Cost-effective Detection of Drive-by-Download Attacks with Hybrid Client Honeypots

by

Christian Seifert

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2010

Abstract

With the increasing connectivity of and reliance on computers and networks, important aspects of computer systems are under a constant threat. In particular, drive-by-download attacks have emerged as a new threat to the integrity of computer systems. Drive-by-download attacks are client-side attacks that originate from web servers that are visited by web browsers. As a vulnerable web browser retrieves a malicious web page, the malicious web server can push malware to a user's machine that can be executed without their notice or consent.

The detection of malicious web pages that exist on the Internet is prohibitively expensive. It is estimated that approximately 150 million malicious web pages that launch drive-by-download attacks exist today. So-called high-interaction client honeypots are devices that are able to detect these malicious web pages, but they are slow and known to miss attacks. Detection of malicious web pages in these quantities with client honeypots would cost millions of US dollars.

Therefore, we have designed a more scalable system called a hybrid client honeypot. It consists of lightweight client honeypots, the so-called low-interaction client honeypots, and traditional high-interaction client honeypots. The lightweight low-interaction client honeypots inspect web pages at high speed and forward only likely malicious web pages to the high-interaction client honeypot for a final classification.

For the comparison of client honeypots and evaluation of the hybrid client honeypot system, we have chosen a cost-based evaluation method: the true positive cost curve (TPCC). It allows us to evaluate client honeypots against their primary purpose of identification of malicious web

pages. We show that costs of identifying malicious web pages with the developed hybrid client honeypot systems are reduced by a factor of nine compared to traditional high-interaction client honeypots.

The five main contributions of our work are:

- High-Interaction Client Honeypot The first main contribution of our work is the design and implementation of a high-interaction client honeypot Capture-HPC. It is an open-source, publicly available client honeypot research platform, which allows researchers and security professionals to conduct research on malicious web pages and client honeypots. Based on our client honeypot implementation and analysis of existing client honeypots, we developed a component model of client honeypots. This model allows researchers to agree on the object of study, allows for focus of specific areas within the object of study, and provides a framework for communication of research around client honeypots.

- True Positive Cost Curve As mentioned above, we have chosen a cost-based evaluation method to compare and evaluate client honeypots against their primary purpose of identification of malicious web pages: the true positive cost curve. It takes into account the unique characteristics of client honeypots, speed, detection accuracy, and resource cost and provides a simple, cost-based mechanism to evaluate and compare client honeypots in an operating environment. As such, the TPCC provides a foundation for improving client honeypot technology. The TPCC is the second main contribution of our work.

- Mitigation of Risks to the Experimental Design with HAZOP - Mitigation of risks to internal and external validity on the experimental design using hazard and operability (HAZOP) study is the third main contribution. This methodology addresses risks to intent (internal validity) as well as generalizability of results beyond the experimental setting (external validity) in a systematic and thorough

manner.

- Low-Interaction Client Honeypots - Malicious web pages are usually part of a malware distribution network that consists of several servers that are involved as part of the drive-by-download attack. Development and evaluation of classification methods that assess whether a web page is part of a malware distribution network is the fourth main contribution.

- Hybrid Client Honeypot System - The fifth main contribution is the hybrid client honeypot system. It incorporates the mentioned classification methods in the form of a low-interaction client honeypot and a high-interaction client honeypot into a hybrid client honeypot system that is capable of identifying malicious web pages in a cost effective way on a large scale. The hybrid client honeypot system outperforms a high-interaction client honeypot with identical resources and identical false positive rate.

For my wife, Megan

Acknowledgements

Numerous individuals and organizations have supported me throughout my studies. I would like to express my sincere gratitude towards them. Without them, this work would not have been possible.

Foremost, I would like to thank my two supervisors, Dr. Peter Komisarczuk and Dr. Ian Welch. They provided me with great guidance and advice throughout my studies. I especially thank them for their encouragement to collaborate with various institutions and individuals. Ramon Steenson, David Stirling, Vipul Delwadia, Florant Mara, Radu Muschevici, and Sebastian Krueger have assisted in client honeypot development and operation, analysis of malicious web pages, exploration of related research questions, etc. They have been a great help and it has been a pleasure to work with them.

Mark Davies, the technical manager of the School of Engineering and Computer Science, has been a key figure enabling my research in the area of computer security. He assisted in setting up a security lab at the school that allowed me to deploy and operate honeypots and analyze malicious web pages.

Further, I would like to thank Dr. Barbara Endicott-Popovsky, Dr. Deborah Frincke, and Dave Dittrich. These individuals have been key in introducing me to and sparking my interest in the field of computer security. Dr. Niels Provos has been an inspiration in my studies. His pioneering work with Honeyd and openness in sharing research results from a corporate setting have greatly encouraged me. Niels has been highly approach-

able and responsive to my questions and requests for advice.

I would especially expand my thanks to Dr. Barbara Endicott-Popovsky. As mentioned, she was a key figure in selecting my research area of computer security. During my PhD studies, she invited me as a visiting scholar to the University of Washington, which greatly enriched my studies. I am grateful for her guidance, encouragement, and advice whenever I sought it.

During my visiting scholar appointment at the University of Washington, I came into contact with two great individuals, Chiraag Aval and Julia Narvaez. Chiraag and Julia did a fantastic job familiarizing themselves with the research matter and exploring research questions themselves. Their enthusiasm for this research area has helped keep me motivated and focused throughout the years.

In the summer of 2008, I had the opportunity to validate and expand my research in a corporate setting. Microsoft Corporation invited me to a summer internship. I especially would like to thank Andrey Zaytsev, Gayathri Venkataraman, Sasi Parthasarathy, and Sarmad Fayyaz for this opportunity. Continuation of the internship in the form of a full-time position with Microsoft allows me to continue to explore and solve problems in the detection of malicious web pages.

My wife, Megan Gleason, has been a key figure keeping me motivated and focused throughout the years. Her open ear to my security monologues and acceptance of the security measures on the home network have been greatly appreciated. Her feedback and proofreading service on numerous papers and articles have earned her an honorary PhD.

Barbara Azzato's technical editing services have greatly enhanced the clarity of several papers. Her feedback and advice have significantly improved my writing style.

I'd like to thank several subject matter experts from a variety of organizations with whom I had the pleasure of discussing the research area of client honeypots and detection of malicious web pages. These discussions

were a great motivator and illustrated the collaborative need to solve these problems. These individuals are Iain Mulholland from HauteSecure, Xeno Kovah, Darien Kindlund, Kathy Wang from the MITRE organization, Piotr Kijewski from Polish Cert and the SpiderMonkey project, and Dr. Jose Nazario from Arbor Networks.

In addition, the Honeynet Project has been a great support for discussing ideas with like-minded individuals and sharing research results and tools with the support of a credible, known international research organization. In particular, I would like to thank Lance Spitzner for his continued encouragement and open arms. The Honeynet Project opened the door to collaboration with the great minds of Dave Dittrich, David Watson, Jamie Riden, and Dr. Thorsten Holz.

Over the course of my studies, my family and friends have been extremely supportive and very understanding of my isolation during the write-up phase.

Finally, I would like to thank the New Zealand government, InternetNZ, and Snort for their financial support throughout my studies.

Contents

List of Figures

Chapter 1

Introduction

With the increasing connectivity of and reliance on computers and networks, important aspects of computer-related systems, namely confidentiality, integrity and availability, are under a constant threat. Confidential data, such as credit card numbers, is stolen [140]; office desktop computers are abused to send email spam [77]; and a power grid outage is caused by a denial-of-service attack on the underlying power grid computer network [42]. All are examples of what could happen when computer security measures fail to protect those aspects.

A particular type of attack that has emerged in recent years is the client-side attack [2]. These attacks target clients. As the client accesses a malicious server, the server delivers an attack to the client as part of the server's response to a client request. Common examples of these attacks are web servers that attack web browsers. As the web browser requests content from a web server, the server returns a malicious page that launches, for example, a so-called drive-by-download attack on the browser. If successful, the web server can push and execute arbitrary programs on the client machine.

High-interaction client honeypots are security devices that are able to find these malicious web servers on a network. However, they have not been suitable for an investigation of malicious web servers on a large scale,

because their slow speed makes them prohibitively expensive on a larger scale. This thesis describes improvements upon high-interaction client honeypots that allow us to collect and analyze a large sample of malicious web pages in a more cost-effective manner.

With the acquired knowledge of malicious web pages, new alternative and faster detection techniques are developed. They are combined into a hybrid client honeypot system that is suitable to quickly detect the majority of malicious web pages. The hybrid client honeypot system is more cost-effective allowing the operation of such a system on a large scale.

The remainder of this chapter is structured as follows. Section 1.1 introduces the thesis of this dissertation. Section 1.2 provides the motivation for the thesis and Section 1.3 summarizes the contributions made by this thesis. Section 1.4 provides an overview of the thesis itself; Section 1.5 presents research discussed in this thesis that has appeared in other publications.

1.1 Thesis

The goal of this work is to develop a client honeypot system that is capable of identifying malicious web pages on a large scale in a more cost-effective and forensically sound manner than is possible with existing client honeypot technology today.

Improved high-interaction client honeypots and a methodology that addresses risks to internal and external validity can be used to obtain forensically sound information about malicious web pages in a more cost-effective manner. Utilizing knowledge about malicious web pages, new alternative and faster detection methods can be developed. These methods could be based on statistical static and dynamic behavioral detection techniques. If these methods are applied to a lightweight, so-called low-interaction client honeypot, these methods can be faster than high-interaction client honeypots, but could possibly exhibit lower detection

accuracy. A hybrid client honeypot system is able to combine the advantages of low- and high-interaction client honeypots into one system, which is capable to identifying malicious web pages on a large scale in a more cost-effective and forensically sound manner.

1.2 Motivation

Client-side attacks are a serious threat because of two main reasons: First, the expectation of an attack occurring via this attack vector is low. A web browser retrieves a web page that is entirely constructed of text (HTML). Web pages are very common and web browsers that are used to retrieve web pages have existed for a long time. The fact that a threat can exist on these web pages that permits a web server to gain complete control of the client is a foreign thought. However, new vulnerabilities in web browsers are regularly disclosed and new exploits keep appearing that permit these attacks to occur [124]. Second, the danger of malicious web pages comes from the fact that very little user interaction is necessary to become compromised. An attack is covert and a simple click on a URL that points to a malicious web page is sufficient to trigger the attack. If the attack is successful, the user will not be required to consent to any malicious action nor will the user notice an attack has occurred.

As described, client-side attacks are a severe threat in themselves, but are extremely dangerous if viewed in the context of self-propagating code when combined with traditional server-side attacks. A so-called contagion worm spreads from clients (e.g., web browsers) to servers (e.g., web servers) and vice versa on existing network traffic like a contagious disease [136]. Since no abnormal network traffic patterns and volume are generated, such a contagion worm is very difficult to detect and could potentially subvert millions of machines.

Traditional defenses, such as antivirus software and firewalls, which are adopted by the majority of corporations and institutions [115], are

ineffective to protect against the threat of client-side attacks. Antivirus technology, which primarily employs signature-based mechanisms, fails to cope with the volume of malware [127]. And firewall technology, designed to block or permit traffic, does not impact malicious web traffic once web traffic is permitted in general.

A good mechanism to protect against client-side attacks is patching [159], which applies a small piece of software to fix the vulnerability on the client that originally allowed a client-side attack to take place. Because the majority of client-side attacks make use of known exploits, patching the operating system, web browser and plug-ins is usually a good strategy to defend against these attacks. However, a recent study examined vulnerable online web browser populations and estimated at least 45% of users did not use the most secure web browser version when accessing web sites; investigating browser plug-ins is estimated to reveal a larger problem [43]. But even if all users apply patches rigorously, the possibility of exposure to attacks that target vulnerabilities, so-called zero-day attacks for which patches are simply not available, still exists.

Alternatively to patching, one can block the user from navigating to a malicious web page if it is known to be malicious. All major search engines [48, 172, 126], some web browsers [51, 97], and some web browser plug-ins [60, 82] take this approach. This approach can protect patched and unpatched browsers, as well as protect against zero-days and older exploits. However, for this approach to be successful, one needs to know about malicious web pages and, considering the estimated size of the problem, this is a challenging task.

Client-side attacks are a large problem in absolute terms. An average estimate based on existing studies on the prevalence of web pages that launch drive-by-download attacks is approximately 0.2% [83, 96, 159]. According to a study in January 2005, 11.5 billion publicly indexable web pages exist [58]. According to Netcraft, approximately 9,800,000 web sites existed at that time, resulting in about 1,173 pages/site on average. Since

January 2005, the Internet has grown significantly. Netcraft reports 66 million web sites in July 2008 [100]. Assuming the number of pages per site has not changed, the Internet consists of approximately 77 billion indexable web pages. If the estimated percentage of malicious web pages is applied, it results in approximately 150 million malicious web pages, a considerable threat that a user of the Internet is exposed to.

