducing the need for a working knowledge of the target protocol or format before starting. This is called zero-knowledge fuzzing [53].

Overall, fuzz testing is a powerful technique that assists developers and testers in discovering, and correcting, software defects. It also is a useful method to verify third-party software before adding it to another system.

2.3 Models & Metrics: Quality & Security

Metrics allow practitioners to establish a baseline of quality, and continued measurement provides a record of continuous improvement [74]. This gives a view of where the system is and where it may need to go to achieve its quality assurance goals. Software metrics support factors that allow the practitioner to make claims about the given system. These factors include concepts such as reliability, usability, maintainability, and robustness. Metrics provide the building blocks and initial values to measure the performance of those factors and to track their performance over time. Within the general framework of software quality metrics is the focus in software security metrics. Both of these families have similar but different goals when analyzing a system.

2.3.1 Software Quality Models

Naik and Tripathy refer to Garvin's work [35] in the perception of quality including his interpretation of the five views of software quality: Transcendental, User, Manufacturing, Product, and Value-Based View [74].

The transcendental view involves the "feeling" of quality, usually through experience. The user view is the view from the perspective of the user, and involves qualities such as reliability, functionality, and usability. The manufacturing view involves the viewpoint of the manufacturer, and is concerned with adherence to specifications. The product view concerns itself with the internal product properties that produce external product properties that exude quality. The value-based view concerns itself with the convergence of "excellence and worth", and the value of a certain level of quality [74]. These five views reflect the stakeholders of software quality and their perceptions of what features make up quality.

Metrics, solely or in conjunction with other metrics, are used to measure and evaluate software quality attributes such as efficiency, complexity, understandability, reusability, tesability, and maintainability [79, 74, 65]. These quality attributes provide a general "health meter" for the system in terms of quality and represent behavioural characteristics of the system [74, 65].

McCall et al. [65, 16, 74] describe a variety of quality factors. They are listed in Table 2.1.

Some of the quality factors in Table 2.1 will be of the highest importance to some systems while in other systems they may be the lowest. If a system is designed specifically for a particular piece of hardware, and will not be used on any other platform, it does not make sense to make portability a top priority. Resources are limited and deadlines are looming. But the question remains, how does one measure a quality factor such as "integrity"? It is a relatively abstract concept that requires an additional information to define and measure. McCall et al [65, 74] acknowledged this and created the quality criteria.

The quality criteria is a more granular and measurable view of the quality factors specified in Table 2.1. They can be measured using metrics, and each quality criterion will have a positive impact on one or more quality factors [74]. Naik and Tripathy's table (17.3) [74], seen here in Table 2.2, is extracted from McCall's descriptions of quality criteria [65].

Now that measurable quality criteria have been defined, quality assurance prac-

Quality Factors	Description
Correctness	How close a system comes to meeting the specifications
Maintainability	How much effort is required to fix a defect in a running system
Reliability	How long a system will run before it fails to provide a given precision
Flexibility	How hard it is to modify a running system
Testability	How hard it is to test a system
Portability	How hard it is to transfer a system from one platform to another
Reusability	How much of the system can be used in other systems
Efficiency	How much code and resources required for the system to run
Usability	How much effort required to learn and run the system
Integrity	Amount, and degree, of measures utilized to restrict access to the system and its data
Interoperability	How hard it is to couple an additional system to a system

Table 2.1: Table of McCall's quality factors [65]

titioners can measure quality factors. Each criterion is associated with one or more quality factors to which it supports. For example, "correctness" is supported by traceability, completeness, and consistency. Additionally, consistency also supports "reliability" and "maintainability" [65]. Metrics are utilized to measure the quality criteria defined above, or can capture some aspect of a quality criterion [74].

Table 2.3 displays the relationship between McCall's quality factors and their supporting quality criteria. It is with these tools that practitioners are able to measure quality. However, it is left up to the practitioner to actually measure the criteria in a meaningful and accurate manner. This may lead to other problems such as improper use of metrics and false measures of quality.

ISO Quality Standards

In addition to McCall's quality model is the international standard ISO 9126 which is broken up into four parts: 9126-1 to 9126-4. ISO 9126-1 details a similar structure to McCall's model with six quality characteristics: Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability [48]. In 2011, ISO 25010 was released

Quality Criteria	Description
Access Audit	How easy it is to audit the system for standards compliance
Access Control	What protects unauthorized access to the system and its data.
Accuracy	How precise computation and outputs are
Communication Commonality	How much are standard protocols and interfaces used?
Completeness	How much of the required functionality has been implemented
Communicativeness	Ease with which inputs and outputs can be assimilated
Conciseness	How compact is the source code (Lines of code)
Consistency	How uniform is the design, notation, and implementation of the system
Data Commonality	Does the system use standard data representations?
Error Tolerance	What is the degree of assurance that the system will continue to run in the event of an error
Execution Efficiency	What is the runtime efficiency of the system?
Expandability	How much can storage and functions be expanded?
Generality	What are the potential applications of the system's components?
Hardware Independence	How much does the system rely on the underlying platform?
Instrumentation	Does the system provide the ability to measure operation, use, and errors?
Modularity	Are the modules of the system highly independent?
Operability	How easy is it to operate the system?
Self-Documentation	How much inline documentation that explains implementation is there?
Simplicity	How easy is it to understand the software
Software System Independence	How independent is the software from its software environment?
Software Efficiency	What is the runtime storage requirements of the system?
Traceability	How easy is it to link software components to requirements?
Training	How easy is it for new users to learn and use the system?

Table 2.2: Table of McCall's quality criteria [74, 65]

Quality Factors	**Supporting Quality Criteria**
Correctness	Traceability, Completeness, Consistency
Reliability	Consistency, Accuracy, Error Tolerance, Simplicity
Efficiency	Execution Efficiency, Storage Efficiency
Integrity	Access Control, Access Audit
Usability	Operability, Training, Communicativeness
Maintainability	Consistency, Simplicity, Conciseness, Self Descriptiveness, Modularity
Testability	Simplicity, Instrumentation, Self Descriptiveness, Modularity
Flexibility	Self Descriptiveness, Expandability, Generality, Modularity
Portability	Self Descriptiveness, Modularity, Software-System Independence, Machine Independence
Reusability	Self Descriptiveness, Generality, Modularity, Software-System Independence, Machine Independence
Interoperability	Modularity, Communications Commonality, Data Commonality

Table 2.3: Table of the relationships of McCall's quality criteria and quality factors [74, 65]

which replaced ISO 9126-1 and named eight quality characteristics as opposed to the six ISO 9126: Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability [50].

Quality Characteristic	**Subcharacteristics**
Functionality	Suitability, Accuracy, Interoperability
Reliability	Maturity, Fault Tolerance, Recoverability
Usability	Understandability, Learnability, Operability
Efficiency	Time Behaviour, Resource Behaviour
Maintainability	Analyzability, Changeability, Stability, Testability
Portability	Adaptability, Installability, Conformance, Replacability

Table 2.4: ISO 9126-1 quality characteristics and their associated subcharacteristics [48]

In both standards, the quality characteristics have subcharacteristics that support them. This support structure is similar to McCall's quality model structure. One key addition in the ISO standards, both 9126 and 25010, is security [48, 50]. In McCall's quality model, security is not inherently addressed. The integrity quality factor suggests some notion, supported by the two criteria "access control" and "access audit", but it is severely restricted in addressing all of the security properties, such as confidentiality, integrity, availability, and non-repudiation, satisfactorily [65].

2.3.2 Software Quality Metrics

The ultimate goal of a software quality metric is to support a quality factor or quality characteristic. This is achieved by supporting subcharacteristics or quality criteria that directly relate to a quality metric. If we look at the quality factor "correctness", we see that one of the supporting quality criteria is "completeness". From Table 2.2 we see that the definition of "completeness" is "How much of the required functionality has been implemented". The practitioner, who is looking to develop a metric to measure this criterion, may count how many features there are, and include all of the functionality required, and then count how many of these required features have been implemented. The resulting ratio, or percentage, will give the practitioner an idea as to how "complete" the software is. This result was derived from the definition of "correctness", and it is left to the practitioner to interpret the definition and develop a metric to accurately measure that criterion.

Some metrics are defined to help practitioners in their work. These measurements assist in larger areas such as measuring software development progress, code review, control flow testing, data flow testing, and system integration testing. They include measurements of the percentage of test cases executed, the percentage of successful functional tests, lines of code, number of lines of code reviewed per hour, total number of hours spent on code reviews per project, code coverage of tests, number of test cases produced, percent of known faults detected, number of faults detected, and the turnaround time for each test-debug-fix cycle [74].

Traditional metrics include cyclomatic complexity [64], code base size (lines of code) [79], and comment percentage of code [79]. McCabe introduced the metric "cyclomatic complexity" in his 1976 paper "A Complexity Measure" [64]. It measures the complexity of a given code base, and in object-oriented programming it is well-suited for measuring the complexity of class methods [79]. Rosenberg introduces additional metrics for object-oriented programming that include weighted methods per class (WMC), response for a class (RFC), lack of cohesion of methods (LCOM), coupling between object classes (CBO), depth of inheritance tree (DIT), and the number of children (NOC) [79].

Metrics are values, but without context and purpose they are useless. They must support a quality factor or characteristic to provide any use to the practitioner.

2.3.3 Software Security Metrics

Security metrics attempt to quantitatively measure aspects of a system's security. They can be used for numerous purposes, but they fall into one of three categories: strategic support, quality assurance, and tactical oversight [55]. Although research in metrics has been widespread, there is still disagreement among practitioners as to how effective they are and how they should be utilized [13, 54]. Jaquith describes security metrics as "the servants of risk management, and risk management is about making decisions. Therefore the only security metrics we are interested in are those that support decision making about risks for the purposes of managing risk" [56]. It is true that security is inherently tied with risk management, however it is still unclear as to how to utilize, and interpret, these metrics in a standardized manner. Metrics provide only a small part of the solution to the larger question of security, and recent initiatives for improved security rely on a more methodology-based approach such as the "Build Security in Maturity Model (BSIMM)" from McGraw et al [66], Common Criteria (CC) [51], ITSEC (which has been largely made obsolete by the introduction of the Common Criteria), and SSE-CMM [49, 55].

Various security metrics have been suggested and used in analysis such as security defect density or the number of defects discovered over time. These metrics are not sufficiently effective as they are "after the fact" and they do not take into account information such as how widely-used the system is - which will significantly affect the amount of defects discovered [54]. Additionally, code complexity has been used to give a measure of relative security in a system. The more complex a system is the more likely it is that there are bugs in it due to the human's ability to concentrate on a limited number of aspects at a given time. However complexity is subjective and does not guarantee an increase in defects consistent with an increase in complexity. It is, however unlikely, possible for a highly-complex system to be implemented correctly and have a significantly-reduced amount of defects. Realistically an increase in complexity will signal an increase in defects, but it will occur in varying degrees and will depend on the methodologies and processes of the implementing group. "It is no coincidence that the highest security evaluation levels typically are awarded to very simple systems" [54]. Furthermore, Bellovin expresses his concern over the difficulty, and "infeasibility", of metrics that measure security strength in software due to the linear structure of the system's security layers. Once an attacker defeats one layer, he can attack the next layer without much interference from the layer just defeated. This immediately negates all metric values associated with the defeated layer and renders them useless [13].

In respect to the apparent difficulties related to metrics and their accurate measuring and interpretation it has been suggested that a focus on vulnerability probability models be used, such as the Microsoft "Relative Attack Surface Quotient" and the amount of code audit effort to gain some insight into the security "state" of a given system [13]. In effect, there exists no "good metric" to differentiate two executables and say that one is better than another from a security standpoint [54].

2.4 Analysis & Tools

Software analysis, and automated program analysis, finds its roots in four reasoning techniques. Each technique relies on the technique below it to make the hierarchy: experimentation, induction, observation, and deduction [97]. Experimentation finds the causes of specific effects from induction. Induction summarizes the observed events into an abstraction. Observation logs targeted events that occur during execution. Deduction takes the abstract (usually code) and attempts to derive "what can or cannot happen in concrete runs" [97]. There are two types of software analysis methods: static analysis and dynamic analysis [74]. Static analysis is a form of dedution. The three remaining raesoning techniques (experimentation, induction, and observation) require execution of the target system and are forms of dynamic analysis [97].

2.4.1 Types of Software Analysis

Static Analysis Tools

All software projects are guaranteed to have one artifact in common - source code. [17]

Static analysis focuses on the analysis of code that has not been run and provides input that can be used to identify various security violations, some runtime errors, and logical errors [9]. The simplest of these tools scan the source code and identify matches that might suggest improper coding practices or vulnerabilities [17]. The power of this approach is that it requires no execution of the target source code. It can even evaluate certain levels of incomplete code [17].

Static analysis includes any analysis on the source code while it is not executing. This includes manual code review [17]. This can be time consuming and it requires the reviewer to have knowledge of the security vulnerabilities and coding practices that they will encounter. Static analysis tools provide an automated solution to this limitation and bring the knowledge of vulnerabilities, coding practices, and defects with them. Consequently, they are far more time-effective in their analysis [17]. They do not make manual review obsolete, however, as a well-informed and qualified reviewer will have significant success when reviewing source code.

It is more accurate to say that static analysis tools are semi-automated as their execution can be automated but they tend to produce large sets of results which are likely to include false-positives [9, 73, 99]. Therefore manual filtering, and possibly triage, of results from a static analysis tool may be required to benefit from this technique. Additionally, static analysis cannot detect poor design decisions or conceptual focuses such as extra security around login forms [17].

These tools vary in methodology, target language, efficiency, and effectiveness. They range from the simple usage of Unix's grep tool to the advanced static analysis tool PREfix used by Microsoft in their development lifecycle [73, 17]. Grep static analysis would utilize simple string pattern matching while other tools may rely on lexical analysis, and abstract syntax trees (AST) to provide more context and granularity in the results [17].

Dynamic Analysis Tools

"Dynamic analysis is the analysis of the properties of a running program" [11]. This is in contrast to static analysis which is the analysis of the code (or artefacts) of a program without execution. Dynamic analysis becomes crucial when details of a program's execution is hidden until runtime. Additionally, with the extensive use of dynamically linked libraries, and practices such as providing Java bytecode on demand, the information available to static analysis tools is dwindling [72]. Because dynamic analysis actually executes the program, the tester can follow execution paths and view the behaviour of the program when given certain inputs.

To be effective, dynamic analysis relies on two key characteristics [11]:

Precision of Information Collected: This assures that the system will not provide false readings.

Program Inputs: The chosen inputs will determine the paths taken by the program.

These two characteristics dictate the effectiveness of dynamic analysis. Inaccuracies in the precision of the information collected cause the results to be inaccurate, and poorly-chosen inputs may not provide an accurate representation of the target system since they may not cause sufficient path coverage.

Dynamic analysis is effective at following paths that are nested deep in a system. Every path in dynamic analysis is feasible by definition. This is in contrast with static analysis which may produce paths that are infeasible [11]. Dynamic and static analysis are complementary and using the strengths of both methods will produce a far more effective results [27].

There are many types of dynamic analysis tools. In fact, the only requirement for dynamic analysis tools is that they execute the target system and analyse the results. This allows for many methodologies and include input fuzz testing [18] or coverage testing tools [95].

2.4.2 Limitations, Direction & Future Work

Both static and dynamic analysis have their limitations. Dynamic analysis is limited in its inability to verify that a particular property is present in the system [11]. Static and dynamic analysis are not replacements for other quality control methodologies as they are unable to detect high-level design flaws [28]. Additionally, static analysis may provide high levels of false-positives, and many of the available tools do not scale to large and complex systems [99]. It has also been found that simple static analysis techniques are not effective for finding buffer overflows [99]. More complex methods are required and there is no static analysis tool that provides sufficient security tests out-of-the-box [1].

Surmounting these limitations, the community for program analysis is active and producing promising results. Analysis tools are currently used to observe the execution of mal-ware and generate “behavioural profiles” [24]. These profiles can be sorted so that manual inspection is only done after the initial sort. This utilizes the inspector's time more efficiently. Static analysis defect density can be used as early indicators of pre-release defect density, it can do this at statistically significant levels, and it can be used to determine the quality level of software components [73].

To continue this work, static analysis tools that are able to analyse large, and complex, software systems is required [99], there is a need to develop best practices that make using static analysis tools more effective [9]. This could be done by codifying "knowledge about security vulnerabilities in a way that makes it accessible to all programmers" [28]. This would shift from the reliance on the developer knowing the security best practices to the program enforcing them. Finally, there is a need to move away from a "penetrate-and-patch" model to a "penetrate-patch-and-prevent" model by constantly updating the knowledge base of the tools used [28].