To detect 150 million malicious web pages is challenging in itself. However, if one takes into account that web pages frequently change, the task increases in difficulty. As we will show in chapter 4, a cost to identify a malicious web page with a high-interaction client honeypots can be as high as 0.30 US dollars (based on sequential algorithm and a base rate of $p = 0.04$). If such cost is assumed, the cost to identify 150 million malicious web pages would be approximately 45 million US dollars. Considering the rate of change on the Internet, repeated identification may become necessary, further increasing the cost. Reducing the cost of identification of malicious web pages in a forensically sound manner is the goal of this work.

1.3 Contributions

This thesis makes five main contributions: 1. development of an open-source high-interaction client honeypot; 2. development and application of a cost-based evaluation method on client honeypots and improvements on client honeypots: the true positive cost curve; 3. application of a hazard and operability study on the experimental design to mitigate risks to internal and external validity; 4. development and evaluation of low-interaction client honeypots that can assess whether a web page belongs to a malware distribution network; and 5. development and evaluation of a cost-effective hybrid client honeypot system that combines a high-interaction client honeypot with low-interaction client honeypots. Each of these contributions is discussed below.

1.3.1 High-Interaction Client Honeypot

The design and implementation of a high-interaction client honeypot Capture-HPC, a client honeypot research platform, is a main contribution of this thesis. While three high-interaction client honeypots existed when this work commenced [96, 157, 159], these systems either were not publicly available or did not meet the resource and forensic requirements necessary to conduct research on malicious web servers in a cost-effective manner. As a result, based on forensic requirements we have developed, a new open-source high-interaction client honeypot, named Capture-HPC, was created, which allows researchers and security professionals to conduct research on malicious web pages and client honeypots. Capture-HPC has been incorporated into the other available open-source high-interaction client honeypot, HoneyClient [157], and is being used in numerous research and commercial projects [135, 160, 46, 27, 171].

Based on our client honeypot implementation and analysis of existing client honeypots, we developed a component model of client honeypots. We identified three core components of a client honeypot: Queuer, Visitor, Analysis Engine. This model allows researchers to agree on the object of study, allows for focus of specific areas within the object of study, and provides a framework for communication of research around client honeypots. As increased understanding results from this model, it allows for improved design and development of client honeypot technology. This model has been accepted as a client honeypot model by the research community [116, 169, 33, 142]. It is further discussed in Chapter 4.

1.3.2 True Positive Cost Curve

The true positive cost curve (TPCC) is the second main contribution of this thesis. The TPCC is a method that takes into account the unique characteristics of client honeypots – speed, detection accuracy, and resource cost – and provides a simple, cost-based mechanism for evaluating and com-

paring client honeypots in an operating environment. As such, the TPCC provides a foundation for improving client honeypot technology.

The applicability of the TPCC in evaluating client honeypots is demonstrated through improvements to client honeypot visitation algorithms developed by us and Wang et al. [159]: the bulk, bulk & sequential, and divide-and-conquer algorithms. The TPCC showed that the performance of the bulk algorithm is generally more cost-effective than the other visitation algorithms; however, under certain conditions, namely a low base rate, the divide-and-conquer algorithm outperforms all other visitation algorithms.

TPCC evaluates a client honeypot in an operating environment. As such, the TPCC may also be used by an operator to evaluate different configurations and settings of the client honeypot within a specific operating environment; in other words, the TPCC can be used to tune a client honeypot in a specific operating environment. Application of the TPCC in such a way is demonstrated by tuning a client honeypot in an operating environment with malicious web pages that employ time bombs or IP tracking functionality.

1.3.3 Mitigation of Risks to the Experimental Design with HAZOP

Mitigation of risks to internal and external validity on the experimental design using hazard and operability (HAZOP) study is the third main contribution of this thesis. This methodology addresses risks to intent (internal validity) as well as generalizability of results beyond the experimental setting (external validity) in a systematic and thorough manner.

Measurement studies are used to illustrate the process of HAZOP. A major risk identified is uncontrolled variables. We use uncontrolled variables as an example to illustrate the impact of failure to mitigate risks appropriately. First, it is shown that the URL source can greatly impact mea-

surements; second, it is shown that time can also have a major impact on measurement.

1.3.4 Low-Interaction Client Honeypots

Malicious web pages are usually part of a malware distribution network that consists of several servers that are involved as part of the drive-by-download attack. Development and evaluation of classification methods that can be incorporated into a low-interaction client honeypot network is the fourth main contribution. These methods are used to assess whether a web page is part of a malware distribution network. In contrast to the high-interaction client honeypot, one would not have to load the web pages in a dedicated system nor monitor the system for unauthorized state changes. Rather, a simulated client could be used to retrieve the web page and the server response analyzed directly. The two methods are based on analyzing the dynamic behavior when loading a web page and statistical analysis of elements found on the page. As shown in this thesis, the methods can be used to identify malicious web pages quickly; however, at the same time, many false alerts would be generated.

1.3.5 Hybrid Client Honeypot System

The fifth main contribution of this thesis is the hybrid client honeypot system. A model is developed that is capable of optimizing resources and estimating cost and detection accuracy of a hybrid client honeypot system based on the underlying low- and high-interaction client honeypot components. The hybrid client honeypot system is capable of identifying malicious web pages in a cost-effective way on a large scale. The hybrid client honeypot system outperforms a high-interaction client honeypot with identical resources and identical false positive rate.

The model allows assessment of whether low-interaction client honeypots can be beneficial when combined into a hybrid client honeypot sys-

tem. Two candidate low-interaction client honeypots, based on statistical static and dynamic behavioral methods, are evaluated. The hybrid client honeypot model is used to identify a low-interaction client honeypot component that could be combined into a beneficial hybrid client honeypot system.

The model is evaluated with an actual implementation of a hybrid client honeypot system.

1.4 Overview

The remainder of this thesis is structured as follows.

Chapter 2 presents background information on client-side attacks and attack detection. The first part of this chapter presents generic mechanisms and definitions around attack detection. The second part describes web-based client-side attacks, and the last part of the chapter focuses on the web-based client-side attack that is the object of this thesis: the drive-by-download attack.

Chapter 3 places the work presented in the context of related work in the field. Existing studies on malicious web servers and drive-by-download attacks are reviewed, showing that these attacks are an increasing problem. The second section reviews defensive intrusion prevention strategies and shows that generic strategies are ineffective to counter the threat of drive-by-download attacks. While our work focuses on the task of identification of malicious web servers, research on defensive techniques exists and is also presented. Detection itself, however, is challenging. Intrusion detection techniques are ineffective to detect drive-by-download attacks and a more able detection mechanism is needed: client honeypots. Section 3.3 reviews existing detection technology with a focus on client honeypots. Chapter 3 concludes with a discussion of gaps from the related work that are addressed by this work.

The true positive cost curve, a cost-based method for evaluating high-

interaction client honeypots is the focus of Chapter 4. The TPCC is the foundation for making improvements to client honeypot technology. Two new algorithms are presented that aim at improving the detection speed of client honeypots. Several visitation algorithms are evaluated with the TPCC. The last part of Chapter 4 illustrates that the TPCC can not only be used to evaluate client honeypots, but also be used to tune client honeypots in an operating environment.

Chapter 5 develops a methodology that is designed to reduce the risk to internal and external validity on the experimental design. The methodology is developed through application of the HAZOP study on the experimental design. The impact of the risks that were specifically addressed by HAZOP are illustrated through uncontrolled variables onto the internal and external validity of measurement studies. First, it is shown that the URL source can greatly impact measurements; second, it is shown that time can also have a major impact on measurement.

In Chapter 6, several new detection techniques that assess whether a web page belongs to a malware distribution network are developed and evaluated. These methods can be incorporated into lightweight low-interaction client honeypots, which are generally faster than high-interaction client honeypots at finding malicious web pages on a network. However, at the same time, they produce false positives and therefore would not be suitable as stand-alone systems to detect malicious web pages.

In Chapter 7, a hybrid client honeypot system is presented and evaluated. The hybrid client honeypot system combines the low- and high-interaction client honeypots into a cost-effective system. First, a model is presented that illustrates the impact of the low- and high-interaction client honeypot components on the overall system. A hybrid implementation is used to validate the model. The TPCC is used to evaluate the hybrid client honeypot system against a system of high-interaction client honeypots with identical resources.

Chapter 8 summarizes the thesis and the contributions made by the

thesis and discusses future work. Most of the future work presented in this chapter touches on further improvement of detection accuracy, understanding the anti-forensic capabilities of malicious web servers, and the speed of client honeypots, as well as increasing the understanding of client-side attacks.

1.5 Publications

Part of the research discussed in this thesis has appeared in other publications. Several of these papers were co-authored with others, but their content was primarily the work of the author of this thesis. The following parts of the thesis are based on previously published work:

- Chapter 4 – We presented early work on the divide-and-conquer visitation algorithm in *Application of divide-and-conquer algorithm paradigm to improve the detection speed of high-interaction client honeypots* at the 23rd Annual ACM Symposium on Applied Computing, 2008.

  The true positive cost curve as a means of evaluating and tuning high-interaction client honeypots in an operating environment was presented in *True Positive Cost Curve: A Cost-Based Evaluation Method for High-Interaction Client Honeypots* at the Third International Conference on Emerging Security Information, Systems and Technologies, SECURWARE, 2009.

  The reduction of malicious web pages identified as a result of choosing a visitation algorithm that requires repeated interaction with a web server to identify malicious web pages was addressed through utilization of a proxy. The proxy generically implements a record/replay mechanism, which was presented and discussed in *Justifying the Need for Forensically Ready Protocols: A Case Study of Identifying Malicious Web Servers Using Client Honeypots* at the 4th Annual IFIP WG 11.9 International Conference on Digital Forensics, 2008. The work was

published as a book chapter "Identifying and Analyzing Web Server Attacks" in Advances in Digital Forensics IV.

- Chapter 5 – A HAZOP analysis was conducted to identify hazards that threaten internal and external validity of measurement studies. Several hazards were identified that revolve around the apparatus: the client honeypot. We choose to make our client honeypot Capture-HPC open source and publicly available, so functional testing and code inspection by the open-source community would reduce functional bugs.

  Further, the hazard around the Analysis Engine's capability to identify unauthorized state changes was addressed with a "real-time" kernel-level state monitoring mechanism, which was presented in *Capture - A Behavioral Analysis Tool for Applications and Documents* at the 7th Digital Forensics Research Workshop Conference, 2007.

  The measurement studies were presented in a peer-reviewed white paper of the Know Your Enemy series, *KYE: Malicious Web Servers* published by the Honeynet Project and in *Measurement Study on Malicious Web Servers in the .nz Domain* at the 14th Australasian Conference on Information Security and Privacy (ACISP), 2009.

- Chapter 6 – The concept of low-interaction client honeypots was first identified in our Technical Report *Taxonomy of Honeypots*, 2006.

  Early works on a signature-based detection approach incorporated in a low-interaction client honeypot were presented in *HoneyC - The Low-Interaction Client Honeypot* at NZCSRCS, 2007.

  The detection mechanism that is based on dynamic behavior when loading a web page was presented and evaluated in *Identification of Malicious Web Pages Through Analysis of Underlying DNS and Web Server Relationships* at the 3rd IEEE Conference on Local Computer Networks, 2008.

The detection mechanism that is based on analyzing static characteristics on the web page was presented and evaluated in *Identification of Malicious Web Pages with Static Heuristics* at the Australasian Telecommunication Networks and Applications Conference, 2008.

- Chapter 7 – An abbreviated model and brief evaluation of the static detection mechanism in a hybrid client honeypot system was presented in *Identification of Malicious Web Pages with Static Heuristics* at the Australasian Telecommunication Networks and Applications Conference, 2008.

Chapter 2

Background

This thesis is concerned with detection of web-based client-side attacks launched by malicious web servers. This chapter presents the concept of intrusion detection, and provides an overview of various types of web-based client-side attacks, particularly drive-by-download attacks. The chapter closes with a definition of the scope of the thesis.

This work adopts the terminology on intrusion detection from the MAFTIA project [20], which is summarized in the glossary in Appendix A. Additional terminology around web-based client-side attacks and drive-by-downloads is also defined in the glossary.

2.1 Computer Attacks and Intrusion Detection

An intrusion detection system is a piece of software and/or hardware designed to detect and alert of attacks that occur on a computer system it is monitoring. Figure 2.1 shows such a computer system. As activity occurs on the computer system, the intrusion detection system monitors the activity. As regular events occur, the computer system performs a service and its intended function. If an event is processed by an error that exists within the system, the system will not or will only partially perform a service or its intended function and a failure occurs. As the intrusion de-

tection system monitors these regular events, it can either raise or elect not to raise an alert. If an alert is raised, a false positive is generated, because an attack has not taken place. If the intrusion detection system elects not to raise an alert, it correctly ignored the event and therefore a true negative is generated.

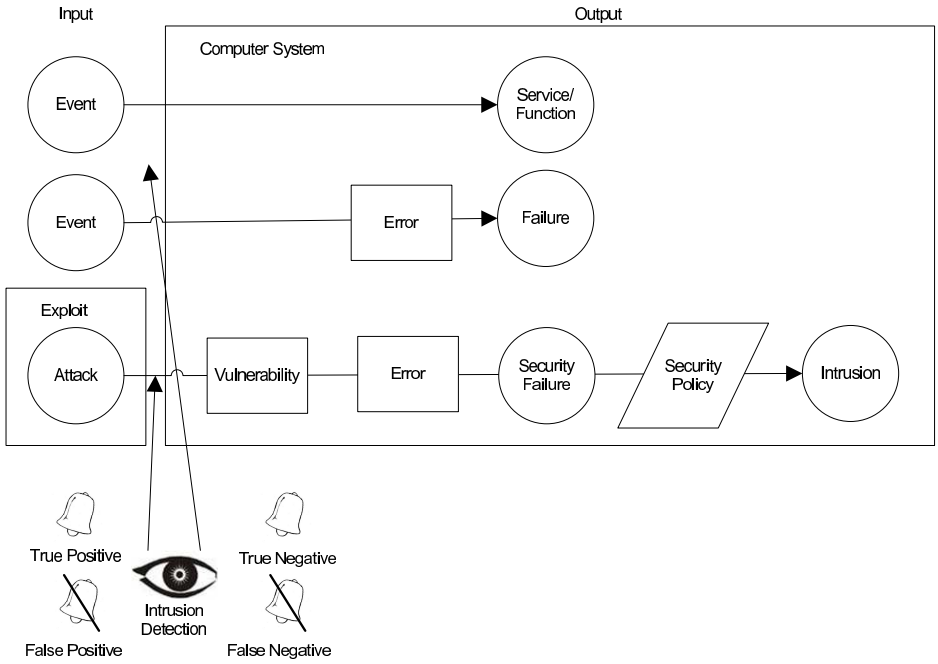


Figure 2.1: Computer Attack and Intrusion Detection Diagram

If the event is malicious, an attack occurs. These attacks are often wrapped into code that performs the attack, the so-called exploit. The attack attempts to generate a security failure by trying to exploit an existing vulnerability within the system. If the attack is successful, a security policy, which is enforced by the system, can be violated and an intrusion occurs. The intrusion detection system monitors the system for attacks. If one is detected, independent of whether the attack is successful, an alert, in this case a true positive, is generated. If the intrusion detection system fails to detect an attack and subsequently does not raise an alert, it generated a false negative. Note that the intrusion detection system does not

need to assess whether or not the attack has been successful. If an attack is observed, an alert should be generated. As such, *intrusion* detection system is a misleading term. A more accurate description would be *attack* detection system.

The security policy that is potentially violated by an attack attempts to enforce the confidentiality, availability, and integrity of the computer system and its data. If the attack is successful, these aspects will be impacted. An impact on availability will be a full or partial denial-of-service; impact on confidentiality will be information disclosure; and impact on integrity will be an alteration of the system, often through the execution of malicious code. These impacts are directly linked to the vulnerability as described by the Common Vulnerability Scoring System [68], which permits one or more of these impacts to be assigned to each vulnerability. As such, a vulnerability and also attack that targets a specific vulnerability are referred to as denial-of-service, information disclosure, and execution vulnerabilities and attacks.

2.2 Web-Based Client-Side Attacks

A description of how attacks can occur is given above. An adversary launches an attack on a computer system that exposes vulnerabilities. While one might get the impression that an interaction that leads to an attack is initiated by an adversary, this is not always correct, as Figure 2.2 illustrates. In a networked environment, an adversary might initiate an interaction by attacking vulnerable services exposed by a server as shown on the left side of the figure. However, an interaction might be initiated by the victim as well. In this scenario, a client might request a service from a malicious server, which returns an attack as part of the server's response that targets a vulnerability of the client, as shown on the right side of the figure. Those cases are referred to as client-side attacks.

Web-based client-side attacks are a particular type of client-side at-

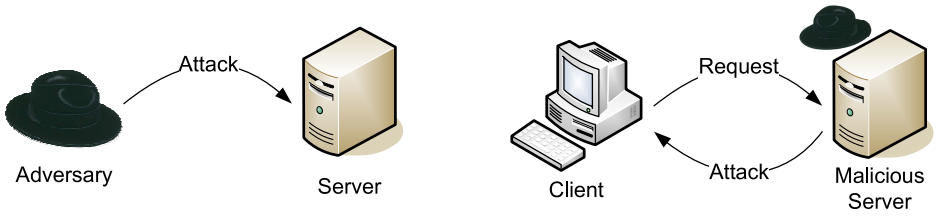


Figure 2.2: Attack on Server vs Attack on Client

tack that occurs on the World Wide Web. These client-side attacks are launched by malicious web servers that attack the user, operating system, web browsers and/or one of its plug-ins. During a web-based client-side attack, the client requests content from a malicious web server and the returned content contains an exploit that executes an attack. That content is usually, but not limited to, a web page. Figure 2.3 lists specific items that can be returned by a web server. All these items may contain an exploit.

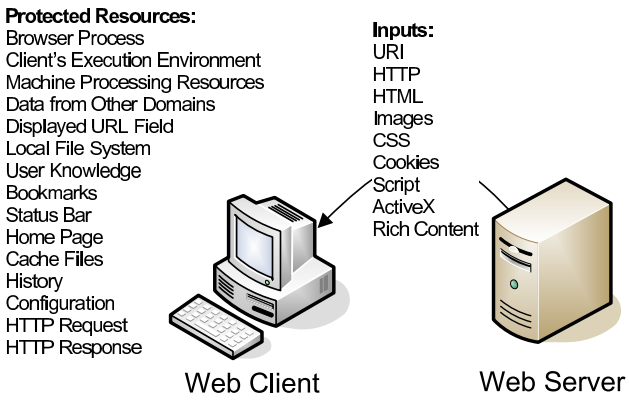


Figure 2.3: Protected Resources

Several types of web-based client-side attacks exist that target one of the protected resources shown in Figure 2.3. In this section, we categorize the types of web-based client-side attacks around the impact they have on these resources. This categorization was first published in Help Net Security.

2.2.1 Confidentiality Impact

Attacks described in this section are all concerned with accessing some confidential information on the client-side. Cookie-, history-, file-, and clipboard-stealing attacks are described, as well as attacks that are able to obtain information about protected internal network topology and user data (aka phishing attacks.)

A cookie is a piece of data that is sent by the server to be stored on the client for retrieval at a later time [73]. Cookies are primarily used to allow the web server to track the client across multiple request/response cycles. Cookies, according to the same origin security policy [101], can only be retrieved by the web server that sets them. As a result, web servers are not able to read cookies from other domains. Cookies themselves are not likely to represent an attack vector on the web client. However, they are a high-value target for attackers, as a cookie, with its purpose of identifying the client, would help with attempts to hijack a session and impersonate a client [129]. Web mail clients, for instance, utilize cookies to identify a user at a later time, so users do not have to provide their credentials each time they would like to access their mail. If an attacker can access the cookie, unauthorized access to the mail account could be obtained, as demonstrated recently [104, 53].

The browser history and the browser cache are other confidential pieces of information attackers can gain access to. As a user visits web pages, the browser records these web pages in its cache and browser history. If an attacker can gain access to the cache or browser history, information such as the user's email service or bank can be inferred and used in subsequent attacks, such as phishing and cookie-stealing attacks. Cache and browser history can be obtained via browser vulnerabilities, JavaScript, cross-side scripting (CSS), inspection of visited link color, and timing attacks [65, 23, 34, 55].

While cookie, cache, and browser history stealing concentrates on assets that are managed by the browser, web-based client-side attacks can

reach beyond the scope of the browser into the underlying operating system. Attacks that allow a web server to access arbitrary files are examples, such as a recently described technique to exploit Microsoft's Internet Explorer 7 "Header Forwards" [120]. The clipboard is another source that should be protected. While early versions of web browsers, such as Microsoft's Internet Explorer, allowed a web page to access the clipboard [117], access to the clipboard has since been restricted to allow access only if specifically granted. Exploit code that seems to get around this restriction has been observed in the wild [14]. Internal network topology is another asset that should be protected, but can be accessed. Special JavaScript network and port scanners exist that allow a malicious web site to obtain information about the internal network topology, such as existence of web servers, routers, and hosts [56, 105, 132].

The last attack presented here that impacts confidentiality is a social engineering attack called phishing. Social engineering attacks aim at exploiting the natural human tendency to trust [54]. In a phishing attack, trust in a web site is abused to fraudulently acquire personal confidential data, such as credentials and bank account information [161]. These web-based client-side attacks present the user with a fraudulent web site, often promoted via spam email that appears to be from a trusted entity, such as a bank. The web site, however, is in fact in the control of the attacker and once the user provides personal information to the web site, the attacker will have obtained this confidential information.

2.2.2 Availability Impact

Attacks that impact availability are concerned with partially or fully consuming the client resources, which reduces or leads to a complete failure of a service the client normally performs. The attacks reviewed are simple crashes, pop-up floods, browser hijacking, network floods, web spam/junk pages, and web pages that commit click fraud.

A denial-of-service is an attack that results in partial or complete consumption of resources that negatively impacts a service [92]. In the setting of a web-based client-side attack, a web page could cause the lock-up or crash of the browser or even the operating system or one of its components. Many browser vulnerabilities exist that permit a malicious web server to launch an availability-impacting attack [124].

While the lock-ups and crashes often occur without malicious intent, there are several availability-impacting attacks for which malicious intent undoubtedly exists. Pop-up floods are used in advertisement attacks [17]. These attacks lead to the display of many unsolicited pop-up windows. While these pop-ups load, network and computing resources are consumed, significantly reducing the availability of the client. This could even lead to browser hijacking, in which the page cannot be left and/or the pop-up cannot be closed [129].

Since web browsers are capable of loading resources (for instance, images) from remote network locations, a malicious web page could conceptually lead to flooding of the network with traffic if a browser does not manage its resources carefully. For instance, a web page that contains a million images from different domains could generate a million domain name service (DNS) requests, potentially overwhelming the local DNS server. A web page that contains large data chunks could potentially clog the network. If browsers are pooled to perform flooding of a network, they are referred to as Puppetnets [74].

Web spam/junk pages are specific malicious web pages that abuse search engine functionality. A search engine is tasked with providing the user with relevant web pages for given user queries. Web spam/junk pages abuse the algorithm of the search engine to lead to a high ranking despite the fact that the content of the web pages is not relevant to the user [59]. As such, these pages abuse the client's resources by displaying non-relevant content. On top of that, these and other pages might be involved in click fraud scams, in which a malicious web page could fraudulently

simulate clicking of advertisements by the user [64].

2.2.3 Integrity Impact

In the context of web-based client-side attacks, attacks that impact integrity usually translate into the ability of an attacker to execute arbitrary code on the client machine. In this section, cross site/domain/zone scripting, drive-by-pharming, hosting of malware, and drive-by-download attacks are described.

Cross site/domain/zone scripting is a vulnerability of web pages that allows execution of injected code in the security context of that page when the user visits such a page [113, 112]. The injected code could be used to steal information, but also could permit execution of arbitrary code on the client if, for instance, the web page is a trusted page in the context of the web browser.

Drive-by-pharming is a web-based client-side attack that modifies the DNS settings of a user's router by merely having a user visit a malicious web page [64]. These attacks do not impact the integrity of the client machine directly, but rather impact the integrity of network components the client relies on.

Hosting of malware is another type of attack that impacts the integrity of the client. In this attack scenario, the malicious web page hosts malware and uses social engineering to entice the user to download and execute the malware. An example of such a technique is a video codec that contains malware, which is presented to be a requirement to view pornographic material [24]. Once the user downloads and executes the malware, the malware has complete control of the machine.

Attacks that do not require user interaction, but rather are capable of pushing and executing malware without a user's notice or consent, are drive-by-download attacks. These attacks usually trigger when a user merely visits a web page [149]. They are the focus of this work and de-

scribed in more detail at the end of this chapter.

Impact Summary

The previous section described the impacts an attack may have in the context of web-based client-side attacks on web browsers. An attack may impact confidentiality, availability, and/or integrity. These attacks do not pose an equal amount of risk to a system. An inherent risk hierarchy exists in which attacks that impact the integrity of a system pose a greater risk to the system than do attacks that impact availability and confidentiality, because once the integrity is compromised, availability and confidentiality may be compromised as well.

The impact an attack may have directly maps to specific vulnerabilities of the web browser. According to MITRE's Common Vulnerabilities and Exposures (CVE) list, 232 vulnerabilities were publicly disclosed that allow remote attacks on a standard installation of Internet Explorer 6.0 on Microsoft Windows XP when our work commenced. Figure 2.4 shows the publicly disclosed vulnerabilities per year. Despite the code maturity of the application, the number of publicly disclosed vulnerabilities is not decreasing.