2.5 Stochastic Optimization

Stochastic optimization is a method that accounts for randomness in the defined problem [83, 82]. This technique emerged due to the difficulties of traditional mathematical optimization methods with uncertainty, and potential trade-offs, in real-world applications [84]. Stochastic optimization uses an iterative, or step-by-step, approach where it starts with a "guess" and moves towards a value that is closer to the maximization, or minimization, goal, and can be applied to both continuous and discrete values [82].

Figure 2.4 identifies the two qualities that determine if stochastic optimization is applicable to a given problem.

Quality A: There is random noise in the measurements of the functions, and/or

Quality B: There is a random choice made in the search direction as the algorithm iterates toward a solution

---

Figure 2.4: Stochastic Optimization Problem Qualities [84]

Defining a stopping criteria with a *guaranteed* level of accuracy in stochastic search problems is difficult, as there will be avenues of exploration that have not been touched in any finite time defined, and an ordinal approach is often used (i.e. A >B) to find a “good-enough” solution [84]. Also due to this difficulty, comparison of competing algorithms is complicated and can be achieved by constraining the number of function evaluations to maintain objectivity or by running them on a known problem [84].

The most prominent algorithms and methods of stochastic optimization are listed in Figure 2.5. Some algorithms are designed specifically for continuous or discrete sets and may not be applicable to one or the other.

- Direct Search Methods
- Recursive Estimation Methods
- Stochastic Gradient Algorithm
- Simultaneous Perturbation Stochastic Approximation
- Annealing-Type Algorithms, and
- Genetic Algorithms

Figure 2.5: Stochastic Optimization Methods and Algorithms [84]

2.5.1 Direct Search Method

Direct search types include the random search and nonlinear simplex (Nelder-Mead) algorithms [84]. They typically require little information and make few assumptions about the underlying data. Direct search methods ignore gradient, called gradient-free, which make them significantly easier to implement [84].

Random search methods explore the domain in a random fashion to find a point that satisfies the minimization or maximization function requirement. They are easily implemented, take a limited number of measurements, generalized, and based on a theoretical background [84].

The Nonlinear Simplex (Nelder-Mead) algorithm is based on the concept of the simplex. It iteratively generates a new point near the current simplex to create a new simplex and evaluate. The goal is to move the simplex closer to the desired function goal of minimization of maximization [84]. The nonlinear simplex has no general convergence theory and cases of nonconvergence have been presented [84].

2.5.2 Recursive Estimation on Linear Models

Recursive estimation is a sequential method that updates the parameter estimates of the problem upon receipt of the previous iteration's results or data [96]. It can be used to estimate parameters for linear regression models and transfer function models. Recursive estimation may be used on non-linear models, such as the non-linear least squares approximation algorithm, through iteration [96].

2.5.3 Stochastic Gradient Algorithm

The stochastic gradient algorithm is used to solve bound constrained optimization problems and requires a differentiable loss function $L(\theta)$ for minimization [84]. The algorithm uses root-finding stochastic approximation to find the minimum θ^* in equation 2.1.

$$g(\theta) = \frac{\partial L}{\partial \theta} = 0 \tag{2.1}$$

The loss function $L(\theta)$ in equation 2.1 can be modeled for random or noisy variables by expressing it in terms of "observed cost as a function of the chosen θ and random effects V" [84]. This new function is expressed in equation 2.2.

$$L(\theta) = E[Q(\theta, V)] \tag{2.2}$$

"The ability to compute the derivative $\partial Q/\partial\theta$ is, of course, fundamental to the stochastic gradient approach", however limitations in the ability to compute $\partial Q/\partial\theta$ in many practical applications has led to the development of gradient-less solutions [84].

2.5.4 Simultaneous Perturbation Stochastic Approximation

Simultaneous perturbation stochastic approximation (SPSA) is a root-discovery-based maximum likelihood estimation algorithm which "relies on a derivative approximation other than the usual finite-difference approximation". It produces estimates based on measurements from the loss function $L(\theta)$ [84], and is more efficient in the number of comparisons used compared to finite-difference stochastic approximation [82]. SPSA excels in working with problems with many variables to be optimized (high dimensionality) and is applicable to both gradient and gradient-free situations [84].

2.5.5 Annealing-Type Algorithms

Annealing-type algorithms attempt to move from the local minima of a loss function $L = L(\theta)$ to its global minimum [84]. They are capable of evaluating both discrete and continuous measurements of $L(\theta)$, and they include the capability to accept a temporary increase in the loss function evaluation for an overall long-term decrease [84]. Annealing-type algorithms are capable of evaluating loss functions with arbitrary "degrees of nonlinearities, discontinuities, and stochasticity" including boundary conditions and constraints [52]. They are simple to implement compared to other optimization techniques, and they statistically guarantee that they will arrive at the optimal solution to the problem function [52, 84].

2.5.6 Genetic Algorithms

Genetic algorithms (GAs) are a type of evolutionary computing technique modelled after the natural evolutionary process [2, 46]. A GA is used to simulate the evolutionary progress of a population towards a certain "fitness" goal [12]. A "population", in this case, can be any group (called an individual, genotype, or chromosome) of features that are evaluated to provide a "fitness value" [92]. An evaluation function must be defined which provides one, or many, performance measures for a given individual. The fitness function then determines which individuals are most "fit" and should be used for creating the next-generation population [12, 92].

Upon execution the GA may use a pre-set initial population or it may produce a random population to begin the evaluation. This initial population is called $g0$ or "generation 0". Each individual in the population is evaluated using the defined evaluation function and their performance values are recorded. After the entire population is evaluated the fitness function determines which members of the population will be selected to create the new generation. This is called selection. Once a subset of individuals is selected the next generation is created by mutation, crossover. This method favours features that help individuals gain a high fitness rating and discourages the promotion of features that reduce the fitness of the individual - survival of the fittest [25, 2]. "The main advantage of a GA is that it is able to manipulate numerous strings simultaneously". This greatly reduces the chances of the GA getting stuck in a local minima [2].

Selection

"Competition-based selection is one of the two cornerstones of evolutionary progress", the other is variation within members of a population [25], and closely resembles "survival of the fittest" [43]. Selection provides the mechanism to separate the fit individuals from the unfit ones based on their fitness evaluation calculations.

There are numerous selection algorithms, however the following algorithms are the most dominant: Fitness Proportionate Selection, Elitist Selection, Rank Selection, and Tournament Selection [40, 36, 21].

Fitness Proportionate Selection (also known as Roulette-Wheel Selection) chooses individuals from a population based on the proportion of their performance compared to their peers [89]. For example, an individual that achieves a high fitness value will be assigned a larger proportion of the possible selection area - a "larger piece of the pie". In this algorithm, individuals that perform well are given the advantage, and individuals with low performances are not eliminated explicitly. The fitness proportionate selection algorithm is outlined in Fig 2.6

1. Evaluate each individual via the fitness function.
2. Sort the individuals with respect to their fitness values (performance).
3. Assign a probability of selection to each individual based on its fitness value (normalized).
4. Repeatedly make a random selection, based on the probabilities provided, until the number of individuals required is met.

Figure 2.6: Fitness Proportionate Selection Algorithm

Fitness proportionate selection can have poor performance and may not converge as quickly as other algorithms [89]. It can also eliminate the best candidate in extreme cases, and can preserve the worst candidate - which would also preserve its poorly-performing features. Fitness proportionate selection has a time complexity of $O(n^2)$ [40].

Elitist Selection is a general term used to refer to selecting the best "individuals" and allowing them to propagate, unchanged, to the next generation [86, 43, 23]. This preserves the traits contained within the individual and helps reduce the amount "genetic drift" that occurrs [5]. Elitist selection includes algorithms such as truncation selection, block selection, and (μ+λ) selection, and it is generally used as a supplement to proportional selection [36]. The elitist selection algorithm includes sorting the population based on fitness and choosing the x individuals with the highest fitness.

"The elitist selection strategy balances the disruptive effects of high crossover and mutation rates" [86]. It tends to be more "exploitive than explorative", and, as such, elitist selection improves the GAs performance but increases the chance of premature convergence [23]

"Elitist strategies are suitable to find a global optimal solution but may fail for problems in which multiple optimal solutions are required" [23]. A simple elitist algorithm will have a time complexity based on its sorting algorithm and can be done in $O(n{\cdot}log(n))$ [40].

Rank Selection was first introduced by Baker [10] to solve the problem of a few super-chromosomes dominating the early generations of a GA [36]. Rank selection attempts to mitigate this issue by ranking chromosomes, or individuals, to determine survival probability [40]. This makes the survival probability proportional to the rank of the individual and not directly to the fitness value it possesses. The pre-defined selection probabilities assigned to each rank can be defined by a linear or exponential ranking scheme [36]. Rank selection has a runtime of $O(n{\cdot}log(n)) + (selectiontime)$ [40]. The algorithm for rank selection is detailed in Figure 2.7.

1. Evaluate each individual via the fitness function.
2. Sort individuals based on their fitness values.
3. Assign to each individual a pre-defined selection probability based on their rank position.
4. Randomly generate a number in the range of probabilities to select an individual.
5. Repeat the random generation and selection process until the desired number of individuals is met.

Figure 2.7: Rank Selection Algorithm [36, 40]

Tournament Selection includes randomly choosing individuals from the population and selecting the one with the highest fitness value from the group [40, 86]. By keeping only the tournament winners, even when the contestants are chosen randomly, the average population fitness is increased [71]. Increasing the size of the tournament will increase the average population fitness at the expense of variance in the new generation. Tournament selection can be thought of as a "noisy form of ranking" [92].

The most common tournament size is 2 [40], called binary tournaments, which converges slower than higher sizes but is resistant to premature convergence. Even with a tournament size of 2, tournament selection is fast with a time complexity of $O(n)$ [40]. Tournament selection is a popular choice for GAs for their time complexity and efficiency in both parallel and non-parallel architectures [71, 29]. The algorithm for tournament selection is detailed in Figure 2.8.

1. Evaluate each individual via the fitness function.
2. Randomly select s individuals from the population, where s is the tournament size.
3. Select the individual from the selection that has the highest fitness value.
4. Repeat the selection process until the number of individuals desired is selected.

Figure 2.8: Tournament Selection Algorithm

Mutation

Mutation provides a level of randomness to the population which allows for greater variance of the traits it may exhibit [2]. It also reintroduces "lost or unexplored genetic material into the population to prevent the premature convergence of the GA to suboptimal solutions" [85]. Mutation usually happens at a low probability [2, 25]. A common probability for mutation is between 0.005 and 0.05 [85].

In a simple example, if an individual is represented as a list of binary values such as (1, 0, 1), there is a probability that the second bit, represented as 0, could mutate to the value 1. This would change the individual to (1, 1, 1). Each bit has the same probability of mutating. Mutation is simulated by generating a random number for each bit in the binary string and comparing it to a given minimum value. If the generated number is lower than the minimum the bit is flipped [25].

Crossover

Crossover (also called mating, recombining, or elitist recombination) is the combination of features from two "parent" individuals to produce offspring with similar "inherited" traits [25, 2]. This recombination occurs with a given probability, usually between 0.5 and 1.0, called the crossover probability or rate [85]. Crossover occurs with the belief that "good" solutions will generate "better" ones while "goodness" is determined from the fitness evaluation [85].

There are various methods for recombining the "genetic material" of the parent individuals to produce offspring, the most popular methods are single-point, two-point, and uniform crossover [85, 86].

Single-point crossover includes splitting the parents at a single, common, point and swapping sections from them to produce two offspring with genetic material from both parents [2].

Two-point crossover includes choosing random start and end points to define the sections for splitting the parents. It "eliminates the single-point crossover bias toward bits at the ends of strings" [86].

Uniform crossover works on a more granular level than n-point crossover. It uses a fixed probability to individually evaluate each bit in the string, or individual [86]. This means that the probability of crossover for each bit is independent of the other bits in that string.

Limitations, Direction & Future Work

Open issues in genetic algorithms include choosing control parameters, the exact roles of crossover and mutation, and convergence properties [86].

Control parameter choice is difficult and context-specific, but research into "generally accepted" parameters has produced promising results [85, 40, 2, 25, 86, 26]. Additionally, the exact role of crossover and mutation includes a delicate balance between the organized and the chaotic. Research into the use of crossover and mutation has produced suggestions for certain classes of problems but specific applications still require domain-specific configuration [85, 40]. Research into convergence properties includes investigations into, and suggestions of, specific selection operators to prevent premature convergence [43, 41, 89].

An additional issue with genetic algorithms is “deceptions”. Deceptions are confusing or misleading “solutions” that lead the genetic algorithm to a solution that is not “globally competitive” [93].

Research in genetic algorithms has focused on keeping a steady pressure towards convergence while avoiding premature convergence to local maxima [40]. Further research into increased performance and memory usage of genetic algorithms has also emerged [5]. Improvements in mutation and crossover algorithms has been steady [21, 29].

Chapter 3

Hermes: An Overview

In this chapter we introduce Hermes, a targeted fuzz-testing framework that generates tangible evidence for security case arguments. Hermes is a proof-of-concept system which employs common static analysis, code coverage, fuzz testing, and genetic algorithm tools and libraries. It hones the traditional scatter-gun approach of fuzz testing and specifically targets areas of the code that have the highest-severity defects.

Hermes is designed to achieve the goals set in section 1.3, and to ultimately provide a more robust security case argument using fuzz testing. Additionally, the Hermes framework is designed to be extended on a structural level via modular design, and on a target protocol level via the protocol definition language structure.

3.1 Architectural Overview

3.1.1 Design

Hermes works on a client-server architecture which facilitates both remote and local testing of targets. The processing for both the genetic algorithm and the fuzz test generation is on the server side while the client is a thin wrapper that includes the EMMA code coverage tool. The client monitors the target and sends the coverage metrics to the server to complete the asynchronous loop of "test, measure, revise" detailed in Figure 3.3.

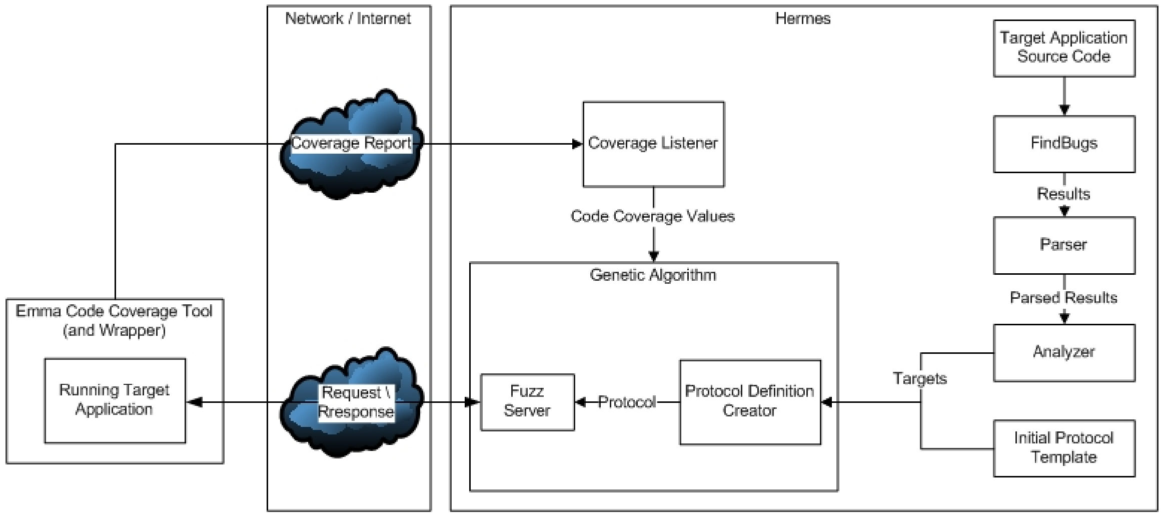

Figure 3.1: A view of Hermes' architecture.

3.1.2 Modules

Preprocessing (Target Painting)

The preprocessing portion of the Hermes framework includes the execution of a static analysis tool, the parsing of the results, and the extraction of targets. Hermes uses the tool "FindBugs" for the initial static analysis phase. The parser module formats the results into a comparable form, and the analyser module selects the targets based on the selection criteria configuration.

FindBugs executes on the target software's source or jar file. It detects potential defects and sorts them based on one of the categories listed in Figure 3.2.

The parser takes the raw output from FindBugs as input and extracts all instances of a potential defect. FindBugs calls the potential defect element a "BugInstance". For each bug instance, the parser includes some context information from the results. It keeps a list of the bug instances that are sent for analysis.

The analyser unpacks the bug instance object and sorts it based on the selection criteria specified. It then selects the most severe, largest, or highest priority defects to be targeted based on the Hermes configuration. These targets are sent to the genetic algorithm controller to start testing the target application.