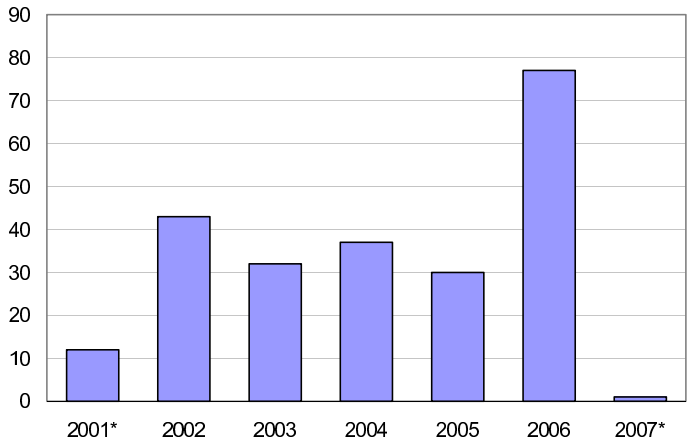


Figure 2.4: Publicly Disclosed IE6 Vulnerabilities per Year - * partial data

Each vulnerability maps to a specific impact it may have on the system in case of a successful attack. As such, a vulnerability may be referred to as denial-of-service, information disclosure, and execution vulnerabilities. Figure 2.5 shows the number of publicly disclosed Internet Explorer 6.0 vulnerabilities per possible impact. The ability of an attacker to execute arbitrary code that would impact the integrity of the system leads the list with 95 vulnerabilities, followed by the 56 vulnerabilities that would impact availability. Thirty-four vulnerabilities would impact confidentiality. It becomes clear from reviewing the graph and description of impacts that integrity is the impact that exhibits the most risk to a web browser and, as a result, is most targeted by attackers. Because of this, we focus on the detection of attacks that impact integrity as part of this work: drive-by-download attacks.

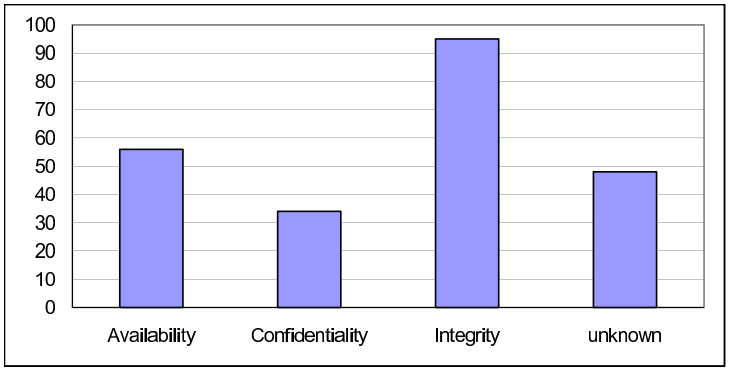


Figure 2.5: Publicly Disclosed IE6 Vulnerabilities per Possible Impact

2.3 Drive-by-Download Attacks

Detection of drive-by-download attacks is the focus of this work. In this section, drive-by-download attacks are described in more detail. As mentioned above, drive-by-download attacks are a specific type of web-based client-side attacks. Figures 2.6 and 2.7 show the steps of a typical drive-by-download attack. First, a web browser requests web pages from a re-

mote web server. As a response, the server returns a web page to the web browser that contains attack code that exploits a web browser's remote code execution vulnerability (Step 1). If the malware is not delivered as part of the attack code's payload, a special payload called a downloader can optionally first pull and then execute malware on the local workstation (Step 2). The entire attack happens without the user's consent or notice.

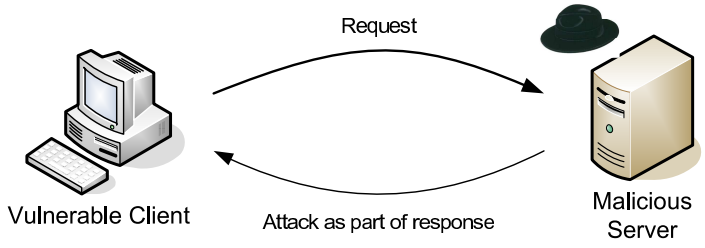


Figure 2.6: Drive-by-Download Attack - Step 1

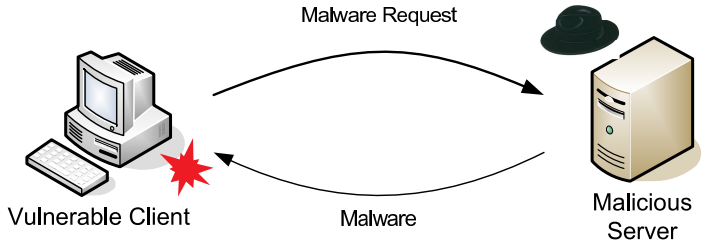


Figure 2.7: Drive-by-Download Attack - Step 2

Drive-by-download attacks target vulnerabilities on the client-side by sending exploits as part of a web server response. The targeted vulnerabilities can naturally reside within the web browser itself. However, through shared libraries and plug-ins, the attack surface expands to the operating system as well as other applications residing on the client. Attacks launched on Microsoft's Internet Explorer can target a particular type of plug-in: ActiveX components.

ActiveX components could be abused in several ways. An ActiveX component may contain a vulnerability that can be exploited by specially

crafted interaction with such a vulnerable ActiveX component. Alternatively, an ActiveX component may simply expose unsafe application programming interface (API) that allows an attacker to push and execute malware onto the client machine by merely making use of the ActiveX component's exposed functionality. The DownloadAndInstall API of the Sina ActiveX component is an example of such a component [148].

A cursory inspection of several vulnerability databases reveals that all operating systems and web browsers contain remote execution vulnerabilities that permit drive-by-download attacks to take place [147, 124, 72]. An analysis of the SecurityFocus vulnerability database [124] reveals that, as of August 2008, MSIE6SP2 and MFF15, two comparable browsers that shared a majority of the market during the same period, contain 44 and 66 remote execution vulnerabilities, respectively. However, if the dominance of MSIE6SP2 in the market is taken into consideration [43, 155], the absolute numbers of vulnerabilities are higher for MSIE6SP2 than for MFF15. An attacker, therefore, gains more return on investment if specializing on MSIE6SP2 attacks.

Based on this information, the scope of this work is defined.

2.4 Scope

This work focuses on the detection of drive-by-download attacks. Because of the prevalence of drive-by-download attacks on Internet Explorer 6.0 SP2, this work focuses on detection of drive-by-download attacks that target this web browser [88]. While major upgrades to the web browser were released and Internet Explorer 8.0 is its latest version, web servers still attack Internet Explorer 6.0. This can be attributed to the fact that many users have not upgraded [43] and a continued return on investment exists for attackers to target the older version of Internet Explorer 6.0.

While it appears that third-party plug-ins are increasingly targeted in these types of attacks, we have chosen to utilize only attacks that target

a stock installation of Internet Explorer. This will keep the experimental setup simple and we assume that this configuration solicits enough attacks to allow us to study drive-by-download attacks and investigate new detection techniques. It is expected that our work will be generalizable beyond the particular configuration of the browser.

In this chapter, we have presented the concept of intrusion detection, provided an overview of various types of web-based client-side attacks, particularly the drive-by-download attacks, as well as defined the scope of the thesis. Our work focuses on the detection of drive-by-download attacks. In the next chapter, related work in this area is reviewed. We review related work on intrusion detection as well as specific work that aims at detection of and protection against drive-by-download attacks. We show the gaps and short comings of the existing work around detection and illustrate how our work intends to more effectively detect drive-by-download attacks.

Chapter 3

Related Work

In this chapter, the related work is reviewed and discussed. The problems of drive-by-download attacks and malicious web servers, as well as characteristics of malicious web servers, are reviewed based on existing white papers and research studies. The second section reviews defensive intrusion prevention strategies and shows that generic strategies are ineffective to counter the threat of drive-by-download attacks. While our work focuses on the task of identification of malicious web servers, research on specific defensive strategies exists and is presented in Section 3.2. Detection of drive-by-download attacks, however, is challenging and a major contribution of this work. Intrusion detection techniques are ineffective to detect drive-by-download attacks and a more able detection mechanism, i.e., client honeypots, is needed. Section 3.3 reviews existing detection techniques, in particular client honeypot technology, and demonstrates where these lack capabilities. This chapter concludes with a discussion of gaps from the related work that are addressed by our work.

3.1 Drive-by-Download Attacks

Several commercial entities work in the space of intrusion detection. Since commencement of the work for this PhD, many commercial entities have

appeared that focus on the detection of and protection against drive-by-download attacks [163, 162, 130, 60, 143, 1, 86, 36, 47, 122, 82]. These entities actively search for malicious web sites and just recently have started to publish white papers with their findings on a regular basis [123, 131, 83, 164, 38, 90]. The latest publications available in the second half of 2008 paint a dire picture of the Internet landscape. Financial gain is the primary motive behind drive-by-download attacks and has attracted a lot of criminal elements. Sophisticated criminal structures have been established to more effectively steal and trade confidential information, such as credit card numbers or online game account credentials, collected from a victim's machine and disseminated [164, 123, 38]. Increasing the chance of exposure that potential victims will be exposed to malicious web sites increases the criminal organization's profits. This is a consistent theme throughout the various white papers.

The majority of the white papers report a significant increase of web-based client-side attacks in the middle of 2008, leading to the web becoming the primary attack vector to infect users with malware [123, 131]. To increase exposure to these attacks, attackers are increasingly turning to hacking and abusing existing legitimate web sites that have an established incoming traffic stream. Attackers break into these systems by targeting the web application [123, 164, 131], with structured query language (SQL) injection being a primary attack vector [123], but other types of injection attacks have been observed [164]. Tools such as BSQL Hacker [79] simplify these attacks and even allow a novice adversary to launch attacks on a large scale. Thousands of web sites infected by SQL injection attacks have been observed [121, 76, 91]. No type of web site seems to be immune to these attacks and even the majority of the top 100 most popular web sites seem to have been involved in malicious activity in the first half of 2008 [164]. Even knowledgeable and security-conscious organizations, such as the US consulate in St. Petersburg, have fallen victim [131]. Sophos reports 83% and Websense reports 75% of malicious web sites are legitimate

sites that have been hacked.

To increase the effectiveness of the malicious web sites, criminals are highly sensitive to location, language, and economic trends [38]. Specific regions, for example, are targeted in so-called campaigns. Web exploitation kits that bundle various exploits support these specialized campaigns. As a result, attacks might differ from country to country [38]. For instance, the majority of malware on Chinese sites might target the theft of passwords from online gamers [90, 131], whereas malware on Brazilian sites is designed to steal bank account information [131]. Through specific campaigns, attackers are seeking to increase their return on investment through mass penetration of specific weaknesses.

As a result, the security companies in this space pay special attention to geography. Data from the majority of reports lists two countries as hosting the majority of web-based client-side attacks: the United States of America and the People's Republic of China [123, 141, 131]. Stopbadware.org found a higher percentage of malicious sites in China and hypothesizes that lack of economic incentives for Chinese hosting providers to clean their sites is a reason for this difference. Contact initiated by Stopbadware.org with US and European hosting providers was fruitful in removing malicious content, whereas in China these efforts were not. McAfee didn't investigate physical location, but rather investigated top level domain names [83]. Their report shows that web sites in .ro (Romania), .info, and .nu (Niue) contain the highest percentage of malicious web sites; .cn (China) is listed as fourth; .us (United States) is in 18th position.

China is named repeatedly in the white papers. Malicious web sites and the underground economy of the Chinese web were the focus of an academic study by Zhuge et al. [174]. Measurements on 215,511 popular Chinese web pages with a high-interaction client honeypot revealed a high percentage of malicious web pages at 1.38%. Provos et al. also observed a high fraction of malicious web pages to be located in China [110]. Similar measurements on the prevalence of malicious web pages by previous

studies on more generic web samples showed lower values of 0.2% [96] and 0.071% [159]. Direct comparison of these values, however, cannot be conducted due to a lack of information provided by the studies and lack of information on how malicious web pages behave, which is discussed at the end of this chapter.

Several studies have investigated whether URLs from various sources influence the potential exposure to drive-by-download attacks. It comes as no surprise that questionable content, such as warez, pornography, links from known bad sites, or spam messages, shows a significantly higher risk of exposure [159, 110, 96]. However, these studies also observe risk of exposure to drive-by-download attacks on reputable web pages. A user avoiding questionable content can lower, but not eliminate, the risk.

Analysis of the components a browser retrieves when loading a page permits the analyst to pinpoint the source of the actual attack code. Provos et al. estimate that 2% of all pages that launched drive-by-download attacks were delivering attacks via advertisements [110]. Considering that reputable web sites with a wide reach could therefore launch attacks, this represents a particularly dangerous situation for end users. But even pages that do not host advertisements ”import” exploits from other hosts. Wang et al. first analyzed these structures in 2005 [159]. A browser that loads a malicious web page can be redirected via multiple pages on numerous hosts until the actual exploit is delivered by a central exploit server. Provos et al. observed that 82% of malicious web pages identified make use of such a structure. These exploit servers might be contacted by numerous malicious web pages. Some exploit servers are used by ”well over 21,000” malicious web pages [110]. These networks of malicious web pages, redirect sites, and exploit servers are also referred to as malware distribution networks. Characterizing whether a web page belongs to such a network is a main contribution of our work.

3.2 Intrusion Prevention

Intrusion prevention is one technique that could be used to defend against drive-by-download attacks. Widely adopted defensive measures, such as antivirus software, network address translation (NAT), and packet filters, have significantly contributed to increased security of computer systems. However, they are ineffective against drive-by-download attacks.