- Bad Practice
- Correctness
- Experimental
- Internalization
- Malicious Code Vulnerability
- Multi-threaded Correctness
- Performance
- Security
- Dodgy Code

Figure 3.2: Categories for potential defects in FindBugs.

Protocol Generator (Test Generation)

The protocol generator module produces a protocol for a specified language. It does this by using a protocol template that specifies the language's syntax. If a developer wishes to extend the language capability of Hermes, they simply specify the syntax using the language libraries provided by Hermes. These libraries create a general representation of languages that allows Hermes to accept different languages as input and treat them the same during execution.

The genetic algorithm controller uses the protocol generator to dynamically change the protocol based on the coverage feedback provided. It uses the protocol generator to add or remove protocol features to test the target application.

Fuzz Server (Testing)

The fuzz server module contains both the fuzz testing logic and the server used to connect with the client. Hermes uses the Sulley[7] fuzzing libraries to manage the fuzz testing mechanism of the testing suite. Sulley is a Python fuzz test framework that is mature and well-used.

The fuzz server takes the protocol that was created by the protocol generator and uses it to fuzz test the target application. The module acts as a server, but the libraries provided allow for Hermes to be extended to work locally without a server aspect. The target application is fuzz tested until it gives no more input and a timeout is triggered, if a maximum number of requests is set and hit, or if a maximum amount of time has been configured and has passed. The results are stored locally for auditing and debugging.

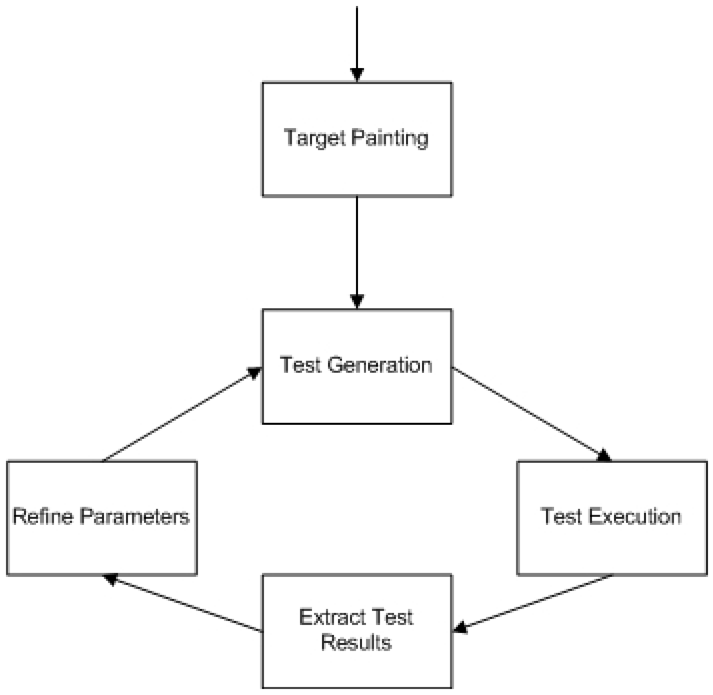

Figure 3.3: Hermes' test flow.

Coverage Wrapper (Test Results)

The coverage wrapper is a module that includes the EMMA code coverage tool and a small mechanism to transmit calculated coverage results back to the server for analysis. The target application is set in the configuration of the wrapper, and, once started, the wrapper will wait until the target has terminated and transmit the report. The wrapper was designed to be as light-weight, and simple, as possible since this software will be on the client.

Genetic Algorithm (Refining)

The genetic algorithm (GA) module includes both the protocol definition and fuzz server modules. The GA module listens for the coverage reports sent from the wrapper, parses them, and revises its protocol based on the feedback enclosed. It stops and starts the fuzz server based on these parameters.

The GA module revises the protocol using the protocol generator to maximize code coverage of the provided targets. The GA module runs until a time limit is met (can be configured to run indefinitely), until the fuzz server has been stopped and started a certain number of times, or until the maximum number of generations configured has been calculated.

3.2 Process Overview

The process which Hermes uses to provide its targeted fuzz testing is split into two sections: the client section and the server section.

The client section is detailed in Figure 3.4. It shows the flow of information as the coverage wrapper is initialized and executed. It specifies the inputs and outputs for the two main mechanisms of the client: EMMA and the coverage transmitter. The server section is shown in Figure 3.5. It displays the flow of information, and the process taken, as FindBugs, the parser, analyser, and genetic algorithm are executed.

---

1. Execute Wrapper.
    (a) **Input:** Target application path (configuration).
    (b) Execute EMMA as a background process.
        i. **Input:** Target application path.
        ii. **Output:** Coverage report.
    (c) Execute coverage transmitter as background process.
        i. Wait until a new coverage report is generated.
        ii. Process new coverage report.
            A. **Input:** Raw coverage report.
            B. Package report.
            C. Send report to server.
    (d) **Repeat Until:** Configured time reached, or if stopped.

Figure 3.4: Hermes Client Process Overview

1. Execute FindBugs.
   (a) **Input:** Target application source (or jar)
   (b) **Output:** Raw results.
2. Execute Parser.
   (a) **Input:** Raw results from FindBugs.
   (b) Extract potential defects and contextual information.
   (c) **Output:** Formatted list of potential defects.
3. Execute Analyser.
   (a) **Input:** Formatted list of potential defects from parser.
   (b) Sort potential defects based on selection criteria.
   (c) Select top potential defects based on selection criteria.
   (d) **Output:** List of target potential defects.
4. Execute Genetic Algorithm.
   (a) **Input:** List of target potential defects, protocol template.
   (b) Generate initial protocol
   (c) Execute Fuzz Server.
      i. **Input:** Protocol definition.
      ii. Wait for requests.
      iii. Handle requests.
         A. **Input:** Request.
         B. **Output:** Generated protocol (fuzzed).
      iv. **Repeat Until:** timeout, max responses, stopped by GA.
   (d) Listen for coverage report from client.
      i. **Input:** Coverage report.
      ii. Parse coverage report
      iii. Refine protocol.
      iv. **Output:** Refined protocol
   (e) Restart Fuzz Server with refined protocol.
   (f) **Repeat Until:** max time reached, max generations reached

Figure 3.5: Hermes Server Process Overview

Chapter 4

Hermes: Implementation

In this chapter we examine Hermes in greater detail. Each component of Hermes is discussed including their design, reasoning, and relationship to each other. Also included in this chapter is the initialization sequences required to start Hermes and analyse a target application.

4.1 Detailed Architecture

4.1.1 Modules

Preprocessing

The purpose of the preprocessing module is to take the target application source code, or jar, and produce a list of targets for Hermes to focus its fuzz testing effort. The preprocessing module includes the FindBugs static analysis tool, parser, and analyser detailed in Figure 3.1. These separate components are called from a prepared script that is configured to analyse the target application. The Linux version of this script is detailed in Figure 4.1.

In the case of Figure 4.1, the target application is located at the relative path "Target_Projects/Crawler4j/crawler4j-3.5.jar" and the application's auxiliary libraries are located at "Target_Projects/Crawler4j/jars". These paths are both relative to Hermes' root directory. FindBugs is called specifying the output format as XML, and to output the results to the file called "fb_results.xml". The parser accepts the XML file generated by FindBugs and, if it is available, the defect density metric. The analyser takes the results file generated by the parser, sorts the potential defects set

```
clear

classpath=Target_Projects/Crawler4j/jars
fboutputfile=fb_results.xml
fbtargetjar=Target_Projects/Crawler4j/crawler4j-3.5.jar
fbddfile=defectdensity.txt

echo "Updating Hermes config and modules..."
sudo python setup.py install

echo "Running FindBugs..."
./findbugs-2.0.2/bin/findbugs -textui -xml -auxclasspath \
$classpath -output $fboutputfile $fbtargetjar
echo "Done."

echo "Running Parser..."
python Parser/parser.py -d $fbddfile -b $fboutputfile
echo "Done."

echo "Running Analyzer..."
python Analyzer/analyzer.py -r parse_results.txt
echo "Done."
```

Figure 4.1: Preprocessing script for Linux

by the configuration, and stores the resulting list of targets in a file specified by the configuration stored in "Config/analysis.py".

The use of scripts was required to combine multiple components that were written in different languages. Each component was developed to execute independently from the other components. The independence of Hermes' components allows it to be easily extended and tested. The use of a script simplifies the calling of these components and allows for system calls to be made such as calling external applications like FindBugs or EMMA.

Protocol Generator

The protocol generator creates protocols based on a template and a list of features to include. Hermes specifies which features will be included, and excluded, with an "individual" from the genetic algorithm. The individual is a binary string where a 1 means that a particular feature is included, and a 0 means that a particular feature is excluded. Therefore, if a set of features included "has wheels", "has engine", and "has drive-shaft" we can specify the inclusion or exclusion of these features using an individual such as (1, 1, ,1). This particular individual would enable all three of the

- Anchors,
- Images,
- Divs,
- IFrames,
- Objects,
- JavaScript, and
- Applets

Figure 4.2: Features included in Hermes Individuals.

features where (1, 0, 1) would enable two features, but remove the “engine” feature.

The individuals specified in Hermes include seven unique features for the target protocol - for this experiment, HTML. The HTML features are listed in Figure 4.2 and are capable of being nested to arbitrary depths.

The protocol generator uses a template to break a protocol down into atomic features. For example, in HTML a specific tag, such as the anchor tag, would be considered a single feature. The features of a protocol are described in the Sulley protocol language. This adds another layer of abstraction but allows for individual features, and attributes within each feature, to be used for fuzz testing. An example of an anchor HTML tag in the Sulley protocol notation is detailed in Figure 4.3.

```

# The following code (Python) will produce the anchor tag:
# <a alt="this will be fuzzed." href="127.0.0.1">So will this.</a>

s_static("<a alt=\"")
s_string("this will be fuzzed.")
s_static("\" href=\"")
s_string("127.0.0.1")
s_static("\">")

s_string("So will this.")

s_static("</a>")
```

Figure 4.3: An HTML anchor tag in the Sulley protocol notation.

Because a developer would have to learn the Sulley protocol notation before they could specify a new language, we have created helper libraries that allow a developer to produce new Sulley protocols programmatically. The libraries automatically escape quotations, add proper indentation, and include contextual commenting for readability to produce syntactically valid Python code. The Hermes helper libraries treat each protocol as an n-ary tree and each instance of a feature is a node. The use of a tree allows for an arbitrary level of nesting, and a simple traversal generates the protocol from the tree representation. An example of an HTML page in the definition tree representation is detailed in Figure 4.4.

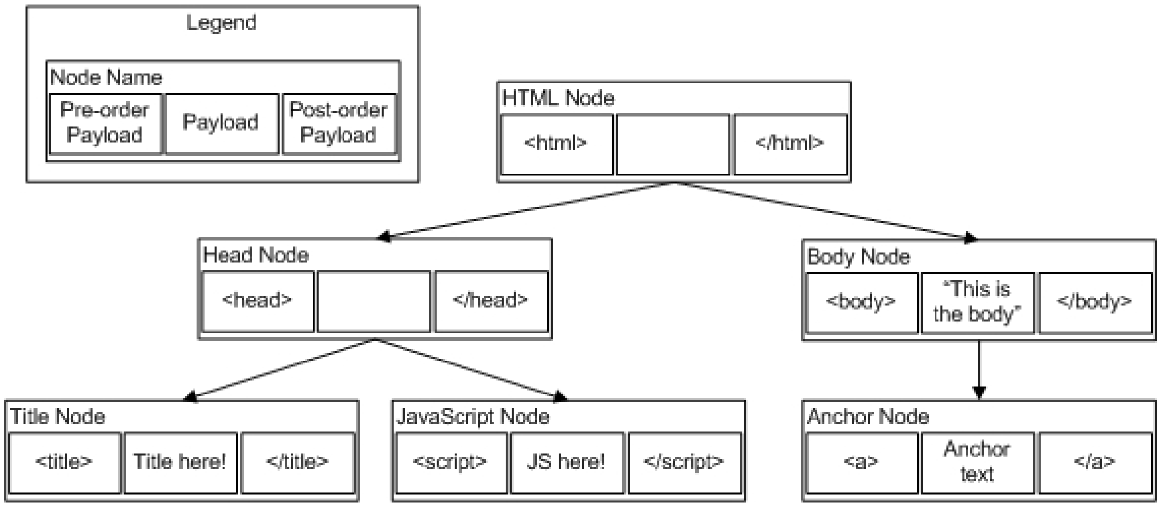

Figure 4.4: A simple HTML page in tree notation.

A traversal is required to convert the tree representation into a protocol. To do this, Hermes executes a pre-order traversal of the tree which includes any pre, or post, payload inclusions. In the example of Figure 4.4, a pre-payload inclusion is displayed in the left-most element within each node. It contains anything that the developer wants to be added to the protocol before the node's payload is added. The post-payload inclusion is displayed in the right-most element within each node. It contains anything that the developer wants added to the protocol after the node's payload is added. A node's children are considered payload. To continue with the example given in Figure 4.4, the pre-payload inclusion of the HTML node is the text "$< html >$" and the post-payload inclusion is "$< /html >$". All children and payloads are contained within these two elements. Attributes are specified by the developer and handled by Hermes' helper libraries. An example of the protocol definition that is generated from an individual that contains only anchor features is detailed in the Appendix, section 8.1.

Fuzz Server

The fuzz server contains the fuzz testing and transmission components of Hermes. It takes a protocol definition as input, and sends the generated protocol as a response to any requests sent to the server. The fuzz server is a self-contained module that includes the Sulley libraries built-in, and a simple HTTP server to handle any HTTP requests it receives.

The HTTP server may be configured to limit the number of responses it gives, the amount of time it will service requests, or to run indefinitely. An HTTP server was chosen due to the nature of the target application used in our evaluations, but the fuzz server module is designed to serve a mutation of the protocol upon request. The nature of the request, be it HTTP or otherwise, does not matter. An example of how the fuzz server generates a response is detailed in Figure 4.5. The *self.Request* variable is an instance of the Sulley fuzz engine. The engine renders a mutation, sends it as a response, and generates a new mutation to be used in the next response.

```

def do_GET(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()

        mutation = self.Request.render()

        self.wfile.write(mutation)
        self.Request.mutate()
```

Figure 4.5: Fuzz Server response for an HTTP GET request.

Coverage Wrapper

The coverage wrapper is a light-weight module that includes the EMMA code coverage tool and a transmitter to send the generated coverage reports to the server. The module is self-contained and includes all of the software required to wrap a target application for Hermes' analysis.

A main script is used to control the execution of the Java-based EMMA, the Python-based transmitter, and the external Java-based target application. This script includes the configuration to find the target application. It is detailed in Figure 4.6. EMMA is executed with the verbose flag to produce the richest data sets, and with the ouput format set to XML. The XML file is loaded and sent to the server by the transmitter to complete the process. The final parameters "test" and "1" give EMMA the directory to dump debug information and the number of threads to use respectively.

```

curr=${PWD}
targetpath=$curr/Target_Projects
emmapath=$curr/Coverage/EMMA/lib

java -XX:-UseSplitVerifier -cp $emmapath/emma.jar emmarun -verbose \
-r xml -jar $targetpath/BasicCrawler.jar test 1

python $curr/Coverage/CoverageWrapper.py -f coverage.xml
```

Figure 4.6: Linux implementation of the Hermes coverage wrapper script.

Hermes contains scripts to execute a single evaluation of the target application and to execute repeated evaluations of the target application. The repeated evaluations script will restart the target application if it crashes or terminates and continue to evaluate it. The single evaluation script will not restart the target application.

Genetic Algorithm

The genetic algorithm module controls the fuzz server and protocol generator modules. It includes the logic for the genetic algorithm to maximize code coverage by manipulating protocol parameters and resetting the fuzz server with these new parameters. Included in this module is a light-weight coverage listener that waits for coverage reports to be sent.

Based on our research of stochastic optimization, the genetic algorithm method applied most naturally to a generalized code coverage maximization algorithm. The genetic algorithm specifically excels when little is known about the controlling parameters that must be manipulated to maximize the fitness metric. This is because many other stochastic optimization methods require prior knowledge about the problem such as the stochastic gradient algorithm, simultaneous perturbation stochastic approximation, and annealing-type algorithms. The nature of fuzz testing lends itself favourably towards a genetic algorithm approach. This is because fuzz testing requires randomization and mutation to be effective, and these properties are inherent to the genetic algorithm approach. Finally, previous research has used genetic algorithms applied to fuzz testing [94, 44].

The coverage listener “listens” for coverage reports sent from the coverage wrapper on the client. It parses the report and extracts overall coverage metrics and statistics, and target-specific coverage metrics at the levels set in the configuration. The default configuration will calculate the line coverage for the entire method that contains the target potential defect. This default was set because static analysis is known to have false positives, but, from our discussion in section 2.3, areas with a high potential defect density, even with false positives, signal an area of interest. By attempting to maximize line coverage of the entire method Hermes is able to target problem areas without suffering from too narrow of a focus.