Antivirus software first appeared in 1988 [167]. The early versions were highly focused on detection of particular viruses. Shortly after, first generation scanners appeared that were able to identify viruses based on simple string matching [144]. Antivirus software initially was tasked with identifying viruses and disinfecting the infected files. The scanning techniques, as a result, were highly specialized to concentrate on binary data within executable files. Malware is continuously making an effort to evade detection by antivirus software. Polymorphism, a technique in which the binary fingerprint of the malware, but not its underlying functionality, changes, is widely adopted today. Antivirus software first needs to update its signature to enable detection of the ”morphed” malware.

Antivirus software is constantly being evaluated and compared [12, 6]. These tests show that detection rates can be as low as 63%. While the average is higher and better antivirus products detect the majority of malware, no antivirus product is able to detect all malware. In the area of drive-by-download attacks, the performance of antivirus products is even worse. Modification of the entry point, insertion of junk instructions, usage of binary packers, obfuscated packer, and modification based on existing antivirus signatures are polymorphic techniques observed by Zhuge et al. on malware pushed by drive-by-download attacks [174]. Malicious web pages are in a good position to adjust malware frequently to evade detection by antivirus software, because the malware is hosted on a central machine that is controlled by the attacker. Provos et al. observed that a small percentage of URLs change the malware as often as every hour

[111]. The detection rates of antivirus software are therefore rather low, about 70% on average [110]. This is the antivirus software detection rate for identifying a malicious binary that is placed as a result of a successful attack. If the attack doesn't succeed or its payload doesn't push a binary, for instance, in case a user account is added or sensitive documents are copied, the antivirus would fail to detect such attacks. As such, the true positive rate of antivirus software on detection of web pages that launch drive-by-download attacks is likely to be lower.

A packet filter, which is a widely adopted defensive tool [52] that restricts communication between networks based on network protocol characteristics, such as transmission control protocol (TCP) ports, usually does not prevent a drive-by-download attack from occurring either. Because a packet filter can either permit traffic or block traffic at the expense of the service, if system administrator permits browsers to access web pages, exposure to drive-by-download attacks exists and the packet filter cannot provide a layer of protection. Similarly NAT, which protects the intranet infrastructure only from access that is initiated from an external entity, does not provide a layer of protection either, because in drive-by-download attacks, access is initiated from within the intranet, permitting external entities to send content, in this case web pages from a web server, to the machines on the intranet. The drive-by-download attack can thus occur through the NAT gateway.

Numerous research efforts are under way to directly protect the client application. BrowserShield, for instance, defuses malicious JavaScript at run-time by rewriting web pages and any embedded scripts into safe equivalents [114]. Self-defending software is being explored by Michael Ernst [30]. This research protects commercial off-the-shelve (COTS) software by detecting attacks in a collaborative environment and automatically applying generated patches to prevent such attacks in the future. Application of this method on the Firefox browser serves as a proof-of-concept. Anagnostakis et al. use anomaly detection and shadow honeypots to protect,

among others, client applications, such as a web browser [5]. Before the web browser is permitted to process the requested data, suspicious data is forwarded to the shadow honeypot, an instrumented browser that detects memory violation attacks. If no attack is detected, the data is forwarded to the end user; if an attack is detected, the data is dropped and the end user effectively protected. A similar approach that uses execution-based web content analysis in disposable virtual machines is presented by Moshchuk et al. [95].

Commercial entities concentrate on blacklisting as a defensive strategy. In those approaches, a request to retrieve a web page is checked against a database of known bad web pages before the request is granted. The latest versions of popular web browsers have adopted a blacklisting approach [97, 102, 51]. Browsers that lack blacklisting capabilities can be enhanced with such functionality through a variety of browser plug-ins or network-based blocking devices [82, 60, 163]. Because a majority of web pages are accessed via search engines [18], the major search engines Google, Bing, and Yahoo provide a layer of protection by blacklisting malicious web pages on their results page [126, 172, 48]. Google went as far as releasing a publicly accessible API to check URLs against Google's blacklist [50]. Blacklisting, however, can provide protection only if the underlying detection mechanism that the blacklists are based on is effective.

Existing work on the detection of malicious web pages is reviewed next.

3.3 Detection

While the research efforts on directly protecting client applications are promising, the need to effectively detect drive-by-download attacks remains. Detection is important for incident response, economic modeling, trend analysis, identification of new attack techniques, etc. Intrusion detection systems, in particular network-based misuse intrusion detection

systems, employ a similar approach to antivirus technology to detect attacks [11]. Snort, a widely adopted open-source intrusion detection system [118], employs pattern-matching technology that could potentially also detect malicious web pages as it passes through the network. However, the attack has to be known by these systems to be detected. In addition, even known attacks can be missed. Obfuscation, which has been observed by many studies mentioned above [159, 111, 174], can make detection difficult. Obfuscation is a technique in which malicious web servers change the representation of the attack code during transmission so the attack code cannot be identified. Upon loading of the web page, the obfuscated attack code is converted into its clear text and is executed. As such, intrusion detection systems are mainly ineffective to detect drive-by-download attacks. Evaluation of detection with intrusion detection signatures is a contribution of this work.

Honeypots, an alternative to intrusion detection systems, are security devices that are designed to detect computer intrusions and attacks. They are dedicated security devices whose value lies in being probed, attacked, and compromised [133]. A honeypot, for instance, could be a vulnerable web server that is not contacted by legitimate users. Attackers that scan for vulnerable web servers will eventually find this web server and attack it. The operator of the honeypot can observe and study the attack.

The origin of honeypots can be traced far back to military concepts and usage, but they first appeared in the area of computer security in the 1980s [139]. They address some of the shortcomings that intrusion detection systems pose. In particular, they are capable of detecting unknown attacks at low false positive rates.

To use honeypots to detect drive-by-download attacks, a new type of honeypot was introduced: the client honeypot. The concept was first articulated in 2004 [134] and studies and specific implementations first appeared in 2005 and 2006 [157, 159, 96]. A client honeypot is a honeypot that finds servers that attack clients. It actively interacts with potentially mali-

cious servers to determine whether they are malicious or benign. Mostly, client honeypots today identify drive-by-download attacks on browsers launched by a web server. However, client honeypots can identify a wider spectrum of client-side attacks and also are capable of detecting attacks on client applications other than web browsers.

Next, common client honeypot components are described, followed by a description of the two major types of client honeypots: low- and high-interaction client honeypots. For each type, specific implementations are reviewed and advantages and disadvantages of the various types are summarized.

3.3.1 Client Honeypot Components

A client honeypot actively interacts with potentially malicious servers to determine whether they are malicious or benign. Based on our client honeypot implementation and analysis of existing client honeypots, we identified three core components of a client honeypot: Queuer, Visitor, Analysis Engine. This model has been accepted as a client honeypot model by the research community [116, 169, 33, 142]. The client, such as a web browser, is controlled via a Visitor component of the client honeypot, which interacts with potentially malicious web servers. Information about what server to interact with and the data to be sent to the server is created by a Queuer component that generates server requests. A Queuer component could be, for example, a web crawler. Lastly, the Analysis Engine assesses whether the server the Visitor interacted with is malicious or benign.

3.3.2 Client Honeypots Types

One can classify honeypots by the interaction level. Possible values of the interaction level are high and low. The high-interaction level denotes that the honeypot system allows for full functional interaction, whereas a low-interaction level signifies that the functionality is limited, for example,

by using emulated services [109]. In the context of client honeypots, the Visitor component determines the interaction level.

High-Interaction Client Honeypots

As mentioned above, a high-interaction client honeypot can fully interact with the server. An actual instance of a vulnerable browser on a dedicated operating system is a natural candidate for a high-interaction client honeypot. As the client interacts with the server, the system, via the Analysis Engine component, monitors the system for unauthorized state changes, such as file modifications or process adjustments that would indicate a successful attack. This approach is not limited to web browsers, and recent research has explored additional client applications, such as instant messaging applications [171] and office applications [116].

Whether a state change is authorized or unauthorized is determined by an implied security policy. A browser process, for instance, is expected to create new files in the browser cache, but is not permitted to create new files in the startup group. A browser process is expected to open the default email client when processing a mailto: link, but is not permitted to execute the command shell. A user might grant exception of the security policy as the user browses the web. For instance, a user can choose to download and execute a program when prompted to do so.

Figures 3.1 and 3.2 show the security policy of a web browser for file system events and process events. A web browser is permitted to read, create, and modify cache, history, cookies, and temporary files as well as files that are generated as part of a web browser crash. (While a crash could indicate an attack, it could also indicate a non-malicious fault. As such, crashes are ignored as part of our work.) Further, a web browser's plug-ins are permitted to read, create, and modify temporary files and plug-in application and user data, which are specific to the plug-in. Any file access, creation or modification of files are considered an unauthorized state change. Permitted process events are fewer. A web browser is

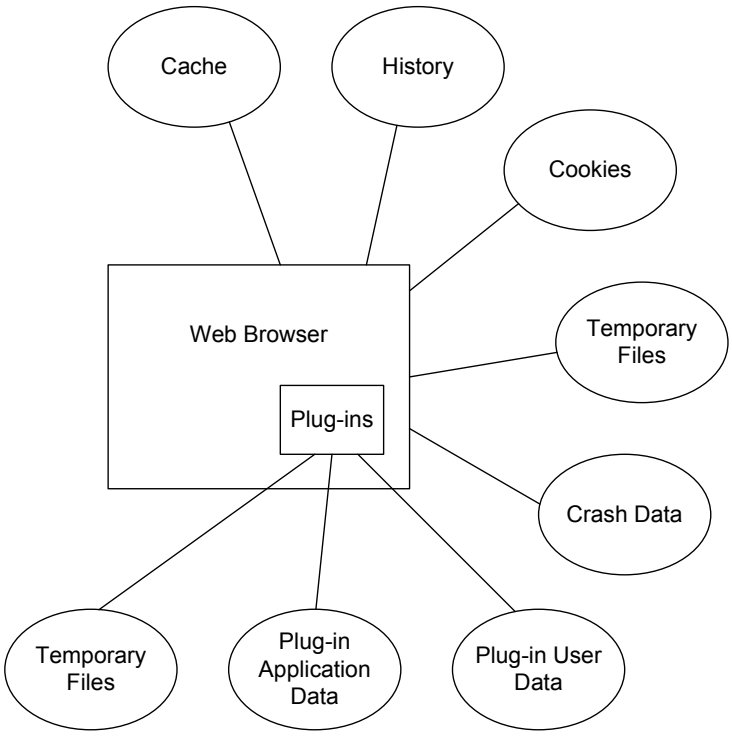


Figure 3.1: Authorized File State Changes of Web Browser and Its Plug-ins

permitted to create and terminate processes that are linked to its protocol handler (e.g., the default email client for mailto: protocol) and processes that are related to a web browser crash and printing. Further, the plug-ins are permitted to create and terminate plug-in-dependent processes (e.g., MSN messenger plug-in is permitted to open the MSN messenger client). Creation or termination of any other processes is considered an unauthorized state change.

Because the system that drives the vulnerable browser is actively exploited in an attack, it cannot be trusted anymore. As a result, it is reset into a clean state before the client honeypot proceeds to inspect additional URLs. All high-interaction client honeypots today use some form of virtualization technology, because it provides an easy way to reset the state of the operating system and client honeypot into a clean state.

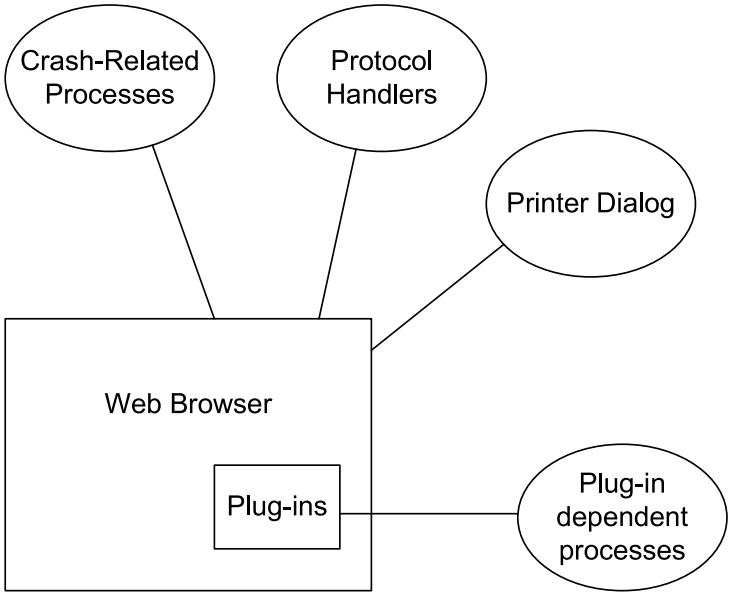


Figure 3.2: Authorized Process State Changes of Web Browser and Its Plug-ins

The main advantage of high-interaction client honeypots that monitor the system for unauthorized state changes is their ability to detect known as well as unknown attacks, because no knowledge about the attacks is applied when detecting them. Rather, the effects of successful attacks are observed. In addition, the false positive rate, given that the security policy is correct, is negligibly low. For each reported attack, it is guaranteed that an attack has taken place.