Hermes uses the DEAP[32] genetic algorithm library to facilitate the generation of individuals for analysis. Each individual of a population is represented as a binary string. This string determines the features that the particular individual contains. Individuals are selected, “bred”, or mutated based on the parameter configuration of the algorithm. A detailed list of the parameters used in the genetic algorithm are shown in section 5.1.

4.2 Initialization and Execution

4.2.1 Starting Hermes’ Server

The Hermes server is controlled by a single script. It calls the preprocessing script detailed in Figure 4.1, and then the Hermes control script to start the fuzz server and coverage listener for the genetic algorithm. A detailed look at the Linux version of this script is shown in Figure 4.7.

4.2.2 Starting Hermes’ Client

The Hermes client is controlled by a single script. It restarts the target application for either a certain duration of time or for a certain number of repetitions. It is detailed in Figure 4.8.

```

sudo ./uni_findbugrun.sh
echo "Resetting any foreign servver values..."
python hermes.py -r
echo "Done"
echo "Starting Hermes Fuzz Server..."
sudo python hermes.py -f > hermes_f_dump.txt 2>&1 &
echo "Done"
echo "Starting Hermes Coverage Listener..."
sudo python hermes.py -c > hermes_c_dump.txt 2>&1 &
echo "Done"
```

Figure 4.7: Hermes server start script for Linux.

```

max_time_minutes=4320
now_ts=$(date +%s)
end_ts=$((now_ts + max_time_minutes*60))

curr=${PWD}
while [ $(date +%s) -lt $end_ts ]
do
    java -XX:-UseSplitVerifier -cp $curr/Coverage/EMMA/lib/emma.jar \
    emmarun -verbose -r xml -jar \
    $curr/Target_Projects/BasicCrawler.jar test 1

    python $curr/Coverage/CoverageWrapper.py -f coverage.xml
done
```

Figure 4.8: Hermes client start script for Linux.

4.2.3 Logging and Reports

Hermes is a framework comprised of multiple independent components. This modular design lends itself to be easily extended. Each component includes its own log that is stored in “Logs/”. These logs include debug and general execution information that may be useful to developers when they are extending Hermes or to track Hermes’ execution progress. The “Logs/” directory contains the logs listed in Figure 4.9.

In addition to the execution logs, Hermes stores all generated coverage reports in a format that makes it easy to read and compare. These logs are stored in the “ServerLogs/” directory. These logs contain the overall results from the analysis, what individual from the genetic algorithm this evaluation pertains to, what were the targets, and further details about the coverage of individual targets.

Finally, the genetic algorithm keeps logs of the coverage reports received and a

CVG_Listener.log: This is the coverage listener log.

EMMAXMLParser.log: This is the log for the XML parser of coverage reports.

fuzzer_lib.log: This is the fuzz server library log.

GA_CRI.log: This is the genetic algorithm log.

hermes.log: This is the general Hermes execution log.

Figure 4.9: Logs contained in the “Logs/” directory.

general overview of each report in a human-readable format. The coverage reports received are stored in "GA/Reports/" and the human-readable equivalents are stored in "GA/Logs/".

Chapter 5

Evaluation and Analysis

This chapter focuses on our evaluation and analysis of Hermes. We evaluated Hermes with respect to the desired features listed in section 1.3. These features will be addressed individually and the results discussed.

5.1 Evaluation Configuration

5.1.1 Hardware

Each evaluation was executed in its own Amazon EC2 t1.micro instance. The t1.micro instances have 1 vCPU and 0.613GiB of ram. We chose the micro instance because, even though the evaluation process is somewhat CPU-intensive, our evaluations are based on the number of requests sent to the server - which is not a CPU-specific metric.

5.1.2 Software

The evaluations were executed on the Ubuntu 14.04 LTS platform. Hermes is written in Python and utilizes the DEAP genetic algorithm libraries - also written in Python. EMMA was used to provide the code coverage feedback and Java was installed to facilitate it. Finally, Git was used as the versioning software to control the Hermes source, and so it was also installed on the platform.

5.1.3 Hermes

The default selection criteria for Hermes' analyser was set to use the FindBugs internal "bugrank" metric. Bugrank is calculated by FindBugs using a combination of the potential defect's category specified in section 3.1 and the type of potential defect found. The bugrank metric represents an overall severity metric for the potential defect.

The genetic algorithm was configured to be more aggressive in its mutation capabilities. This allows for more features to be brought back into the "gene pool" if there is an early, dominating, individual that does not converge to a maximum later in the analysis, or if the population size is small. The initial configuration of the genetic algorithm is detailed in Figure 5.1.

P(Crossover) = 0.5

P(Mutation) = 0.05

Number of Generations = 30

Individuals per Generation = 10

Selection Algorithm = Tournament Selection (size=3)

Mutation Limit = 3600

Figure 5.1: Initial configuration of Hermes' genetic algorithm

High growth ratios and increased mutation rates allow for quicker convergence in simple problems but suffer with more complex problems. These issues can be mitigated partially by increased population sizes and multiple populations with varying success [40]. In the configuration detailed in Figure 5.1, a small population is used with a higher mutation, and lower crossover, probability. A higher population will increase variability within the population, which will include more individuals with high fitness, but it will slow the convergence to a maximum. For the purposes of time we chose a small population size of 10 for 30 generations. The mutation limit of 3600 was chosen to limit the amount of time required to evaluate a single individual in the genetic algorithm.

Typical crossover probabilities lie in the 0.5 to 1.0 range, while typical mutation probabilities are in the 0.005 to 0.05 range [85]. Tournament selection was used because “ranking and tournament selection are shown to maintain strong growth under normal conditions, while proportionate selection without scaling is shown to be less effective in keeping a steady pressure towards convergence” [40].

Because the target application is a crawler, detailed in section 5.3.2, Hermes was configured to act as a “honeypot” server where it captures the crawler by providing non-repeating and self-directed links back to itself. Once the crawler was captured, the server would respond with mutated HTML mixed with the valid HTML and begin the fuzz test process. The use of valid HTML within the server’s response ensures that the crawler is always provided a valid HTML link back to the server so that it continues to be captured.

5.2 Metrics

5.2.1 Code Coverage

Code coverage is a measure of the amount of source code a specific test suite is able to evaluate.

There are three common forms of code coverage: function coverage, path coverage, and statement coverage [62]. Function coverage is the number of functions that are called by a given test suite. Statement coverage is the total number of individual statements executed by the test suite. Path coverage measures the coverage of all possible routes through the executed code. These values are compared to the total number of functions, statements, or paths in the target application to produce a code coverage percentage.

In addition the the general definition of code coverage, Sutton [87] defines code coverage within the context of fuzz testing to be “the amount of process state a fuzzer induces a target’s process to reach and execute”.

Code coverage was chosen as a metric for Hermes because, although it is “well-known that random testing usually provides low code coverage, and performs poorly overall [in that respect]” [37, 15], code coverage has been extensively used as a metric to measure the performance of fuzz testing [75, 44, 94, 53, 88, 18, 62], additionally, there is a general “lack of measurable parameters that describe fuzz test completeness” to draw from [18]. Specifically, we chose line coverage for the entire method where a target bug type was identified as the metric used for our analysis.

EMMA

The tool used to measure code coverage for Hermes is the EMMA coverage tool [80]. EMMA is a powerful, open-source, Java code coverage tool. It can be configured for class, method, line or block code coverage and produces a variety of output formats for analysis. Additionally, EMMA does not specifically "require access to the source code and degrades gracefully with decreasing amount of debug information" [80]. It is a pure Java application that requires no external libraries to execute.

5.2.2 Performance

The evaluation of Hermes requires a performance metric. We chose to evaluate Hermes based on the number of mutations that the fuzzer generates to achieve a target code coverage. A baseline performance metric is set by executing an undirected and exhaustive fuzz test on the target while logging the number of mutations computed and the level of code coverage achieved. In the context of the Crawler4J crawler, the number of mutations equals the number of crawler requests to the server as each request included a single mutation of the protocol.

Previous research has utilized a variety of performance metrics including number of fuzzed inputs, total errors found, errors found per hour, and number of distinct errors found per hour [33, 44, 94, 88]. The number of fuzzed inputs (mutations) was chosen because it is less subjective than a pure time comparison. A time comparison could be improved by simply increasing the CPU power of the host machine and would introduce a subjective aspect to our analysis.

5.3 Evaluation Procedure

The Hermes evaluation process is best described using an analogy:

A house is infested with cockroaches. From a top-down view of the house, and with the roof removed, an exterminator can spray pesticide in an undirected fashion for 1 hour and never look at the results of his actions. This may kill a great number of cockroaches, but this is far from an optimum strategy, and it is difficult to reliably reproduce similar results.

> Now, the exterminator could improve their method by spraying pesticide randomly and indiscriminately for 1 hour with their eyes open to see how they did. This would give more contextual information such as where most of the cockroaches are killed and where the biggest ones are killed.
>
> Finally, the exterminator could spray pesticide for 1 hour with their eyes open while looking at where the highest concentration and largest cockroaches are, and focusing their efforts in those areas. This method starts to maximize the limited amount of pesticide to the areas that require the most attention.

From the analogy, the exterminator's blind, undirected, 1-hour treatment is equivalent to fuzz testing for a given amount of time without any further processing. In a security case, this would include the statement "Fuzz tested for x hours". The second method of spraying pesticide blindly then looking after is equivalent to fuzz testing, noting the code coverage, and noting the number of defects found. In a security case, this would include the statement "Fuzz tested for x hours, achieved y code coverage, and discovered z defects". Finally, the last method of spraying pesticide with eyes open and focusing on the areas that require the most attention is equivalent to running static analysis to find the problem areas, targeting the fuzz testing based on the static analysis results, and logging the results. In a security case, this would include the statement "Targeted fuzz testing to find w vulnerability for x hours, achieved y code coverage, and discovered z defects.

5.3.1 Overview

For the evaluation of Hermes, we chose to compare the last two methods in the analogy specified at the beginning of this chapter. The methods are detailed in Figure 5.2.

Undirected Fuzz Testing Method

The undirected fuzz testing method includes taking a full protocol and exhaustively fuzz testing the target application with it. A full protocol is equivalent to a protocol generated with all features turned on. In the genetic algorithm, this would mean that the individual who produces this protocol is a string of all ones. The protocol

1. Undirected fuzz test that notes code coverage and detected defects.
2. Directed fuzz test that targets a particular bug type and notes code coverage and detected defects.

Figure 5.2: Compared methods of Hermes analysis.

is exhaustively fuzz tested by sending every possible mutation of the protocol to the target application. The execution of this method is monitored and the code coverage calculated and the number of mutations that were generated are logged. A full, undirected, and exhaustive fuzz test produces the best-possible code coverage for that protocol and is essentially brute-forcing the protocol. This also means that it will produce the maximum number of mutations for that protocol. These values establish the baseline that will be used to compare the directed fuzz testing method.

Directed Fuzz Testing Method

The directed fuzz testing method utilizes the features of Hermes to generate a targeted protocol for the target application. It attempts to maximize the code coverage of the target areas within the target application while reducing the number of mutations required to achieve this coverage value. It does this by dynamically generating a protocol and evaluating it based on the code coverage of the target areas. In the genetic algorithm the code coverage is the main fitness criteria for the protocol. It continually generates new protocols and evaluates them in an effort to produce a protocol that maximizes code coverage and minimizes redundant features in the protocol. Redundant features are features that do not contribute to an increase in code coverage of the target areas. They increase execution time and are undesirable. Once the protocol that represents the best-fit individual is generated, it is used to exhaustively fuzz test the target application. This is essentially brute-forcing the targeted protocol to produce a worst-case evaluation. The code coverage achieved and number of mutations are both logged for comparison and evaluation.

We chose to select the best-fit individual as the the highest-performing individual after all of the generations in the genetic algorithm have been evaluated. Another popular selection method is to use a convergence interval that terminates the genetic algorithm once the difference of its values converges to within the scope of a given interval. Given the nature of Hermes' randomized testing, the chosen population size, and the higher mutation rate, we decided that a convergence interval would be less ideal as it is possible for larger jumps in the fitness values. The highest-performing individual method ensures that the best-fit individual is selected after each evaluation.

Method Comparison and Evaluation

For each evaluation, Hermes was configured to target a certain percentage of offending code in the target application. "Offending code" is based on the calculated FindBugs bug rank sorted by severity. Both the undirected and directed methods were executed and their code coverage of the target areas measured. The protocol for the baseline calculation was kept constant and EMMA was configured to track the code coverage for a given target area. This baseline code coverage was then compared to the calculated values of Hermes' directed fuzz approach.

Directed protocols were generated by Hermes' genetic algorithm with specific targets (in this case percentage of most severe offending code) in mind. The resulting best-fit protocol was then exhaustively evaluated to produce code coverage and number-of-mutation metrics for that target range.

The configured percentages of offending code in our evaluations started at 10% and incremented by 10% up until 100% inclusively. Therefore ten evaluations were executed for both the directed and undirected methods. In our evaluation, if Hermes is targeting the top 10% of offending code it is targeting the 10% of potential defects with the highest calculated bug rank severity.

5.3.2 Target Application

The target application selected for evaluation is the Crawler4J [34] library. To fuzz test this library a simple crawler was developed using Crawler4J and configured to connect to the Hermes fuzz test server.

The Crawler4J library was selected for evaluation because it is a Java-based application with its most recent revision being downloaded just under 15,000 times, it is designed to be extended, and it is open-source. The fact that Crawler4J is open-source allowed Hermes to provide full introspection and utilize its white-box features to their fullest extent.

5.3.3 Procedure Steps

Our evaluation procedure is broken into two sections: the undirected fuzz testing baseline and the directed fuzz testing using Hermes.

Our first step is to establish a baseline for the metrics specified in section 5.2. This is achieved by executing the undirected fuzz testing method on the target application and logging both the code coverage and the number of mutations for the configured target area percentage.

Once a baseline is established, the evaluation of the directed fuzz testing method is executed with respect to the same metrics. Each directed evaluation is compared to its associated undirected counterpart for the given target area percentage. For example, the directed and undirected evaluations for the top 10% of offending code are compared, and so on. The procedure steps are outlined in Figure 5.3.

1. Execute undirected fuzz testing method on target application.
   (a) Configure method for target area percentage (10%, 20%, 30%, ..., 100%).
   (b) Exhaustively fuzz test target application with a full protocol.
   (c) Record metrics
2. Execute directed fuzz testing method on target application.
   (a) Configure method for target area percentage (10%, 20%, 30%, ..., 100%).
   (b) Generate targeted protocol using Hermes' genetic algorithm.
   (c) Exhaustively fuzz test target application using targeted protocol.
   (d) Record metrics.
3. Compare results.

Figure 5.3: Evaluation procedure.

Targeted % of Code	Mutations	Mean Coverage	Standard Deviation	Complement Mean Coverage	Complement Std. Deviation
10	67788	0.8625	0.1304	0.4343	0.4664
20	67788	0.79	0.1485	0.4339	0.4663
30	67788	0.8425	0.1575	0.4334	0.4663
40	67788	0.7036	0.3606	0.4343	0.4664
50	67788	0.7283	0.3548	0.4337	0.4662
60	67788	0.785	0.3145	0.4337	0.4664
70	67788	0.8148	0.2901	0.433	0.4662
80	67788	0.7834	0.3248	0.4334	0.4663
90	67788	0.7493	0.3339	0.433	0.4663
100	67788	0.7415	0.3315	0.4328	0.4663

Table 5.1: Baseline results from an undirected and exhaustive fuzz test with a full protocol

5.4 Results and Analysis

5.4.1 Experiments

Baseline: Undirected Fuzz testing with Code Coverage

We calculated our baseline by brute-force fuzzing the full protocol definition. By brute-forcing the protocol we are able to produce the worst-case values for number of mutations, mean code coverage, the standard deviation for the mean code coverage, the mean code coverage of the targeted code's complement, and the standard deviation for the complement's mean code coverage. These results are detailed in Table 5.1. The mean code coverage is used with the standard deviation to provide an overall value for all of the sections of targeted code. Each code section's individual code coverage is recorded in Appendix 8.2. The target code complement data represents the code coverage that is *not* part of the target scope. The total number of possible unique mutations for the full target protocol is 67788. The baseline exhaustively evaluates the protocol which explains the constant in the "Mutations" column.

Figure 5.4 provides a bird's-eye view of the baseline results. It follows an expected behaviour where the code coverage is high and the standard deviation is low when only looking at small sections of the code, but as the amount of target code increases so does the standard deviation and the mean coverage decreases.

For the baseline, we observed that a noticeable drop in code coverage and increase in standard deviation when the target code reaches 40%. This may signal that a

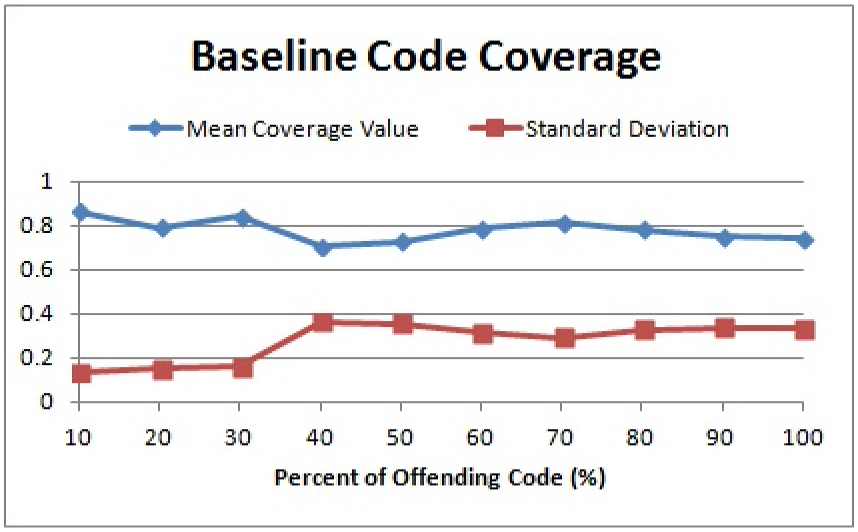

Figure 5.4: Baseline code coverage and standard deviation

defect with little or no code coverage was added that was not in the previous set. In fact, observation of the detailed baseline results for 30% and 40% in Appendix 8.2 show that *two* defects are added with little code coverage. This would cause the drop in coverage we observed. The discrepancy between 30% and 40% is most prevalent in the standard deviation values where we observe a jump from a standard deviation of 0.1575 to 0.3606 – more than double.

Directed Fuzz Testing with Code Coverage

Hermes evaluates a target application in two steps:

1. It first finds a best-fit candidate protocol that performs best under a restricted number of mutations – in this case 3600 mutations. Table 5.2 details the results for each best-fit candidate.

2. It then exhaustively fuzzes the best-fit protocol to fully evaluate the target application with respect to the given target code. Table 5.3 details the results of exhaustively fuzzing the best-fit protocols.

The results in Table 5.2 are already surprisingly similar to the baseline detailed in Table 5.1 – with a significant reduction in mutations. Most of the mean coverage results for the best-fit protocols are within 3% of their baseline counterparts. This is true for all values except the 80% evaluation which differed by 5.58%. After further

Targeted % of Code	Mutations	Mean Coverage	Standard Deviation	Complement Mean Coverage	Complement Std. Deviation
10	3600	0.8425	0.1622	0.4116	0.4624
20	3600	0.775	0.1636	0.412	0.4625
30	3600	0.8312	0.172	0.4116	0.4624
40	3600	0.6954	0.3623	0.4115	0.4624
50	3600	0.7208	0.3569	0.4115	0.4624
60	3600	0.752	0.3131	0.4094	0.4619
70	3600	0.8064	0.2962	0.4102	0.4622
80	3600	0.7276	0.3472	0.41	0.4621
90	3600	0.7441	0.3343	0.4097	0.4622
100	3600	0.7365	0.3318	0.4108	0.4624

Table 5.2: Results from the best-fit candidates produced by Hermes

investigation it was revealed that this drop in accuracy was due to a significantly-lower code coverage in a single section of offending code. As with the baseline, we observe the jump in standard deviation at the 30% to 40% mark. Finally, these values do not represent the entire best-fit protocol and they must be exhaustively evaluated to assure that the results are not simply “surface” matches.

Targeted % of Code	Mutations	Mean Coverage	Standard Deviation	Complement Mean Coverage	Complement Std. Deviation
10	41964	0.8625	0.1304	0.4171	0.4637
20	51648	0.79	0.146	0.4307	0.4656
30	41964	0.8425	0.1575	0.4347	0.4662
40	35508	0.7027	0.3607	0.4161	0.4635
50	51648	0.7283	0.3548	0.432	0.4657
60	50185	0.785	0.3145	0.4311	0.4658
70	41964	0.81	0.2939	0.4162	0.4638
80	23672	0.7834	0.3248	0.4147	0.4634
90	49162	0.7493	0.3339	0.4316	0.4659
100	49981	0.7415	0.3315	0.4319	0.4658

Table 5.3: Results from exhaustively evaluating the generated best-fit protocols

The results from an exhaustive evaluation of the best-fit protocols are detailed in Table 5.3. Here, we observe fluctuations in the number of mutations, and thus the computation time, of the evaluations. This is the result of Hermes tailoring each protocol to attack the specified set of potential defects while pruning any redundant or useless features to minimize the total number of mutations. Furthermore, 8 of the

10 evaluations achieve parity with their baseline counterparts on mean code coverage, and the other 2 evaluations are within 0.5% of *their* baseline counterparts.

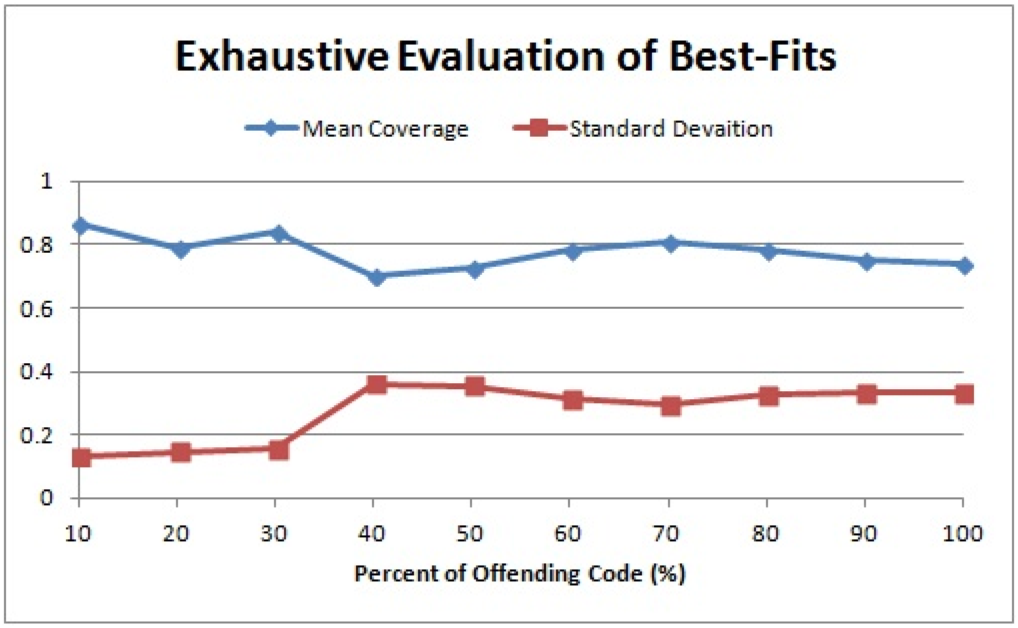

Figure 5.5: Exhaustive Evaluation: Best-fit code coverage and standard deviation

The standard deviation results followed a similar trend set by the mean coverage. Seven of the ten evaluations achieved parity with their baseline counterparts with the other 3 evaluations within 0.4% of their baseline counterparts. In one case, the 20% evaluation, we observed a *decrease* in the standard deviation of the mean coverage while continuing to maintain mean coverage parity.

5.4.2 Analysis

Best-Fit Protocols and their Accuracy

The best-fit protocols generated by Hermes were produced by selecting the best-performing protocol in a 3600-mutation evaluation (detailed in Table 5.2). The resulting protocols were then exhaustively evaluated (fuzzed) to produce the values shown in Table 5.3.

Figures 5.6 and 5.7 compare the code coverages and standard deviations of the best-fit evaluations and the full (exhaustive) best-fit evaluations. In Figure 5.6 we observe that the full evaluation achieves better code coverage in every evaluation, but does not deviate from the initial best-fit evaluation with 3600 mutations in a significant manner. The two major discrepancies are at the 60% and 80% evaluations with a difference of 3.3% and 5.58% respectively. In Figure 5.7 the full evaluation achieves a lower or equivalent standard deviation compared to the initial evaluation. The notable differences are at the lower percentage targets (10%, 20%, and 30%) and at the 80% evaluation. At these areas we see a lower standard deviation than the initial best-fit evaluations.

The similarity between the initial and full best-fit evaluations may be due to the size of the target application or the size of the individual used in the genetic algorithm. The deviation between the initial and full evaluations may increase with a change in either of these two factors – which are discussed further in Chapter 6.

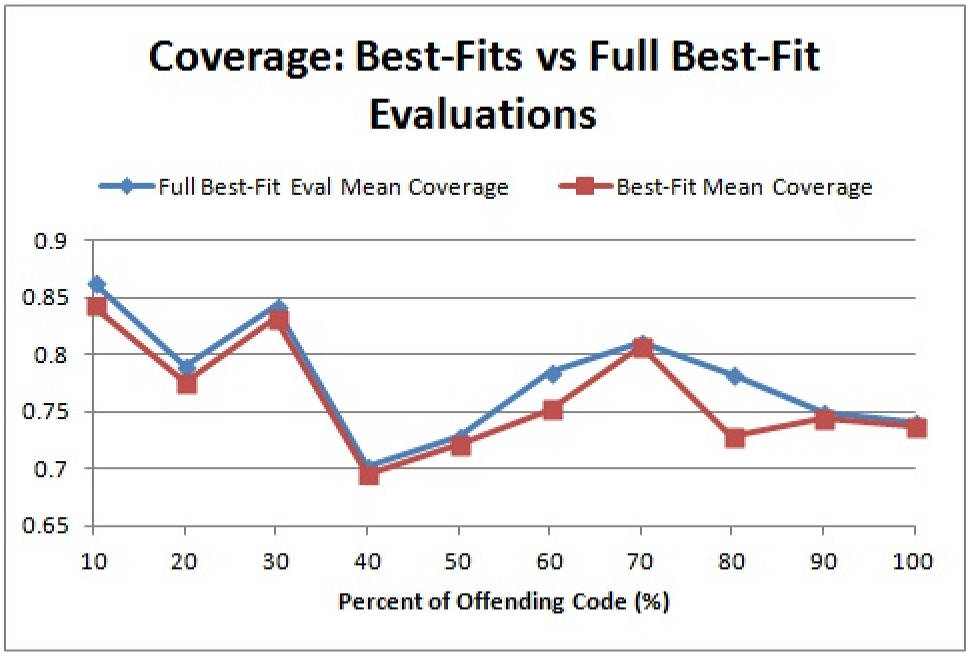

Figure 5.6: Code Coverage: Best-Fit Evaluations (3600 mutations) vs. Full Best-Fit Evaluations

Comparing Directed and Undirected Approaches

The directed (full best-fit) evaluations and the undirected (baseline) evaluations are compared in Table 5.4. In this table we observe that the full best-fit mean coverage and standard deviation results achieve parity with their baseline counterparts in all but a few areas – and the areas that *did not* achieve parity are still within 0.5% of their targets.

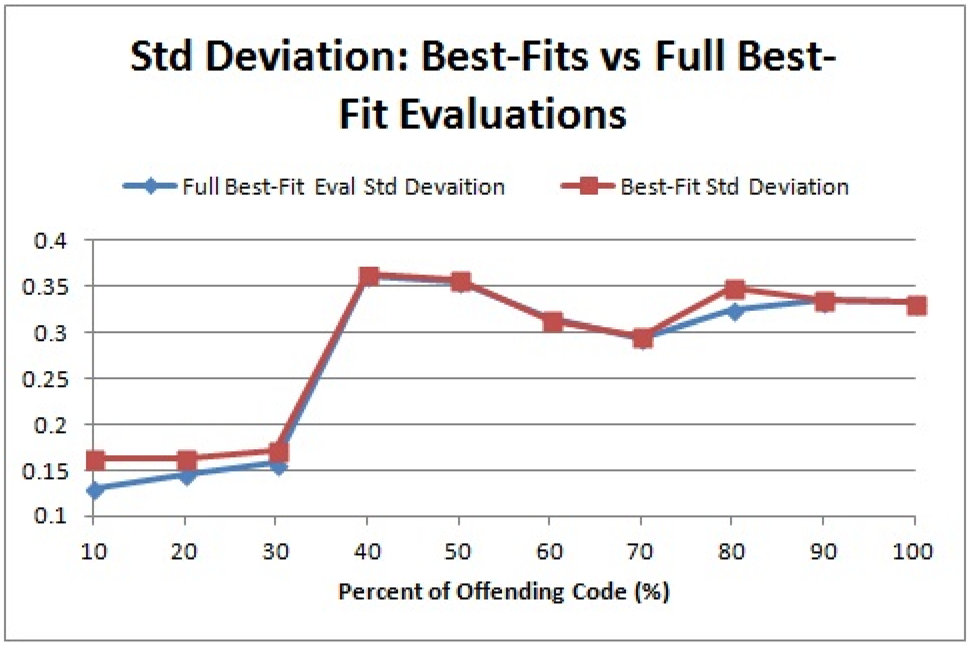

Figure 5.7: Standard Deviation: Best-Fit Evaluations (3600 mutations) vs. Full Best-Fit Evaluations

Additionally, we observe from Table 5.4 that *every* evaluation was able to reduce the number of mutations (and thus computation time) required. In the case of the 80% evaluation Hermes was able to reduce the number of mutations by 65% from 67788 to 23672 mutations while achieving complete parity in both mean code coverage *and* standard deviation. The minimum improvement observed from our evaluations is a decrease in the number of mutations by 23.8% (from 67788 to 51648 mutations) while maintaining parity within 0.5% of mean code coverage and standard deviation.

5.5 Desired Features

Chapter 1 specified certain acceptance criteria for a solution. In this section we address Hermes' feature-set and if they satisfy the criteria outlined.

5.5.1 Target Specific Types of Defects

Hermes is designed to target specific types of defects. In our evaluation we targeted the top percentage of offending code based on the FindBugs bugrank metric, but evaluations can be based on any configured metric or sorting algorithm. For example,

Percentage (%)	Difference in Mean Coverage (%)	Difference in Std Deviation	Baseline # Mutations	Full Best-Fit # Mutations	Difference in # Mutations
10	0	0	67788	41964	25824
20	0	0.0025	67788	51648	16140
30	0	0	67788	41964	25824
40	0.0009	0.0001	67788	35508	32280
50	0	0	67788	51648	16140
60	0	0	67788	50185	17603
70	0.0048	0.0038	67788	41964	25824
80	0	0	67788	23672	44116
90	0	0	67788	49162	18626
100	0	0	67788	49981	17807

Table 5.4: A comparison of the baseline and the full evaluations of best-fit protocols

it is possible to configure Hermes to specifically target buffer overflow vulnerabilities.

These configurations allow security practitioners to target specific vulnerabilities. The related fuzz tests provide targeted evidence which can be used to support a security assurance argument. Each report provides evidence for a separate argument. For example, the results of a configuration that targets buffer overflows could be used to argue that the target application is specifically resistant to buffer overflow attacks.

5.5.2 Achieve Code Coverage Parity with Existing Methods

For our evaluations we used the existing fuzz testing methods as our baseline. With this in mind, we were able to achieve equivalency with the baseline for both mean code coverage *and* standard deviation to within 0.5%.

5.5.3 Improve the Performance of Fuzz Testing Method

Our evaluations were able to reduce the computation time of an equivalent evaluation by a minimum of 23.8% and up to 65%.

5.5.4 Integrate Framework Without Significant Overhead

Hermes is completely independent from other testing tools. This independence allows it to be attached to existing test suites or continuous integration frameworks easily and without interference. Analysis, and generation of new maximized protocols, can be executed, continuously, behind the scenes, and on a separate system. It produces a "best-so-far" protocol definition after each iteration which means that the current results can be used for testing while Hermes works towards improving them.

5.5.5 Repeatable and Reviewable Evidence for Security Cases

Each best-fit result file (specified in Appendix 8.3) specifies what individual produced the reported results. Hermes includes the capability to take an individual's specification and recreate the protocol that the given individual represents. Once the protocol is created, Hermes is able to use it to fuzz test the target application.

This feature allows Hermes to repeat any fuzz tests that have been done in the past. The best-fit result file specifies what targets are included in its evaluation and includes the detailed values for each target. This allows each evaluation to be reviewed easily. Both of these features provide a far better security assurance argument than the baseline fuzz testing method as they are repeatable and easily reviewable.

Chapter 6

Future Work

6.1 Hermes Framework Improvements

6.1.1 Improved Fuzz Test Library

Currently, the fuzz test library includes both the fuzz test generation and the HTTP server. Extraction of the fuzz test generation into its own module would make Hermes more easily extensible and would contribute to a more manageable framework.