When our work commenced, only a few high-interaction client honeypots were mentioned in the literature: HoneyClient [157], HoneyMonkey [159], University of Washington (UW) client honeypot [96], and the Pezzonavante Honeyclient [25]. These client honeypots focus on malicious web servers, which they interact with by driving a web browser on the dedicated honeypot system. HoneyClient detects successful attacks by monitoring changes to a list of files, directories, and system configuration after the HoneyClient has interacted with a server. HoneyMonkey also

detects intrusions by monitoring changes to a list of executable files and registry entries, but HoneyMonkey goes a step further by adding monitoring of the child processes to its repertoire to detect client-side attacks. In addition, HoneyMonkey contains a *vulnerability specific exploit detector* that records an identifier on what vulnerability was targeted by a specific exploit; however, the study merely mentions the existence of such a detector, but does not provide any empirical data on the performance of the detector. The UW client honeypot uses event triggers of file system activity, process creation, registry activity, and browser crashes to identify client-side attacks.

There were some additional technical differences between the implementations. The initial version of HoneyClient took snapshots before and after the client application interacted with a set of servers in sequence. The snapshot mechanism was slow, but multiple servers between snapshots reduced this time to lower levels. Interacting with multiple servers between snapshots did not allow HoneyClient to determine which server launched the attack. A similar approach was adopted by HoneyMonkey. However, HoneyMonkey implements a pipeline in which, once an attack is identified, the URLs it requested between snapshots are forwarded to a system that interacts with each server in sequence. A system that monitors state changes in real time permitted this system to more quickly pinpoint the server that launched the attack.

All client honeypots interact with a potentially malicious server with a vulnerable client. Because an attack might not immediately trigger, the clients wait a period before a final classification. Honeymonkey and UW client honeypot waited approximately two minutes. Spycrawler sped up the clock of the client honeypot system, so they could reduce the visitation time further. Their system was able to inspect approximately 15,000 URLs per client honeypot system per day. This compares to 4,000 URLs for the HoneyMonkey system.

In addition, client honeypots have been observed to miss attacks [159,

96, 174]. This is particularly problematic when conducting studies on the prevalence of malicious web pages on the Internet. Time bombs, in which an exploit triggers only after an extensive period has passed; non-deterministic behavior of malicious web pages; attacks that trigger only on user interaction; and triggering an attack only when accessing a web server from a specific network are: all behavior that is suspected or has been observed by these studies. These all lead to the client honeypot failing to identify an attack causing a false negative result.

Client honeypots are faced with difficulties in comprehensively detecting malicious web pages. They are slow and, because the detection algorithms are based on monitoring unauthorized state changes in a dedicated system, they require a lot of computing resources. In addition, they tend to miss attacks. With the existence of millions of web servers and billions of web pages, client honeypots may be prohibitively expensive for inspecting a large portion of the web. Efficiency improvements to high-interaction client honeypots and the introduction of a hybrid client honeypot system that significantly reduces the costs of identifying malicious web pages are major contributions of our work.

Low-Interaction Client Honeypots

The low-interaction client honeypot is the second major type of client honeypot. We note the lack of low-interaction client honeypots for detection of drive-by-download attacks. Low-interaction client honeypots simulate clients and assess whether an attack has occurred. Because active exploitation may not occur, the low-interaction client honeypot does not classify a response by monitoring the system for unauthorized state changes, but rather by inspecting the response directly. Signatures, heuristics, and security predicates are possible techniques through which low-interaction client honeypots are able to detect attacks.

The advantages of low-interaction client honeypots are multifold. Because a client can be simulated, these types of client honeypots are usu-

ally more lightweight and able to scale better than high-interaction client honeypots. In addition, because they are not looking for the effects of an attack, an attack does not need to trigger for a low-interaction client honeypot to classify a web page as malicious. This allows these honeypots to detect certain types of attacks that high-interaction client honeypots might miss. For instance, if an attack triggers only on some action of the user, a low-interaction client honeypot can still inspect the response and detect the attack code.

Specific implementations of low-interaction client honeypots that detect drive-by-download attacks have appeared since we identified the lack of low-interaction client honeypots in our taxonomy. We review these implementations next. Some low-interaction client honeypots detect malicious web sites that contain threats other than drive-by-downloads. We refer the interested reader to the bibliography [63, 96, 108]. The first implementation of a low-interaction client honeypot designed to detect drive-by-downloads was our HoneyC.

Stuurman et al. analyzed a set of malicious and benign web page content using static techniques to determine whether differences exist that would permit classification of web pages using low-interaction client honeypots [142]. The researchers investigated obfuscated JavaScript, strings after deobfuscation, and the existence of iFrames. They observed that obfuscated JavaScript and certain strings are indicators of a malicious web page. They observed the existence of iFrames on both benign and malicious web pages and concluded that they were not suitable for classifying pages. However, a low-interaction client honeypot system was not built by the researchers and an evaluation of these methods as part of a low-interaction client honeypot was not discussed. Our work presented in Section 6.3 shows that an iFrame that has specific characteristics can be a good indicator of whether a page is malicious or benign.

The Caffeine Monkey engine is a tool that is targeted at collection, detection, and analysis of malicious JavaScript [33]. It uses a combination

of static and dynamic analysis techniques to classify JavaScript. Instead of looking at absolute numbers of JavaScript elements, ratios of function calls are taken into account. Obfuscation that might hinder such an analysis is addressed by automatically deobfuscating the JavaScript code with an instrumented JavaScript engine.

HoneySpider, a system built by NASK/CERT Polska, GOVCERT.NL and SURFnet, contains a low-interaction client honeypot that detects malicious web pages based on obfuscation [135]. It uses a machine learning algorithm built based on x character N-grams. While no published research is available, the researchers report some promising results [69].

Provos et al. used a machine learning algorithm based on a set of heuristics, such as the existence of iFrames, obfuscated JavaScript, etc., to classify a web page [110]. A cross validation predicted a detection accuracy of 0.001 false positive and 0.6 true positive rates. However, no details on the classification method were disclosed.

While these approaches try to inspect static characteristics, Nazario takes a different approach with the low-interaction client honeypot Phoneyc [99]. According to Nazario, Phoneyc was designed to specifically address the shortcomings of our HoneyC system. Phoneyc is capable of detecting attacks on scriptable ActiveX components. It uses a simulated browser with ActiveX interface to detect specific attacks through vulnerability-specific predicates [66], similar to the vulnerability-specific exploit detector presented by Wang et al. [159]. Phoneyc is therefore in the position to categorize attacks based on the vulnerabilities they are exploiting. Because it is based on specific predicates, it is not capable of detecting unknown attacks. While a description of the tool is available, at the time of this writing no quantitative research that evaluated the approach has been published. A similar approach to Nazario's work has been presented by Buescher et al. [19].

As already mentioned, low-interaction client honeypots have some disadvantages over high-interaction client honeypots. Approaches in which

predefined knowledge is required are not capable of detecting unknown attacks. Further, interacting with a server using a simulated client runs the risk of being detected and, as a result, no attack is launched and detection fails. In the case of a web browser, this could be done passively, such as evaluation of browser headers as demonstrated by Ruef [119], or actively by, for example, utilizing functionality usually not present in simulated clients, such as the ability to initialize ActiveX components, as recently demonstrated by Hoffman [61]. These capabilities have been observed in which the web page does not contain an attack if accessed with a simulated client, such as wget, but does contain the attack when accessed with a real client, such as MSIE6 [165]. The end result will be identical: the low-interaction client honeypot will not detect an attack.

Low-interaction client honeypots produce false positives. This is another disadvantage. In particular, low-interaction client honeypots that utilize machine learning methods produce false positives. An example is obfuscation. While obfuscation is usually encountered on malicious web pages, obfuscation also has its legitimate use. It is used by advertisement companies to protect themselves from click-fraud tools and can also be used to protect from intellectual property theft. A low-interaction client honeypot that alerts on obfuscated code may produce false positives in those instances.

Overall, however, low- and high-interaction client honeypots both have their advantages and disadvantages that on balance do not elevate one technology over the other. Rather, the purpose and goals of operating a client honeypot determine which technology should be used. Pouget et al. compared the interaction levels of server honeypots [106] and concluded they are complementary in nature and allow for more accuracy, depending on the circumstances of deployment and goals of data collection. For example, it might be unnecessary to deploy a high-interaction server honeypot on a global scale as global data is likely to be similar; low-interaction server honeypots are more suited for this situation. On

the other hand, low-interaction server honeypots are not suited for an in-depth investigation of attacker's actions once a server honeypot has been successfully compromised. High-interaction server honeypots are required to meet these goals, as they expose the full functional spectrum of a computer system to the attacker and therefore allow for collection of the desired data. A similar approach seems viable for client honeypots. A system that combines components to address the false positive rate of high-interaction honeypots was presented by Anagnostakis et al. [5]. Anomaly-based intrusion detection is used for an initial assessment of the data. Once suspicious data with a large false positive rate is identified, it is forwarded to a honeypot for a final assessment. Application of this approach to client honeypots was suggested by the researchers. Sidiroglou et al. present a hybrid system to protect against malicious email attachments [128]. Suspicious email attachments are opened in an instrumented virtual machine. If dangerous actions such as writing to the Windows Registry are detected, the mail is deemed malicious and is quarantined. As part of our work, we will present a hybrid system of low- and high-interaction client honeypots that is capable of finding malicious web sites on the network in a cost-effective way.

3.4 Gaps in the Related Work

Through an overview of the related work, this section identifies some specific gaps in the existing research that we are planning to fill with our research to detect malicious web pages in a cost-effective way. Four areas are the focus of our work: client honeypot technology, an evaluation method of client honeypot technology, a methodology that reduces risk to internal and external validity, and development of a method to assess whether a page belongs to a malware distribution network using low-interaction client honeypots and subsequently hybrid client honeypots in a more cost effective way.

First, we focus on client honeypot technology. A review of the related work shows that several different high-interaction client honeypot systems exist. However, at the time our work commenced, only MITRE's HoneyClient was publically available, whereas Moshchuk's implementation and Microsoft's HoneyMonkey were not publicly available. MITRE's HoneyClient did not fulfill our requirements around speed, stability, extensibility, and ability to collect digital evidence to conduct research in this area. An open research platform to study malicious web pages was missing. We addressed this gap through implementation of the open-source client honeypot Capture-HPC, which is publicly and freely available.

Second, we focus on evaluation techniques. From the description of the existing high-interaction client honeypots, it appears that they all use a similar approach to detect malicious web pages, but do contain some obvious differences, such as visitation algorithm, classification delay, vulnerability exposure, etc. However, the differences do not translate into a metric of increased ability to detect malicious web pages. In general, no model or method to evaluate client honeypots exists. As a result, comparison of various client honeypots is not possible and research in this area to further implement improvements is hindered.

In Chapter 4, we address this gap. First, we present a model that allows one to assess the effectiveness of high-interaction client honeypots and therefore provides a mechanism to objectively compare high-interaction client honeypots. The basis of this model is a cost model of operating client honeypots. Factors that feed into this model are speed, detection accuracy, resource costs, and the base rate – percentage of the malicious web servers – p_m that a client honeypot is presented with. We present our high-interaction client honeypot Capture-HPC with improvements on its visitation algorithm to illustrate how the method can be used to evaluate high-interaction client honeypots.

Third, we concentrate on the risks to internal and external validity of the experimental design that aims at identifying malicious web servers on

the network with client honeypots. Internal validity captures the intent of the researchers, whereas external validity is about the ability to generalize results beyond the experimental setting. The related work section presents various studies and white papers that report on measurement of malicious web servers on the network. However, the results differ considerably. For instance, Moshchuk's study presents percentage of malicious web pages of 1.5% in May 2005. During the same time frame, Wang observes a percentage of 0.071%. These numbers raise the question of whether the risks to internal and external validity were mitigated as part of the studies' experimental setup.

In Chapter 5, we apply the HAZOP study on the experimental design of the measurement study to identify, prioritize, and mitigate threats to internal and external validity in a systematic and thorough manner. We illustrate how the lack of control on dependent variables may pose a risk to internal and external validity. First, we illustrate that the URL source can greatly impact measurements; second, we show how time also has a major impact on measurement. HAZOP allows us to mitigate risks to internal and external validity of our studies in a systematic and thorough manner.

The fourth gap in efficiently detecting malicious web pages is filled through the low-interaction client honeypots which determine whether a web page belongs to a malware distribution network. Early studies on malicious web servers identified the structures of a malware distribution network. Provos et al. and Wang et al. showed that many malicious web pages utilize centralized exploit servers. However, very little work has been done to characterize these structures to the point that they can be used to characterize whether a web page belongs to such a network. We employ static and dynamic analysis techniques to determine whether a web page belongs to a malware distribution network.

Equipped with this knowledge, we model, develop, and evaluate a hybrid client honeypot system that is able to detect malicious web pages

much more cost-effectively than existing methods. In Chapter 7, we present this hybrid client honeypot system. It combines a low-interaction client honeypot that incorporates the method of assessing whether a page belongs to a malware distribution network with the traditional high-interaction client honeypot into a hybrid system. An evaluation shows that such a hybrid client honeypot system is capable of detecting malicious web pages much more cost effectively than existing approaches.