6.1.2 More Atomic Language Features

Hermes' evaluations currently consists of a limited feature-set of HTML. Improving the feature-set, by adding more atomic features, may improve code coverage, and may allow Hermes to test deeper-nested code structures. An increase in the feature set would increase the size and complexity of the genetic algorithm's individual. For example, if there are five features, the individual will be of length 5, and if there are 10 features the individual will be of length 10. This is difficult as an increase in the individual's size is accompanied by a significant increase in the genetic algorithm's computation time.

6.2 Evaluation Improvements

6.2.1 Genetic Algorithm Configuration

The initial genetic algorithm configuration of Hermes provides a balance between execution time, mutation and crossover probabilities, and population and generation size. However a more in-depth analysis, paired with the more atomic features specified in section 6.1.2 may provide resistance to features that dominate early in the evaluation but become limited later in the evaluation. This new suggested configuration is specified in Figure 6.1. This would require increasing the population size of the data set while reducing the mutation probability from 0.05 to 0.001. This new mutation probability reflects the larger population size as features will not be eliminated early with a larger population.

P(Crossover) = 0.6

P(Mutation) = 0.001

Number of Generations = 30

Individuals per Generation = 100

Selection Algorithm = Tournament Selection (size=2)

Figure 6.1: Another suggested configuration of Hermes' genetic algorithm

6.2.2 Increase Best-Fit Mutations for Genetic Algorithm

In our initial best-fit evaluations we chose 3600 mutations as the number of mutations to be used. This number was chosen due to time constraints. In the future, we would like to modify this number and observe the effects it has on the results. In our evaluations, we used a relatively small target application. A larger application may require an increase in the initial best-fit number of mutations to provide accurate results.

6.2.3 Further Target Applications

Future analysis would include a greater set of target applications. This would provide a richer set of evaluations to compare. Additionally, the target application we evaluated was relatively small. In the future we would like to evaluate larger applications in an attempt to observe Hermes' ability to cope with larger data sets and targets.

Chapter 7

Conclusions

In this thesis, we investigated the state of the art in fuzz testing, their use in security assurance cases, and the models and metrics that security assurance cases rely on for evidence of assurance. Then, we surveyed methods in stochastic optimization as a route to improve code coverage of targeted fuzz testing.

Evidence in security assurance cases must be definitive, convincing, and accurate. The more specific the evidence the stronger the associated assurance argument. Our research has revealed a lack in assurance with the evidence provided by fuzz testing. This is not to say that fuzz testing is inadequate for use as evidence in security assurance cases, but we argue that there are improvements that can be made.

In this thesis we introduced Hermes; our solution that targets a particular bug type for fuzz testing and provides targeted evidence for assurance cases beyond simply executing for a certain duration. Hermes' features were able to achieve and satisfy our specifications for a potential solution and are listed below:

- Target specific types of defects
- Achieve code coverage parity with an undirected and exhaustive fuzz test
- Reduce the time required to fuzz test targets
- Provide repeatable and reviewable evidence for security cases, and
- Easily integrate into existing test frameworks

Our results showed a promising improvement in execution time (improvements from 23.8% to 65%), while targeting specific bug types (in this evaluation we chose the most-severe defects), and achieving equivalent code coverage to an exhaustive fuzz test (within 0.5%). Hermes provides a practical approach to gathering specific and targeted evidence for security assurance cases.

Chapter 8

Appendix

8.1 A Protocol Created by Hermes

```

'''
    Author: HTMLTreeConstructor
    Description:
        Auto Generated Protocol Definition: Chromosome:
        [1, 0, 0, 0, 0, 0, 0]

    Source: PDHelpers.py
    Created: 2014-05-15 19:00:38.530000
'''

from sulley import *
import random

s_initialize("Protocol Definition")

s_static("<html>")
s_static("<head>")
s_static("<title>")
s_string("Sulley Says Hello!")
s_static("</title>")
s_static("</head>")
s_static("<body>")

# Beginning of block: body_block
if s_block_start("body_block"):
    s_string("body_block+assurance")

    # Beginning of block: body_block_a1_block
    if s_block_start("body_block_a1_block"):
        s_string("body_block_a1_block+assurance")

        # Begin <a> tag
        s_static("<a ")
        s_static("alt=\"")
        s_string("body_block_a1")
        s_static("\" ")
        s_static("href=\"127.0.0.1/")
        s_string("body_block_a1")
        s_static("\" ")
        s_static(">")
        s_static("</a>")
        # End <a> tag

    s_block_end("body_block_a1_block")


    # Begin <a> tag
    s_static("<a ")
    s_static("alt=\"")
    s_string("body_block_a1_block")
    s_static("\" ")
    s_static("href=\"")
    s_checksum("body_block_a1_block", algorithm="sha1")
    s_static("\" ")
    s_static(">")
    s_static("</a>")
    # End <a> tag

s_block_end("body_block")

# Begin <a> tag
s_static("<a ")
s_static("alt=\"")
s_string("body_block")
s_static("\" ")
s_static("href=\"")
s_checksum("body_block", algorithm="sha1")
s_static("\" ")
s_static(">")
s_static("</a>")
# End <a> tag

s_static("</body>")
s_static("</html>")
```

8.2 Full Result Files for Baseline

8.2.1 Evaluation: 10%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 67788
Start Time of Fuzz Server:              2014-07-02 06:21:32.865175
End Time of Fuzz Server:                2014-07-02 18:21:33.827517

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.8625
Standard Deviation:     0.130455931256

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.61, 0.81, 0.88, 1.0]
        Line CVG (lc):  [0.66, 0.95, 0.84, 1.0]
------------------------------

TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.433984
Standard Deviation:     0.466233461338

TARGETS:
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.2.2 Evaluation: 20%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 67788
Start Time of Fuzz Server:              2014-07-02 06:01:22.307708
End Time of Fuzz Server:                2014-07-02 18:01:23.029898

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.79
Standard Deviation:     0.148548533034

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 0.61, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.67, 0.66, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.433977884475
Standard Deviation:     0.466319368072


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.2.3 Evaluation: 30%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 67788
Start Time of Fuzz Server:              2014-07-02 05:55:13.449823
End Time of Fuzz Server:                2014-07-02 17:55:13.663996

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.8425
Standard Deviation:     0.157539677542

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 1.0, 1.0, 0.61, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.67, 1.0, 1.0, 0.66, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.433475476641
Standard Deviation:     0.466326244953


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.2.4 Evaluation: 40%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 67788
Start Time of Fuzz Server:            2014-07-02 06:10:32.277796
End Time of Fuzz Server:              2014-07-02 18:10:32.692414

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.703636363636
Standard Deviation:     0.360662842486

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 0.61, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 0.66, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.434389924287
Standard Deviation:     0.466430225727


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.2.5 Evaluation: 50%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 67788
Start Time of Fuzz Server:              2014-07-04 06:17:33.634430
End Time of Fuzz Server:                2014-07-04 18:17:34.107048

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.728333333333
Standard Deviation:     0.354890436927

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.61, 0.81, 0.57, 0.88,
                        1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.66, 0.95, 0.62, 0.84,
                        1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.43380078637
Standard Deviation:     0.466295927046


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.2.6 Evaluation: 60%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 67788
Start Time of Fuzz Server:             2014-07-07 05:47:45.353690
End Time of Fuzz Server:               2014-07-07 17:47:46.283561

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.785
Standard Deviation:     0.314571136629

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.61, 0.81, 0.41, 0.57,
                        0.59, 0.83, 1.0, 1.0, 1.0, 1.0, 1.0, 0.86, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.66, 0.95, 0.39, 0.62,
                        0.64, 0.93, 1.0, 1.0, 1.0, 1.0, 1.0, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.432914419012
Standard Deviation:     0.466364347935


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.2.7 Evaluation: 70%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 67788
Start Time of Fuzz Server:              2014-07-07 05:53:06.604539
End Time of Fuzz Server:                2014-07-07 17:53:07.357370

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.8148
Standard Deviation:     0.290146445782

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.61, 0.81, 0.41, 0.57,
                        0.59, 0.83, 1.0, 1.0, 1.0, 1.0, 1.0, 0.87, 1.0, 1.0, 1.0,
                        0.86, 0.9, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.66, 0.95, 0.39, 0.62,
                        0.64, 0.93, 1.0, 1.0, 1.0, 1.0, 1.0, 0.79, 1.0, 1.0, 1.0,
                        0.84, 0.88, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.433171870441
Standard Deviation:     0.466326410893


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->TLDList (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList-><static initializer>
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->contains (String): boolean
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->getInstance (): TLDList
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->createParameterMap (String): SortedMap
        Bug Type:       SF_SWITCH_NO_DEFAULT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.2.8 Evaluation: 80%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 67788
Start Time of Fuzz Server:              2014-07-12 05:56:11.700854
End Time of Fuzz Server:                2014-07-12 17:56:12.246326

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.783461538462
Standard Deviation:     0.324806952224

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.61, 0.81, 0.41, 0.0,
                        0.57, 0.59, 0.83, 1.0, 1.0, 1.0, 1.0, 1.0, 0.87, 1.0, 1.0,
                        1.0, 0.86, 0.9, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.66, 0.95, 0.39, 0.0,
                        0.62, 0.64, 0.93, 1.0, 1.0, 1.0, 1.0, 1.0, 0.79, 1.0, 1.0,
                        1.0, 0.84, 0.88, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.433213191303
Standard Deviation:     0.466327234226


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        DocIDServer->addUrlAndDocId (String, int): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->TLDList (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList-><static initializer>
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->contains (String): boolean
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->getInstance (): TLDList
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->createParameterMap (String): SortedMap
        Bug Type:       SF_SWITCH_NO_DEFAULT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.2.9 Evaluation: 90%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 67788
Start Time of Fuzz Server:              2014-07-07 06:08:03.266410
End Time of Fuzz Server:                2014-07-07 18:08:04.125452

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.74935483871
Standard Deviation:     0.333959003509

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.61, 0.81, 0.41, 0.0,
                        0.73, 0.81, 0.57, 0.51, 0.59, 0.83, 1.0, 1.0, 1.0, 1.0, 1.0,
                        0.87, 1.0, 1.0, 1.0, 0.88, 0.9, 1.0, 0.0, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.66, 0.95, 0.39, 0.0,
                        0.58, 0.7, 0.62, 0.58, 0.64, 0.93, 1.0, 1.0, 1.0, 1.0, 1.0,
                        0.79, 1.0, 1.0, 1.0, 0.84, 0.88, 1.0, 0.0, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.432816880841
Standard Deviation:     0.466263069893


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        DocIDServer->addUrlAndDocId (String, int): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        DocIDServer->getNewDocID (String): int
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        DocIDServer->getDocId (String): int
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        Parser->parse (Page, String): boolean
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->TLDList (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList-><static initializer>
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->contains (String): boolean
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->getInstance (): TLDList
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->createParameterMap (String): SortedMap
        Bug Type:       SF_SWITCH_NO_DEFAULT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        Parser->parse (String): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Parser->parse (InputSource): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
```

8.2.10 Evaluation: 100%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 67788
Start Time of Fuzz Server:              2014-07-08 05:57:19.376283
End Time of Fuzz Server:                2014-07-08 17:57:19.509619

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.7415625
Standard Deviation:     0.331550423908

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.52, 0.0, 1.0, 0.61, 0.81, 0.41,
                        0.0, 0.73, 0.81, 0.57, 0.51, 0.59, 0.83, 1.0, 1.0, 1.0, 1.0,
                        1.0, 0.87, 1.0, 1.0, 1.0, 0.86, 0.9, 1.0, 0.0, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.5, 0.0, 1.0, 0.66, 0.95, 0.39,
                        0.0, 0.58, 0.7, 0.62, 0.58, 0.64, 0.93, 1.0, 1.0, 1.0, 1.0,
                        1.0, 0.79, 1.0, 1.0, 1.0, 0.84, 0.88, 1.0, 0.0, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.432877172484
Standard Deviation:     0.466304784286


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        RobotstxtServer->fetchDirectives (URL): HostDirectives
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        DocIDServer->addUrlAndDocId (String, int): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        DocIDServer->getNewDocID (String): int
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        DocIDServer->getDocId (String): int
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        Parser->parse (Page, String): boolean
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->TLDList (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList-><static initializer>
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->contains (String): boolean
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->getInstance (): TLDList
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->createParameterMap (String): SortedMap
        Bug Type:       SF_SWITCH_NO_DEFAULT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        Parser->parse (String): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Parser->parse (InputSource): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
```

8.3 Best-Fit Result Files

8.3.1 Evaluation: 10%

```
Individual: [1, 0, 1, 1, 1, 0, 0]

Number of responses from Fuzz Server: 3600
Start Time of Fuzz Server:            2014-02-12 12:45:43.860000
End Time of Fuzz Server:              2014-02-12 13:45:46.463000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.8425
Standard Deviation:     0.162230545829

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.51, 0.81, 0.86, 1.0]
        Line CVG (lc):  [0.58, 0.95, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.411640180784
Standard Deviation:     0.462402249351


TARGETS:
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.3.2 Evaluation: 20%

```
Individual: [1, 0, 1, 1, 1, 0, 1]

Number of responses from Fuzz Server: 3600
Start Time of Fuzz Server:              2014-06-27 05:24:24.925127
End Time of Fuzz Server:                2014-06-27 06:24:30.093008

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.775
Standard Deviation:     0.163681601491

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.67, 0.51, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.66, 0.58, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.412007879761
Standard Deviation:     0.462537897071


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.3.3 Evaluation: 30%

```
Individual: [1, 0, 1, 1, 1, 0, 0]

Number of responses from Fuzz Server: 3600
Start Time of Fuzz Server:              2014-02-10 06:24:29.390765
End Time of Fuzz Server:                2014-02-10 07:24:58.012464

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.83125
Standard Deviation:     0.172005632175

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.67, 1.0, 1.0, 0.51, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.66, 1.0, 1.0, 0.58, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.411601750547
Standard Deviation:     0.462465377146


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.3.4 Evaluation: 40%

```
Individual: [1, 0, 1, 1, 0, 0, 0]

Number of responses from Fuzz Server: 3600
Start Time of Fuzz Server:              2014-02-09 02:57:00.284000
End Time of Fuzz Server:                2014-02-09 03:57:28.588000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.695454545455
Standard Deviation:     0.362349997463

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.67, 0.0, 1.0, 1.0, 1.0, 0.51, 0.0, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.66, 0.0, 1.0, 1.0, 1.0, 0.58, 0.0, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.411537675234
Standard Deviation:     0.462406518129


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.3.5 Evaluation: 50%

```
Individual: [1, 1, 1, 0, 1, 0, 1]

Number of responses from Fuzz Server: 3600
Start Time of Fuzz Server:              2014-02-14 11:01:36.231518
End Time of Fuzz Server:                2014-02-14 12:02:15.568391

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.720833333333
Standard Deviation:     0.356988756437

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.67, 0.0, 1.0, 1.0, 1.0, 0.51, 0.0, 1.0, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.66, 0.0, 1.0, 1.0, 1.0, 0.58, 0.0, 1.0, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.41155597723
Standard Deviation:     0.462440615554


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.3.6 Evaluation: 60%

```
Individual: [1, 1, 1, 1, 1, 1, 0]

Number of responses from Fuzz Server: 3600
Start Time of Fuzz Server:              2014-05-01 21:45:50.971020
End Time of Fuzz Server:                2014-05-01 22:46:23.670907

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.752
Standard Deviation:     0.313107010461

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.44, 0.0, 1.0, 0.81, 0.67, 0.0, 1.0, 1.0, 1.0, 0.71, 1.0,
                                0.41, 0.57, 0.58, 0.59, 0.71, 1.0, 1.0, 1.0, 1.0]
        Line CVG (lc):  [0.51, 0.0, 1.0, 0.95, 0.66, 0.0, 1.0, 1.0, 1.0, 0.68, 1.0,
                                0.39, 0.62, 0.8, 0.64, 0.79, 1.0, 1.0, 1.0, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.409444607268
Standard Deviation:     0.461912881844


TARGETS:
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
       ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
```

8.3.7 Evaluation: 70%

```
Individual: [1, 0, 1, 1, 1, 0, 0]

Number of responses from Fuzz Server: 3600
Start Time of Fuzz Server:              2014-05-02 03:59:16.002160
End Time of Fuzz Server:                2014-05-02 04:59:38.302744

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.8064
Standard Deviation:     0.296201012827