Chapter 4

True Positive Cost Curve

A client honeypot is a honeypot that finds servers that attack clients. It interacts with servers to determine whether they are malicious or benign. High-interaction client honeypots classify potentially malicious web pages by driving a vulnerable web browser to retrieve these pages and monitoring the system for unauthorized state changes, such as file modifications or process creations that would indicate a successful attack. Whether a state change is authorized or unauthorized is determined by an implied security policy that defines what is permitted and prohibited. A browser process, for instance, is expected to create new files in the browser cache, but is not permitted to create new files in the program startup group.

High-interaction client honeypots are tasked with classifying web pages as malicious or benign, and the ability to identify many malicious URLs quickly is a crucial task. Quick identification makes it possible to react to the threat in a timely manner, collect a large sample to evaluate attack trends, and identify zero-day attacks before they are widespread. As mentioned in the previous chapter, high-interaction client honeypots present challenges in achieving this goal: They are inherently slow and are known to miss attacks.

As research addresses these shortcomings, it becomes increasingly important to evaluate client honeypot technology. Evaluation not only al-

lows one to compare different technologies and improvements in the technologies, but also to detect and react to changes in the attack landscape.

In this chapter, we present a method for evaluating high-interaction client honeypots in an operating environment against their primary purpose, which is to identify malicious web pages in that environment. The method therefore takes into account the client honeypot technology as well as the operating environment as factors.

The method presented here is designed to evaluate a client honeypot's purpose of identification of malicious web pages. More specifically, it is designed to evaluate the ability to identify malicious web pages with identical resources. A client honeypot will evaluate better if it is able to identify more malicious web pages. Companies that provide blacklisting services against malicious web pages, such as browser software companies, search engines, or security perimeter defense companies may want to achieve this goal. However, alternatively there may be different applications for client honeypots. For instance, research institutions may want to develop client honeypots that are able to identify all malicious web pages in a sample, independent of the cost. This would, for instance, allow them to research advanced attack scenarios. The evaluation method presented in this chapter would not be suitable to evaluate client honeypots against the latter scenario.

After the method and the factors it considers are presented, we introduce and evaluate improvements on high-interaction client honeypot technology with this method. We present new visitation algorithms that show significant improvements to the ability of client honeypot technology to find malicious web pages on the network. In the last part of this chapter, we show how the characteristics of the operating environment impact the performance of a high-interaction client honeypot.

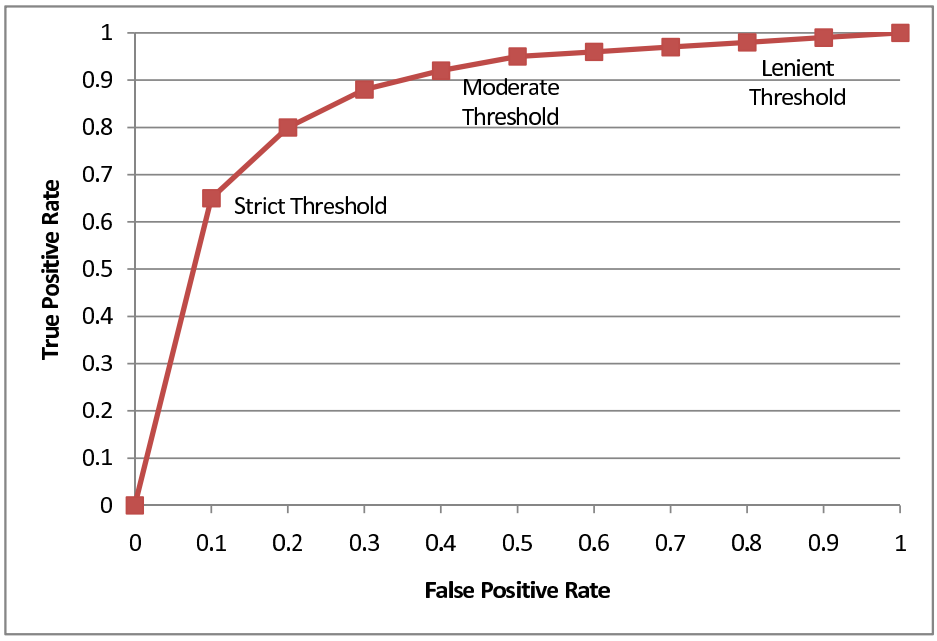


Figure 4.1: Receiver Operator Characteristics Curve Example

4.1 Evaluation

Within the general field of intrusion detection systems (IDSs), the need for evaluation has been identified for many years [78, 3, 10]. Effectiveness, efficiency, ease of use, security, interoperability, and transparency are some characteristics that could be evaluated. The effectiveness of an IDS at detecting intrusions has received intensive scrutiny in the research [78, 138, 10, 44, 57, 21, 84]. IDS effectiveness was initially simply expressed in the form of true positive and false positive rates. For anomaly-based IDSs, however, these figures are not sufficient, because a sensitivity threshold on those systems can be manipulated that affected these rates. Receiver operator characteristic (ROC) curves combine the true positive and false positive rates over a variety of sensitivity threshold settings into one graph as shown in Figure 4.1 [80]. However, while the concept of true positive and false positive rates is easily conveyed and understood, the true

positive and false positive rates are also quite deceiving, because distribution of attacks and non-attacks in the event stream, also known as the base rate, can lead to a counterintuitive distribution of attacks and non-attacks in the events for which the IDS raised alerts [10]. As Axelsson describes, because attacks are usually rare events, a set of alerts usually consists primarily of false positives. The Bayesian detection rate, expressing the probability of an intrusion in the case of an alert, was introduced to address this shortcoming of the true and false positive rates.

Cost-based models, in which costs are associated with the various conditions around events, attacks, and alerts, have been proposed [138, 44]. Cost and how those costs are linked to the base rate and detection accuracy of the system are easily understood. However, Gu et al. offer the critique that a lack of good risk analysis models makes objective selection of the cost factors difficult [57]. They propose a new metric, the Intrusion Detection Capability C_{ID}, which is founded in information theory. Its value is based on the reduced uncertainty of the input given the intrusion detection output. It is a numeric value that takes into account the true positive, false positive, and base rates and combines these into one number. With this abstraction, however, comes a disadvantage, in that it is difficult to link to specific quantities of interest to an operator of an IDS. For instance, an improved C_{ID} cannot be easily linked to the ability of an IDS to detect twice as many attacks as before. A sophisticated framework to evaluate IDSs was presented by Cardenas et al. [21]. They introduce the intrusion detection operating characteristic (IDOC) curves as a new IDS performance tradeoff that combines in an intuitive way the variables that are more relevant to the intrusion detection evaluation problem.

The evaluation techniques described above could be adopted to evaluate high-interaction client honeypots. However, unique characteristics of client honeypots make this approach impractical. First, the ability to detect attacks is different. High-interaction client honeypots are dedicated devices that find malicious web pages on a network and have a negligi-

ble false positive rate, which is primarily associated with incorrect security policies. While IDSs do struggle with false positives, high-interaction client honeypots primarily struggle with the false negative rate, i.e., their inability to identify all attacks. Second, high-interaction client honeypots are active devices that are tasked with finding malicious web pages. Resource costs associated with this task are of much greater importance than with IDSs.

We have developed the true positive cost curve for high-interaction client honeypots as a method for evaluation of high-interaction client honeypots in an operating environment as presented in this section. The method borrows some aspects of IDS evaluation, but also incorporates the unique characteristics of high-interaction client honeypots. At the core of the method stands the true positive cost curve, a simple yet effective method for evaluating high-interaction client honeypots in an operating environment. Because high-interaction client honeypots are primarily tasked with finding malicious web pages on a network, the true positive cost curve simply expresses the cost per malicious web page c_{URL} identified over the base rate p, the percentage of malicious web pages in the sample. Cost per malicious web page identified was chosen for the evaluation model because it maps to the primary goal of a high-interaction client honeypot – to identify malicious web pages – and can easily be compared. A client honeypot's performance is better if the overall cost to identify a malicious web page is lower. The base rate p is plotted on the x-axis to break out the impact of different base rates on the cost of identifying a malicious web page. As the base rate p increases and more malicious pages exist in the sample, the cost to identify a malicious web page is reduced, because a client honeypot will naturally identify more malicious web pages as it encounters more malicious web pages.

The example in Figure 4.2 shows the true positive cost curve for high-interaction client honeypots A and B. The y-axis plots the operating cost in US dollars to identify one malicious web page. High-interaction client

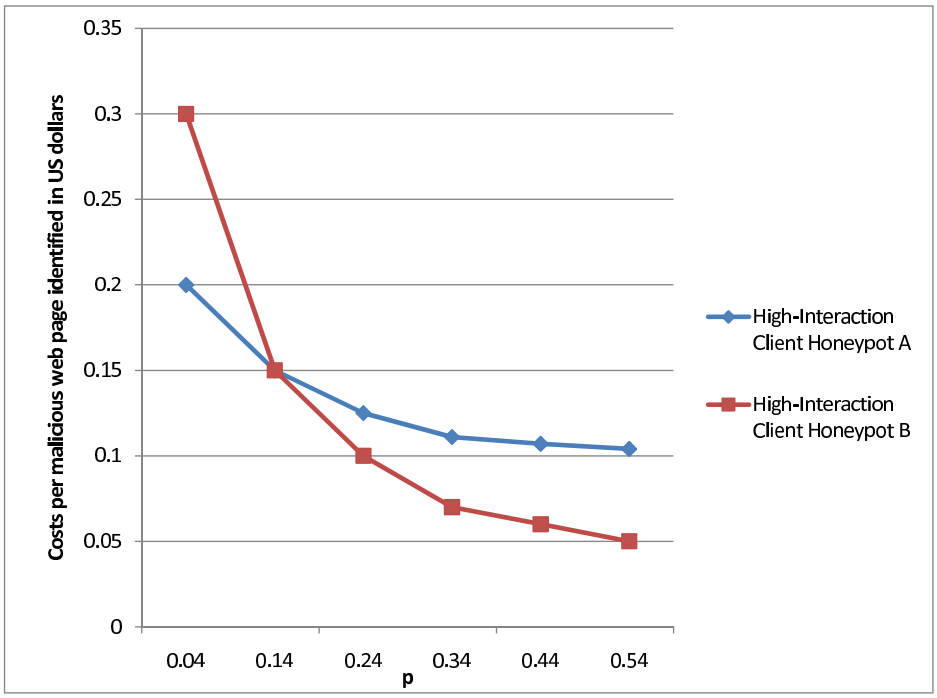


Figure 4.2: True Positive Cost Curve for High-Interaction Client Honeypots - Example

honeypot A shows a cost c_{URL} of about 0.30 US dollars for $p = 0.04$, whereas high-interaction client honeypot B shows a cost c_{URL} of about 0.20 US dollars. Client honeypot A is therefore more performant for that base rate. However, as the base rate increases, the two costs decrease at different rates. At a base rate of about $p = 0.14$, the cost of client honeypot B becomes lower than the cost of client honeypot A, and it remains lower with increasing base rate. As the figure shows, client honeypot B is more performant than high-interaction client honeypot A for greater values of p.

The example illustrates that the question of which client honeypot performs better than the other is not easily answered. The answer depends on a variety of factors. Client honeypot A's performance is better for a low and typical value of the base rate p. However, if the base rate were

to change because of a difference in attack landscape or the ability to feed malicious URLs with a high base rate to the client honeypot, client honeypot B would be more effective.

Cost is the central metric for evaluating high-interaction client honeypots. The factors that determine this cost are presented next.

4.1.1 Cost and Cost Factors

Cost associated with identifying a malicious web page c_{URL} is a simple yet effective way to compare client honeypots. Stolfo and Gaffney proposed to use cost to compare and evaluate IDSs [138, 44]. The main criticism of useing cost for evaluation is the uncertainty of the cost factors and the subjective association of cost. In their models, the classification errors of false positives and false negatives were the main driver of cost. Association of a cost that comes with a false alarm is indeed highly subjective. An operator has to investigate the alarm-associated attack and this can take a few minutes to days. It is even more difficult to place a price tag on missing an actual attack. Cost could range from the negligible cost associated with stealing a few processor cycles to the cost associated with recovering an entire data center. When carried into intangible assets such as losing a trade secret or the reputation of a company, association of an attack with a cost is even more difficult.

Cost needs to directly map to characteristics of the client honeypot in its ability to identify malicious web pages as well as the characteristics of the operating environment, as shown in Figure 4.3. These cost factors should be measurable and map to actual costs in an objective way. The main cost factors are speed, resource costs, detection accuracy, and the base rate. Operating characteristics, such as network location and evasion techniques employed by the attackers, manifest themselves with the detection accuracy for a given base rate. These characteristics are described below.