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0]
        Block CVG (bc): [0.67, 0.0, 1.0, 1.0, 1.0, 0.51, 0.0, 1.0, 0.81, 0.41, 0.57,
                        0.48, 0.83, 1.0, 1.0, 1.0, 1.0, 1.0, 0.87, 1.0, 1.0, 1.0,
                        0.86, 0.9, 1.0]
        Line CVG (lc):  [0.66, 0.0, 1.0, 1.0, 1.0, 0.58, 0.0, 1.0, 0.95, 0.39, 0.62,
                        0.52, 0.93, 1.0, 1.0, 1.0, 1.0, 1.0, 0.79, 1.0, 1.0, 1.0,
                        0.84, 0.88, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.410203394791
Standard Deviation:     0.462287548161


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->TLDList (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList-><static initializer>
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->contains (String): boolean
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->getInstance (): TLDList
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->createParameterMap (String): SortedMap
        Bug Type:       SF_SWITCH_NO_DEFAULT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.3.8 Evaluation: 80%

```
Individual: [1, 1, 0, 1, 1, 1, 1]

Number of responses from Fuzz Server: 3600
Start Time of Fuzz Server:              2014-05-02 05:23:03.738737
End Time of Fuzz Server:                2014-05-02 06:23:09.772632

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.727692307692
Standard Deviation:     0.34724528786

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0]
        Block CVG (bc): [0.44, 0.0, 1.0, 0.81, 0.67, 0.0, 1.0, 1.0, 1.0, 0.05, 0.71,
                        1.0, 0.87, 1.0, 1.0, 1.0, 0.41, 0.0, 0.57, 0.58, 0.59, 0.71,
                        1.0, 1.0, 1.0, 1.0]
        Line CVG (lc):  [0.51, 0.0, 1.0, 0.95, 0.66, 0.0, 1.0, 1.0, 1.0, 0.09, 0.68,
                        1.0, 0.79, 1.0, 1.0, 1.0, 0.39, 0.0, 0.62, 0.8, 0.64, 0.79,
                        1.0, 1.0, 1.0, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.410016112495
Standard Deviation:     0.462111383805


TARGETS:
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        URLCanonicalizer->createParameterMap (String): SortedMap
        Bug Type:       SF_SWITCH_NO_DEFAULT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        TLDList->TLDList (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList-><static initializer>
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->contains (String): boolean
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->getInstance (): TLDList
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        DocIDServer->addUrlAndDocId (String, int): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
```

8.3.9 Evaluation: 90%

```
Individual: [1, 1, 1, 1, 0, 0, 1]

Number of responses from Fuzz Server: 3600
Start Time of Fuzz Server:              2014-05-02 06:00:21.199086
End Time of Fuzz Server:                2014-05-02 07:00:48.076416

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.744193548387
Standard Deviation:     0.334353243698

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0]
        Block CVG (bc): [0.67, 0.0, 1.0, 1.0, 1.0, 0.51, 0.0, 1.0, 0.81, 0.41, 0.0,
                        0.73, 0.81, 0.57, 0.51, 0.59, 0.83, 1.0, 1.0, 1.0, 1.0, 1.0,
                        0.87, 1.0, 1.0, 1.0, 0.86, 0.89, 1.0, 0.0, 1.0]
        Line CVG (lc):  [0.66, 0.0, 1.0, 1.0, 1.0, 0.58, 0.0, 1.0, 0.95, 0.39, 0.0,
                        0.58, 0.7, 0.62, 0.58, 0.64, 0.93, 1.0, 1.0, 1.0, 1.0, 1.0,
                        0.79, 1.0, 1.0, 1.0, 0.84, 0.81, 1.0, 0.0, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.409777484995
Standard Deviation:     0.46229593563


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        DocIDServer->addUrlAndDocId (String, int): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        DocIDServer->getNewDocID (String): int
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        DocIDServer->getDocId (String): int
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        Parser->parse (Page, String): boolean
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->TLDList (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList-><static initializer>
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->contains (String): boolean
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->getInstance (): TLDList
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->createParameterMap (String): SortedMap
        Bug Type:       SF_SWITCH_NO_DEFAULT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        Parser->parse (String): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Parser->parse (InputSource): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
```

8.3.10 Evaluation: 100%

```
Individual: [1, 1, 1, 1, 1, 0, 1]

Number of responses from Fuzz Server: 3600
Start Time of Fuzz Server:              2014-05-02 05:56:22.831720
End Time of Fuzz Server:                2014-05-02 06:56:29.894734

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.7365625
Standard Deviation:     0.331818939474

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0]
        Block CVG (bc): [0.67, 0.0, 1.0, 1.0, 1.0, 0.51, 0.0, 1.0, 0.81, 0.52, 0.41,
                        0.0, 0.73, 0.81, 0.57, 0.51, 0.59, 0.83, 1.0, 1.0, 1.0, 1.0,
                        1.0, 0.87, 1.0, 1.0, 1.0, 0.88, 0.89, 1.0, 0.0, 1.0]
        Line CVG (lc):  [0.66, 0.0, 1.0, 1.0, 1.0, 0.58, 0.0, 1.0, 0.95, 0.5, 0.39,
                        0.0, 0.58, 0.7, 0.62, 0.58, 0.64, 0.93, 1.0, 1.0, 1.0, 1.0,
                        1.0, 0.79, 1.0, 1.0, 1.0, 0.84, 0.81, 1.0, 0.0, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.410829061081
Standard Deviation:     0.462486530038


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        RobotstxtServer->fetchDirectives (URL): HostDirectives
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        DocIDServer->addUrlAndDocId (String, int): void
```

```
    Bug Type:       DM_DEFAULT_ENCODING
    ------------------------------
    DocIDServer->getNewDocID (String): int
    Bug Type:       DM_DEFAULT_ENCODING
    ------------------------------
    DocIDServer->getDocId (String): int
    Bug Type:       DM_DEFAULT_ENCODING
    ------------------------------
    Frontier->getNextURLs (int, List): void
    Bug Type:       UW_UNCOND_WAIT
    ------------------------------
    Parser->parse (Page, String): boolean
    Bug Type:       DM_DEFAULT_ENCODING
    ------------------------------
    HtmlContentHandler->startElement (String, String, String, Attributes): void
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    HtmlContentHandler->endElement (String, String, String): void
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    HtmlContentHandler->HtmlContentHandler (): void
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    HtmlContentHandler->characters (char [], int, int): void
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    HtmlContentHandler->getBaseUrl (): String
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    HtmlContentHandler->getBodyText (): String
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    HtmlContentHandler->getOutgoingUrls (): List
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    TLDList->TLDList (): void
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    TLDList-><static initializer>
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    TLDList->contains (String): boolean
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    TLDList->getInstance (): TLDList
    Bug Type:       SS_SHOULD_BE_STATIC
    ------------------------------
    URLCanonicalizer->getCanonicalURL (String, String): String
    Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
    ------------------------------
    URLCanonicalizer->createParameterMap (String): SortedMap
    Bug Type:       SF_SWITCH_NO_DEFAULT
    ------------------------------
    URLCanonicalizer->getCanonicalURL (String): String
    Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
    ------------------------------
    Parser->parse (String): void
    Bug Type:       DM_DEFAULT_ENCODING
    ------------------------------
    Parser->parse (InputSource): void
    Bug Type:       DM_DEFAULT_ENCODING
    ------------------------------
```

8.4 Best-Fit Exhaustive Evaluation Result Files

8.4.1 Evaluation: 10%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 41964
Start Time of Fuzz Server:            2014-09-21 21:45:51.972000
End Time of Fuzz Server:              2014-09-22 08:44:39.027000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.8625
Standard Deviation:     0.130455931256

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.61, 0.81, 0.88, 1.0]
        Line CVG (lc):  [0.66, 0.95, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.417123726346
Standard Deviation:     0.463769500048


TARGETS:
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.4.2 Evaluation: 20%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 51648
Start Time of Fuzz Server:              2014-09-23 23:21:59.097000
End Time of Fuzz Server:                2014-09-24 12:51:37.142000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.79
Standard Deviation:     0.146021307425

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 0.61, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.69, 0.66, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.430755348566
Standard Deviation:     0.465627993137


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.4.3 Evaluation: 30%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 41964
Start Time of Fuzz Server:              2014-09-24 19:30:29.691000
End Time of Fuzz Server:                2014-09-25 08:02:54.939000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.8425
Standard Deviation:     0.157539677542

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 1.0, 1.0, 0.61, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.67, 1.0, 1.0, 0.66, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.434763498763
Standard Deviation:     0.466277615308


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.4.4 Evaluation: 40%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 35508
Start Time of Fuzz Server:              2014-09-25 00:36:37.656000
End Time of Fuzz Server:                2014-09-25 10:00:36.290000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.702727272727
Standard Deviation:     0.360759071179

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 0.61, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.66, 0.0, 1.0, 1.0, 1.0, 0.0, 0.66, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.416173684977
Standard Deviation:     0.463554349087


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.4.5 Evaluation: 50%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 51648
Start Time of Fuzz Server:              2014-09-26 18:54:22.594000
End Time of Fuzz Server:                2014-09-27 08:14:29.511000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.728333333333
Standard Deviation:     0.354890436927

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.61, 0.81, 0.57, 0.86, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.66, 0.95, 0.62, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.432071376548
Standard Deviation:     0.46576243519


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.4.6 Evaluation: 60%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 50185
Start Time of Fuzz Server:              2014-09-25 17:29:31.728000
End Time of Fuzz Server:                2014-09-26 06:25:02.719000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.785
Standard Deviation:     0.314571136629

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.61, 0.81, 0.41, 0.57, 0.59,
                        0.83, 1.0, 1.0, 1.0, 1.0, 1.0, 0.86, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.66, 0.95, 0.39, 0.62, 0.64,
                        0.93, 1.0, 1.0, 1.0, 1.0, 1.0, 0.84, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.431195858247
Standard Deviation:     0.465824384606


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.4.7 Evaluation: 70%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 41964
Start Time of Fuzz Server:              2014-09-28 18:40:26.709000
End Time of Fuzz Server:                2014-09-29 05:38:28.375000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.81
Standard Deviation:     0.293965984427

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.61, 0.81, 0.41, 0.57, 0.48,
                        0.83, 1.0, 1.0, 1.0, 1.0, 1.0, 0.87, 1.0, 1.0, 1.0, 0.86, 0.9, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.66, 0.95, 0.39, 0.62, 0.52,
                        0.93, 1.0, 1.0, 1.0, 1.0, 1.0, 0.79, 1.0, 1.0, 1.0, 0.84, 0.88, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.416238866988
Standard Deviation:     0.463811381593


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
```

```
------------------------------
HtmlContentHandler->HtmlContentHandler (): void
Bug Type:       SS_SHOULD_BE_STATIC
------------------------------
HtmlContentHandler->characters (char [], int, int): void
Bug Type:       SS_SHOULD_BE_STATIC
------------------------------
HtmlContentHandler->getBaseUrl (): String
Bug Type:       SS_SHOULD_BE_STATIC
------------------------------
HtmlContentHandler->getBodyText (): String
Bug Type:       SS_SHOULD_BE_STATIC
------------------------------
HtmlContentHandler->getOutgoingUrls (): List
Bug Type:       SS_SHOULD_BE_STATIC
------------------------------
TLDList->TLDList (): void
Bug Type:       SS_SHOULD_BE_STATIC
------------------------------
TLDList-><static initializer>
Bug Type:       SS_SHOULD_BE_STATIC
------------------------------
TLDList->contains (String): boolean
Bug Type:       SS_SHOULD_BE_STATIC
------------------------------
TLDList->getInstance (): TLDList
Bug Type:       SS_SHOULD_BE_STATIC
------------------------------
URLCanonicalizer->getCanonicalURL (String, String): String
Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
------------------------------
URLCanonicalizer->createParameterMap (String): SortedMap
Bug Type:       SF_SWITCH_NO_DEFAULT
------------------------------
URLCanonicalizer->getCanonicalURL (String): String
Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
------------------------------
```

8.4.8 Evaluation: 80%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 23672
Start Time of Fuzz Server:              2014-09-30 07:59:13.447000
End Time of Fuzz Server:                2014-09-30 14:02:52.477000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.783461538462
Standard Deviation:     0.324806952224

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.61, 0.81, 0.41, 0.0, 0.57,
                        0.59, 0.83, 1.0, 1.0, 1.0, 1.0, 1.0, 0.87, 1.0, 1.0, 1.0, 0.86,
                        0.9, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.66, 0.95, 0.39, 0.0, 0.62,
                        0.64, 0.93, 1.0, 1.0, 1.0, 1.0, 1.0, 0.79, 1.0, 1.0, 1.0, 0.84,
                        0.88, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.414700642523
Standard Deviation:     0.463457172317


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        DocIDServer->addUrlAndDocId (String, int): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->TLDList (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList-><static initializer>
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->contains (String): boolean
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->getInstance (): TLDList
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->createParameterMap (String): SortedMap
        Bug Type:       SF_SWITCH_NO_DEFAULT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
```

8.4.9 Evaluation: 90%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 49162
Start Time of Fuzz Server:              2014-09-30 17:47:12.697000
End Time of Fuzz Server:                2014-10-01 06:28:15.324000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.74935483871
Standard Deviation:     0.333959003509

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 0.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.61, 0.81, 0.41, 0.0, 0.73,
                        0.81, 0.57, 0.51, 0.59, 0.83, 1.0, 1.0, 1.0, 1.0, 1.0, 0.87, 1.0,
                        1.0, 1.0, 0.86, 0.9, 1.0, 0.0, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.66, 0.95, 0.39, 0.0, 0.58,
                        0.7, 0.62, 0.58, 0.64, 0.93, 1.0, 1.0, 1.0, 1.0, 1.0, 0.79, 1.0,
                        1.0, 1.0, 0.84, 0.88, 1.0, 0.0, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.431667640187
Standard Deviation:     0.465955228591


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        DocIDServer->addUrlAndDocId (String, int): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        DocIDServer->getNewDocID (String): int
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        DocIDServer->getDocId (String): int
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        Parser->parse (Page, String): boolean
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->TLDList (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList-><static initializer>
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->contains (String): boolean
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->getInstance (): TLDList
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->createParameterMap (String): SortedMap
        Bug Type:       SF_SWITCH_NO_DEFAULT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        Parser->parse (String): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Parser->parse (InputSource): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
```

8.4.10 Evaluation: 100%

```
Individual: PD_Creator/protocol.py

Number of responses from Fuzz Server: 49981
Start Time of Fuzz Server:              2014-10-02 19:35:30.960000
End Time of Fuzz Server:                2014-10-03 08:28:20.980000

------------------------------
TARGET INFORMATION

Settings:               (line coverage, method granularity)

Mean Coverage Value:    0.7415625
Standard Deviation:     0.331550423908

Class CVG (cc):         [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Methd CVG (mc):         [1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                        1.0, 1.0, 1.0, 1.0, 0.0, 1.0]
        Block CVG (bc): [0.68, 0.0, 1.0, 1.0, 1.0, 0.52, 0.0, 1.0, 0.61, 0.81, 0.41, 0.0,
                        0.73, 0.81, 0.57, 0.51, 0.59, 0.83, 1.0, 1.0, 1.0, 1.0, 1.0, 0.87,
                        1.0, 1.0, 1.0, 0.88, 0.9, 1.0, 0.0, 1.0]
        Line CVG (lc):  [0.67, 0.0, 1.0, 1.0, 1.0, 0.5, 0.0, 1.0, 0.66, 0.95, 0.39, 0.0,
                        0.58, 0.7, 0.62, 0.58, 0.64, 0.93, 1.0, 1.0, 1.0, 1.0, 1.0, 0.79,
                        1.0, 1.0, 1.0, 0.84, 0.88, 1.0, 0.0, 1.0]

------------------------------
TARGET COMPLEMENT INFORMATION

Mean Coverage Value:    0.431987731853
Standard Deviation:     0.465875201183


TARGETS:
        CrawlController$1->run (): void
        Bug Type:       REC_CATCH_EXCEPTION
        ------------------------------
        Page->setContentData (byte []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        Page->getContentData (): byte []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->getFetchResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        Page->setFetchResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        RobotstxtServer->fetchDirectives (URL): HostDirectives
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        PageFetchResult->getResponseHeaders (): Header []
        Bug Type:       EI_EXPOSE_REP
        ------------------------------
        PageFetchResult->setResponseHeaders (Header []): void
        Bug Type:       EI_EXPOSE_REP2
        ------------------------------
        PageFetcher->fetchHeader (WebURL): PageFetchResult
        Bug Type:       SWL_SLEEP_WITH_LOCK_HELD
        ------------------------------
        IdleConnectionMonitorThread->shutdown (): void
        Bug Type:       NN_NAKED_NOTIFY
        ------------------------------
        Counters->setValue (String, long): void
        Bug Type:       DM_NUMBER_CTOR
        ------------------------------
        DocIDServer->addUrlAndDocId (String, int): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        DocIDServer->getNewDocID (String): int
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        DocIDServer->getDocId (String): int
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Frontier->getNextURLs (int, List): void
        Bug Type:       UW_UNCOND_WAIT
        ------------------------------
        Parser->parse (Page, String): boolean
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        HtmlContentHandler->startElement (String, String, String, Attributes): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->endElement (String, String, String): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->HtmlContentHandler (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->characters (char [], int, int): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBaseUrl (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getBodyText (): String
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        HtmlContentHandler->getOutgoingUrls (): List
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->TLDList (): void
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList-><static initializer>
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->contains (String): boolean
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        TLDList->getInstance (): TLDList
        Bug Type:       SS_SHOULD_BE_STATIC
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String, String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        URLCanonicalizer->createParameterMap (String): SortedMap
        Bug Type:       SF_SWITCH_NO_DEFAULT
        ------------------------------
        URLCanonicalizer->getCanonicalURL (String): String
        Bug Type:       ES_COMPARING_STRINGS_WITH_EQ
        ------------------------------
        Parser->parse (String): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
        Parser->parse (InputSource): void
        Bug Type:       DM_DEFAULT_ENCODING
        ------------------------------
```

Bibliography

[1] Sap analysis: Open source static analysis tools for security testing of java web applications.

[2] Introduction to evolutionary computing techniques. In *Electronic Technology Directions to the Year 2000, 1995. Proceedings.*, pages 122–127, May 1995.

[3] Ieee standard–adoption of iso/iec 15026-2:2011 systems and software engineering–systems and software assurance–part 2: Assurance case. *IEEE Std 15026-2-2011*, pages 1–28, 2011.

[4] Isaac Agudo, José L Vivas, and Javier López. Security assurance during the software development cycle. In *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, page 20. ACM, 2009.

[5] Chang Wook Ahn and Rudrapatna S Ramakrishna. Elitism-based compact genetic algorithms. *Evolutionary Computation, IEEE Transactions on*, 7(4):367–385, 2003.

[6] Dave Aitel. The advantages of block-based protocol analysis for security testing. *Immunity Inc., February*, 2002.

[7] Pedram Amini and Aaron Portnoy. Sulley: A pure python fully-automated and unattended fuzzing framework, May 2013.

[8] T Scott Ankrum and Alfred H Kromholz. Structured assurance cases: Three common standards. In *High-Assurance Systems Engineering, 2005. HASE 2005. Ninth IEEE International Symposium on*, pages 99–108. IEEE, 2005.

[9] Nathaniel Ayewah, David Hovemeyer, J David Morgenthaler, John Penix, and William Pugh. Using static analysis to find bugs. *Software, IEEE*, 25(5):22–29, 2008.

[10] James Edward Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the international conference on genetic algorithms and their applications*, pages 101–111, 1985.

[11] Thomas Ball. The concept of dynamic analysis. In *Software EngineeringESEC/FSE99*, pages 216–234. Springer, 1999.

[12] Richard K Belew, John McInerney, and Nicol N Schraudolph. Evolving networks: Using the genetic algorithm with connectionist learning. In *In*. Citeseer, 1990.

[13] Steven M Bellovin. On the brittleness of software and the infeasibility of security metrics. *Security & Privacy, IEEE*, 4(4):96–96, 2006.

[14] Robin E Bloomfield, Sofia Guerra, Ann Miller, Marcelo Masera, and Charles B Weinstock. International working group on assurance cases (for security). *Security & Privacy, IEEE*, 4(3):66–68, 2006.

[15] Cristian Cadar, Daniel Dunbar, and Dawson R Engler. Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In *OSDI*, volume 8, pages 209–224, 2008.

[16] Joseph P Cavano and James A McCall. A framework for the measurement of software quality. In *ACM SIGMETRICS Performance Evaluation Review*, volume 7, pages 133–139. ACM, 1978.

[17] Brian Chess and Gary McGraw. Static analysis for security. *Security & Privacy, IEEE*, 2(6):76–79, 2004.

[18] Toby Clarke. Fuzzing for software vulnerability discovery. *Department of Mathematic, Royal Holloway, University of London, Tech. Rep. RHUL-MA-2009-4*, 2009.

[19] Toby Clarke and Jason Crampton. Fuzzingor how to help computers cope with the unexpected.

[20] Lukasz Cyra and Janusz Górski. Expert assessment of arguments: A method and its experimental evaluation. In *Computer Safety, Reliability, and Security*, pages 291–304. Springer, 2008.

[21] Kusum Deep and Hadush Mebrahtu. Combined mutation operators of genetic algorithm for the travelling salesman problem. *International Journal of Combinatorial Optimization Problems & Informatics*, 2(3), 2011.

[22] Jared DeMott. The evolving art of fuzzing. *DEF CON*, 14, 2006.

[23] Dumitru Dumitrescu, Beatrice Lazzerini, Lakhmi C Jain, and Anca Dumitrescu. *Evolutionary computation*, volume 18. CRC press, 2000.

[24] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys (CSUR)*, 44(2):6, 2012.

[25] Agosten E Eiben and James E Smith. *Introduction to evolutionary computing*, volume 2. Springer Berlin, 2010.

[26] Agoston E Eiben, Zbigniew Michalewicz, Marc Schoenauer, and JE Smith. Parameter control in evolutionary algorithms. In *Parameter setting in evolutionary algorithms*, pages 19–46. Springer, 2007.

[27] Michael D Ernst. Static and dynamic analysis: Synergy and duality. In *WODA 2003: ICSE Workshop on Dynamic Analysis*, pages 24–27. Citeseer, 2003.

[28] David Evans and David Larochelle. Improving security using extensible lightweight static analysis. *software, IEEE*, 19(1):42–51, 2002.

[29] Vladimir Filipović. Fine-grained tournament selection operator in genetic algorithms. *Computing and Informatics*, 22(2):143–161, 2012.

[30] Elizabeth Fong, Michael Kass, Thomas Rhodes, and Frederick Boland. Structured assurance case methodology for assessing software trustworthiness. In *Secure Software Integration and Reliability Improvement Companion (SSIRI-C), 2010 Fourth International Conference on*, pages 32–33. IEEE, 2010.

[31] Justin E Forrester and Barton P Miller. An empirical study of the robustness of windows nt applications using random testing. In *Proceedings of the 4th USENIX Windows System Symposium*, pages 59–68, 2000.

[32] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. Deap: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, 2012.

[33] Vijay Ganesh, Tim Leek, and Martin Rinard. Taint-based directed whitebox fuzzing. In *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on*, pages 474–484. IEEE, 2009.

[34] Y Ganjisaffar. crawler4j - open source web crawler for java, 2012.

[35] David A Garvin. What does product quality really mean. *Sloan management review*, 26(1):25–43, 1984.

[36] Mitsuo Gen and Runwei Cheng. *Genetic algorithms and engineering optimization*, volume 7. John Wiley & Sons, 2000.

[37] Patrice Godefroid, Nils Klarlund, and Koushik Sen. Dart: directed automated random testing. In *ACM Sigplan Notices*, volume 40, pages 213–223. ACM, 2005.

[38] Patrice Godefroid, Michael Y Levin, David Molnar, et al. Automated whitebox fuzz testing. NDSS, 2008.

[39] Karen M Goertzel, Theodore Winograd, Holly L McKinley, Lyndon J Oh, Michael Colon, Thomas McGibbon, Elaine Fedchak, and Robert Vienneau. Software security assurance: A state-of-art report (sar). Technical report, DTIC Document, 2007.

[40] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. *Urbana*, 51:61801–2996, 1991.

[41] David E Goldberg, Kalyanmoy Deb, and James H Clark. Genetic algorithms, noise, and the sizing of populations. *Complex systems*, 6:333–362, 1991.

[42] John Goodenough, Howard Lipson, and Chuck Weinstock. Arguing security-creating security assurance cases. *rapport en ligne (initiative build security-in du US CERT), Université Carnegie Mellon*, 2007.

[43] Erik D. Goodman. Introduction to genetic algorithms. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, GECCO '07, pages 3205–3224, New York, NY, USA, 2007. ACM.

[44] Liu Guang-Hong, Wu Gang, Zheng Tao, Shuai Jian-Mei, and Tang Zhuo-Chun. Vulnerability analysis for x86 executables using genetic algorithm and fuzzing. In *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*, volume 2, pages 491–497. IEEE, 2008.

[45] Jon G Hall and Lucia Rapanotti. Assurance-driven design. In *Software Engineering Advances, 2008. ICSEA'08. The Third International Conference on*, pages 379–388. IEEE, 2008.

[46] John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[47] Michael Howard and Steve Lipner. *The security development lifecycle*, volume 11. Microsoft Press, 2009.

[48] ISO IEC. Iec 9126-1: Software engineering-product quality-part 1: Quality model. *Geneva, Switzerland: International Organization for Standardization (ISO)*, 2001.

[49] ISO IEC. Iso/iec 21827:2008, information technology – security techniques – systems security engineering – capability maturity model (sse-cmm). *Geneva, Switzerland: International Organization for Standardization (ISO)*, 2008.

[50] ISO IEC. Iec 25010: 2011, systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models. *International Organization for Standardization (ISO)*, 2011.

[51] ISO IEC. Iso/iec 15408 version 3.1: 2012, common criteria (cc) for information technology security evaluation. *Geneva, Switzerland: International Organization for Standardization (ISO)*, 2012.

[52] Lester Ingber. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

[53] Vincenzo Iozzo. 0-knowledge fuzzing. 2010.

[54] Martin Gilje Jaatun. Hunting for aardvarks: Can software security be measured? In *Multidisciplinary Research and Practice for Information Systems*, pages 85–92. Springer, 2012.

[55] Wayne Jansen. *Directions in security metrics research.* DIANE Publishing, 2010.

[56] Andrew Jaquith. *Security metrics: replacing fear, uncertainty, and doubt.* Addison-Wesley Professional, 2007.

[57] Leon Juranić. Using fuzzing to detect security vulnerabilities. *Retrieved Apr*, 26:2012, 2006.

[58] Tim Kelly and Rob Weaver. The goal structuring notation–a safety argument notation. In *Proceedings of the dependable systems and networks 2004 workshop on assurance cases.* Citeseer, 2004.

[59] Rainer Koelle and M Hawley. Sesar security 2020: How to embed and assure security in system-of-systems engineering? In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2012*, pages E8–1. IEEE, 2012.

[60] Steve Lipner. The trustworthy computing security development lifecycle. In *Computer Security Applications Conference, 2004. 20th Annual*, pages 2–13. IEEE, 2004.

[61] Howard Lipson and Chuck Weinstock. Evidence of assurance: Laying the foundation for a credible security case. *rapport en ligne (initiative build security-in du US CERT), Université Carnegie Mellon*, 2008.

[62] Branko Marović, Marcin Wrzos, Marek Lewandowski, Andrzej Bobak, Tomasz Krysztofiak, Rade Martinović, Spass Kostov, Szymon Kupinski, Stephan Kraft DFN, Ann Harding, et al. Gn3 quality assurance best practice guide 4.0.

[63] Andrew Marshall, Michael Howard, Grant Bugher, Brian Harden, Charlie Kaufman, Martin Rues, and Vittorio Bertocci. Security best practices for developing windows azure applications. *Microsoft Corp*, 2010.

[64] Thomas J. McCabe. A complexity measure. *Software Engineering, IEEE Transactions on*, (4):308–320, 1976.

[65] Jim A McCall, Paul K Richards, and Gene F Walters. *Factors in software quality.* General Electric, National Technical Information Service., 1977.

[66] Gary McGraw, Sammy Migues, and Jacob West. Building security in maturity model (bsimm4), 2012.

[67] Samuel A Merrell, Andrew P Moore, and James F Stevens. Goal-based assessment for the cybersecurity of critical infrastructure. In *Technologies for Homeland Security (HST), 2010 IEEE International Conference on*, pages 84–88. IEEE, 2010.

[68] Ann Miller and Krishna Mohan Moleyar. Arguing security using a probabilistic risk assessment model. In *DSN 2007 Workshop on Assurance Cases for Security–The Metrics Challenge*, 2007.

[69] Barton P Miller, Louis Fredriksen, and Bryan So. An empirical study of the reliability of unix utilities. *Communications of the ACM*, 33(12):32–44, 1990.

[70] Barton Paul Miller, David Koski, Cjin Pheow Lee, Vivekananda Maganty, Ravi Murthy, Ajitkumar Natarajan, and Jeff Steidl. *Fuzz revisited: A re-examination of the reliability of UNIX utilities and services.* University of Wisconsin-Madison, Computer Sciences Department, 1995.

[71] Brad L Miller and David E Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3):193–212, 1995.

[72] Markus Mock. Dynamic analysis from the bottom up. In *WODA 2003 ICSE Workshop on Dynamic Analysis*, page 13. Citeseer, 2003.

[73] Nachiappan Nagappan and Thomas Ball. Static analysis tools as early indicators of pre-release defect density. In *Proceedings of the 27th international conference on Software engineering*, pages 580–586. ACM, 2005.

[74] Sagar Naik and Piyu Tripathy. *Software testing and quality assurance: theory and practice.* Wiley-Spektrum, 2011.

[75] Peter Oehlert. Violating assumptions with fuzzing. *Security & Privacy, IEEE*, 3(2):58–62, 2005.

[76] Moussa Ouedraogo, Haralambos Mouratidis, Djamel Khadraoui, and Eric Dubois. An agent-based system to support assurance of security requirements. In *Secure Software Integration and Reliability Improvement (SSIRI), 2010 Fourth International Conference on*, pages 78–87. IEEE, 2010.

[77] Bhanuchander Reddy Poreddy and Steven Corns. Arguing security of generic avionic mission control computer system (mcc) using assurance cases. *Procedia Computer Science*, 6:499–504, 2011.

[78] Thomas Rhodes, Frederick Boland, Elizabeth Fong, and Michael Kass. Software assurance using structured assurance case models. *Journal of Research of the NIST*, 115(3):209, 2010.

[79] Linda H Rosenberg and Lawrence E Hyatt. Software quality metrics for object-oriented environments. *Crosstalk Journal, April*, 1997.

[80] Vlad Roubtsov et al. Emma, a free java code coverage tool, 2005.

[81] John Rushby. Runtime certification. In *Runtime Verification*, pages 21–35. Springer, 2008.

[82] James C Spall. A stochastic approximation technique for generating maximum likelihood parameter estimates. In *American Control Conference, 1987*, pages 1161–1167. IEEE, 1987.

[83] James C Spall. A stochastic approximation algorithm for large-dimensional systems in the kiefer-wolfowitz setting. In *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, pages 1544–1548. IEEE, 1988.

[84] James C Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

[85] M Srinivas and Lalit M Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(4):656–667, 1994.

[86] Mandavilli Srinivas and Lalit M Patnaik. Genetic algorithms: A survey. *Computer*, 27(6):17–26, 1994.

[87] Michael Sutton, Adam Greene, and Pedram Amini. *Fuzzing: Brute force vulnerability discovery.* Addison-Wesley Professional, 2007.

[88] Ari Takanen, Jared D Demott, and Charles Miller. *Fuzzing for software security testing and quality assurance.* Artech House on Demand, 2008.

[89] Dirk Thierens and David Goldberg. Convergence models of genetic algorithm selection schemes. In *Parallel problem solving from naturePPSN III*, pages 119–129. Springer, 1994.

[90] José Luis Vivas, Isaac Agudo, and Javier López. A methodology for security assurance-driven system development. *Requirements Engineering*, 16(1):55–73, 2011.

[91] Tielei Wang, Tao Wei, Guofei Gu, and Wei Zou. Taintscope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 497–512. IEEE, 2010.

[92] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[93] L Darrell Whitley. *Fundamental principles of deception in genetic search.* Citeseer, 1991.

[94] Zhiyong Wu, J William Atwood, and Xueyong Zhu. A new fuzzing technique for software vulnerability mining. *Proceedings of the IEEE CONSEG*, 9, 2009.

[95] Qian Yang, J Jenny Li, and David M Weiss. A survey of coverage-based testing tools. *The Computer Journal*, 52(5):589–597, 2009.

[96] Peter C Young. *Recursive estimation and time-series analysis: an introduction for the student and practitioner.* Springer, 2011.

[97] Andreas Zeller. Program analysis: A hierarchy. In *Proceedings of the ICSE Workshop on Dynamic Analysis (WODA 2003)*, pages 6–9. Citeseer, 2003.

[98] Xingyu Zhao, Dajian Zhang, Minyan Lu, and Fuping Zeng. A new approach to assessment of confidence in assurance cases. In *Computer Safety, Reliability, and Security*, pages 79–91. Springer, 2012.

[99] Misha Zitser, Richard Lippmann, and Tim Leek. Testing static analysis tools using exploitable buffer overflows from open source code. In *ACM SIGSOFT Software Engineering Notes*, volume 29, pages 97–106. ACM, 2004.