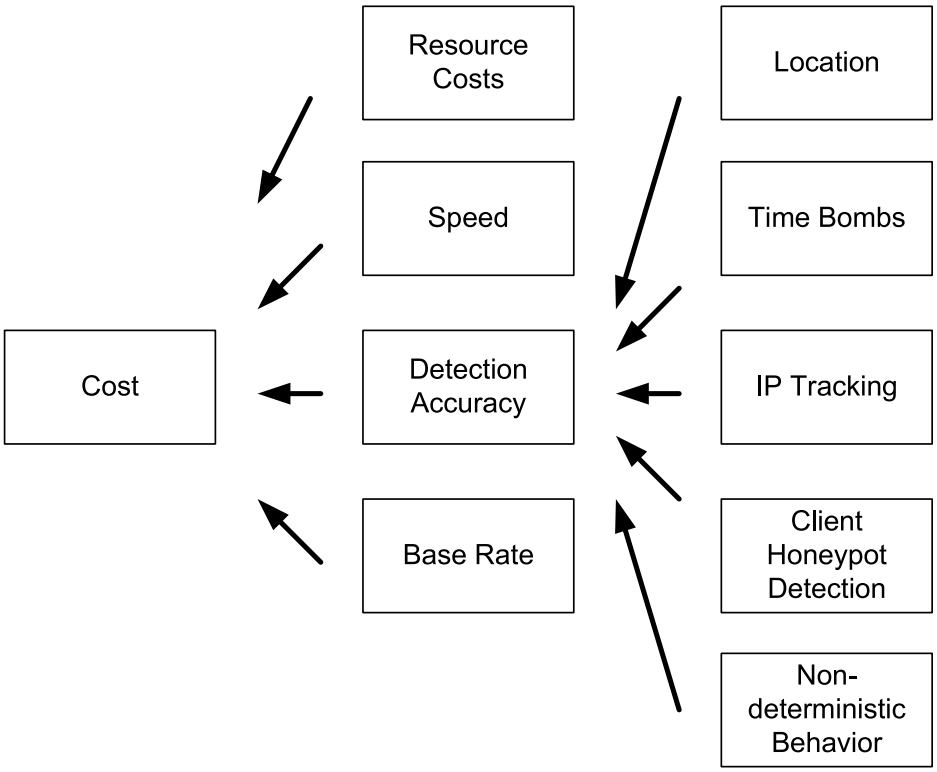


Figure 4.3: Cost Factors

Speed

High-interaction client honeypot speed influences certain operating costs. Between two otherwise identical high-interaction client honeypots, the one that is faster will be able to inspect more web pages over a given time frame and, if their ability to detect malicious pages is identical, the faster high-interaction client honeypot will detect a greater number of malicious web pages while consuming fewer resources at a lower cost. The desirable property of speed is therefore captured as part of the cost measurement.

Speed is expressed as t_{Algo}: seconds required for client honeypot to inspect a sample of N URLs.

Resource Costs

Clearly, resources are consumed while the high-interaction client honeypot inspects potentially malicious web pages. These resources, such as hardware costs as well as costs associated with network and power consumption, are captured by c_r. When utilizing machines in a computing cloud, usage of computing resources is often given in hourly rates that map to directly to c_r.

Detection Accuracy

Detection accuracy is the third factor that largely impacts operating costs. While a high-interaction client honeypot's false positive rate is negligible, the false negative rate FN drives the cost of identifying malicious web pages using a high-interaction client honeypot. The false negative rate expresses the failure of a high-interaction client honeypot to detect a malicious web page when it is inspecting one.

A malicious web page can employ techniques that cause the high-interaction client honeypot to fail at detecting it. The ability of a client honeypot to detect malicious web pages is heavily influenced by characteristics of the operating environment. The main characteristics are time bombs, location, IP tracking, client honeypot detection, and non-deterministic behavior. These are suspected to exist or have been observed by various studies [96, 159, 174].

- Time bombs are exploits contained on a malicious web page in which the exploit triggers only after a given period of time has elapsed.
- Location of a client honeypot is an operating characteristic that influences detection accuracy. Some malicious web pages selectively attack a client based on its location. A web server might launch an attack when accessed from Germany, but not if accessed from New Zealand. This behavior could even be influenced as granular as specific networks and IP addresses.

- IP tracking is a technique used by malicious web pages to launch an attack just once. Repeated interaction with the same page would result in the web server hiding the attack by serving a false benign page and, therefore, a client honeypot would fail to detect the malicious nature of the page.

- Client honeypot detection can cause a client honeypot to fail to detect a malicious web page. Passive fingerprinting [119], active fingerprinting [61], and behavioral aspects, such as access to robots.txt or not loading embedded images [67], are techniques a malicious web page can use to identify a client honeypot and selectively serve a benign web page instead of its usual malicious web page.

- Non-deterministic behavior of malicious web pages brings about sporadic attacks, causing a client honeypot to occasionally fail to classify the web page as malicious.

Between two otherwise identical high-interaction client honeypots, the one with the lower false negative rate will be able to identify more malicious web pages using the same resources, resulting in an overall lower cost per malicious web page identified. The desired characteristic of high detection accuracy is therefore captured as part of our cost measurement.

Base Rate

The base rate p, the percentage of malicious web pages in the sample that are inspected by the client honeypot, is the last factor that influences costs. While the base rate is not a characteristic of the client honeypot, but rather an operating characteristic, it directly impacts a client honeypot's ability to identify malicious web pages and therefore impacts cost. A client honeypot might be designed and optimized to identify malicious web pages with a high base rate, as with high-interaction client honeypot B shown in Figure 4.2. However, the cost associated with such a high-interaction

client honeypot might be much higher than the cost of using alternative client honeypot implementations for low base rates.

4.1.2 Calculation of Cost

$$c_{URL} = \frac{t_{Algo} c_r}{Np(1 - FN)} \tag{4.1}$$

The factors described in the previous section are applied to Equation 4.1 to calculate c_{URL}. Time t_{Algo}, which is the time to inspect the sample N, is multiplied by the resource costs per time unit c_r. It is divided by the number of malicious web pages identified in the sample, which is the number of web pages in the sample multiplied by the base rate and the true positive rate of the client honeypot: $Np(1 - FN)$.

A simple example illustrates how these factors can be mapped to our cost metric. While the base rate and subsequently the detection accuracy are not known, one can draw on existing measurement studies to estimate them. The existing measurement studies [159, 96, 110, 174] show an average base rate of approximately 0.1%. A client honeypot that is capable of inspecting sample N of 3,000 web pages in t_{Algo} 24 hours on a small Amazon EC2 cloud computing instance [4] will inspect approximately three malicious web pages a day. If the false negative rate is 33%, the client honeypot will identify two of these malicious web pages. With a cost c_r of 0.125 US dollars per hour (equivalent to the hardware specs of a small Amazon EC2 cloud computing instance [4]), the resource cost for the 24-hour period is 3.00 US dollars. As a result, the cost to identify a malicious web page c_{URL} with this client honeypot is approximately 1.50 US dollars.

Were the base rate to increase, the cost would decrease. Assuming a base rate of 0.5%, the client honeypot would inspect 15 malicious URLs. With the false negative rate of 33%, the client honeypot would identify approximately 10 malicious URLs. The cost to identify a malicious web page would fall to 0.30 US dollars.

However, this example is greatly simplified. A changing base rate is most likely to influence the speed and therefore hardware cost. An empirical evaluation utilizing a fully classified sample is desirable.

$$c_{URL} = \frac{t_{Algo}c_r + c_{MA}}{Np(1 - FN)} \tag{4.2}$$

As mentioned above, high-interaction client honeypots have a negligible false positive rate, which is why the false positive rate is not taken into account in Equation 4.1. Although false positives were not encountered in this evaluation, they are conceptually possible; for instance, false positives could result from incorrectly configured client honeypots (e.g., security policy, operating system) or advanced attacks that manipulate the client honeypot without being detected by it. The cost model could be extended to incorporate false positives by distributing the cost of manually analyzing all web pages that were classified as malicious c_{MA} across the number of malicious web pages that were confirmed to be malicious, as shown in Equation 4.2. However, once the cost of manual analysis is taken into account, special consideration needs to be given to keep this value objective, such as standardization of consultancy costs and analysis times.

An example illustrates how the false positives could be incorporated into our cost metric. We assume the identical client honeypot from the previous example. However, this time the client honeypot produces one false positive. As a result, three web pages are marked as malicious. A manual analysis of these three web pages to confirm the client honeypot's classification is assumed to take 30 minutes. With an hourly consultancy rate of 75 US dollars an hour, the cost C_{MA} would be 37.5 US dollars. The cost to identify the two malicious web pages, according to Equation 4.2, would rise to 40.50 US dollars or 20.25 US dollars per malicious web page (c_{URL}).

Despite this, experience has shown that a properly configured high-interaction client honeypot does not produce false positives. As a consequence, it is assumed the cost of false positives is negligible and, as a

result, that cost is not incorporated into the cost metric.

In chapter 6, we introduce a new type of client honeypot: the low-interaction client honeypot. It is capable of classifying web pages with a simulated client, but as a consequence produces many false positives. While it is theoretically possible to evaluate these low-interaction client honeypots with the TPCC, it would be difficult to remain objective and consistent as all the malicious web pages identified by the low-interaction client honeypot would need to be manually verified. The cost associated with such verification is likely to be high and would result in an overpowering cost factor in the evaluation. As a result, we employ the TPCC only on client honeypots whose false positive rate is negligible.

Next, we present improvements on the visitation algorithm of high-interaction client honeypots. We introduce two new visitation algorithms that significantly reduces the cost of the client honeypots. The two visitation algorithms directly influence the factors around speed and detection accuracy and are therefore good cases to evaluate with the presented cost metric and true positive cost curve. Following, the impacts of the characteristics of the operating environment on high-interaction client honeypots are presented and discussed. The true positive cost curves are used to demonstrate how the characteristics influence the cost of finding a malicious URL, a true positive, within an operating environment.

4.2 Improvements to High-Interaction Client Honeypots

Speed of a client honeypot is crucial and the improvements that we introduce in this section are primarily targeted at improving the speed of client honeypots and, as a result, bringing down the cost of identifying malicious web pages.

An estimation of the size of the threat illustrates why speed is so impor-

tant. According to a study in January 2005, 11.5 billion publicly indexable web pages exist [58]. According to Netcraft, approximately 9,800,000 web sites existed at that time, resulting in about 1173 pages/site on average. Since Janurary 2005, the Internet has grown significantly. Netcraft reports 66 million web sites in July 2008 [100]. Assuming the number of pages per site has not changed, the Internet consists of approximately 77 billion indexable web pages.

Various measurement studies exist that provide insight into the threat generated by these 77 billion indexable web pages. These studies have observed a percentage of malicious web pages that attack Microsoft's Internet Explorer 6 without a service pack (IE6SP0) ranging from 0.071% to 1.5% and a percentage of malicious web pages that attack Microsoft's Internet Explorer 6 with service pack 2 (IE6SP2) of 0.1% [157, 96]. A study conducted on Chinese web sites revealed a higher percentage of 1.38% [174].

Considering the web consists of approximately 77 billion indexable web pages in July 2008 and applying the values mentioned above, the upper bound of malicious web pages that attack IE6SP0 is estimated to be between 55 million and 1.1 billion. According to Moshchuk, there seems to be a bias of factor 15 on popular pages [96]. As such, the estimates could be reduced to one fifteenth, or 3.6 million to 77 million URLs. Google has already identified several million malicious URLs. As such, a lower bound of 3.6 million can be assumed. Assuming the measurement studies missed half of the attacks, the upper bound is likely to be higher at about 150 million URLs.

Identification of these malicious web pages is a large task and is infeasible with the current performance characteristics of client honeypots. A client honeypot does have a service time of a few seconds. According to Wang, most web pages exhibit malicious behavior within the first 30 seconds [158]. Even a sample of 1,000 malicious web pages would take a long time to collect or consume a lot of resources. With a base rate of

0.1%, 1,000,000 URLs would need to be inspected. If a client honeypot spends about 30 seconds to retrieve and inspect a web page, it would take approximately a year to inspect this number of pages. With significant resources, for instance 50 client honeypots, this time period could be reduced to about one week. The cost to identify 1,000 pages would be approximately 1,000 US dollars at an hourly rate of 0.125 US dollars per hour. This number certainly falls into a feasible range for small businesses or research institutions. However, identification of all 150 million malicious pages would run to millions of US dollars. Even with inspection of only a small portion of web pages, such as 1%, the cost would be almost one million US dollars. These figures are infeasible for small businesses and research institutions.

Being able to sample malicious web pages with high-interaction client honeypots, however, is an important task for both businesses and research institutions. A set of malicious web pages used for blacklisting needs to be comprehensive and up-to-date. Repeated measurement allows researchers to determine attack trends and anticipate future attack techniques. To collect a sample of sufficient size, the existing client honeypot technology needs to be improved upon and speed as a major cost factor is the hindering characteristic in achieving this goal.

Client honeypots consist of three components, as shown in Figure 4.4, that offer many opportunities to improve their speed.

- The Queuer component specifies what URLs to visit. A Queuer could intelligently select URLs based on characteristics of the URL or on classification history to increase the likelihood of finding a malicious web page.

- The Analysis Engine component assesses whether a web page is malicious by monitoring unauthorized state changes that result from a successful attack. Alternative means of detection – for instance, detection techniques that do not require a classification delay to be

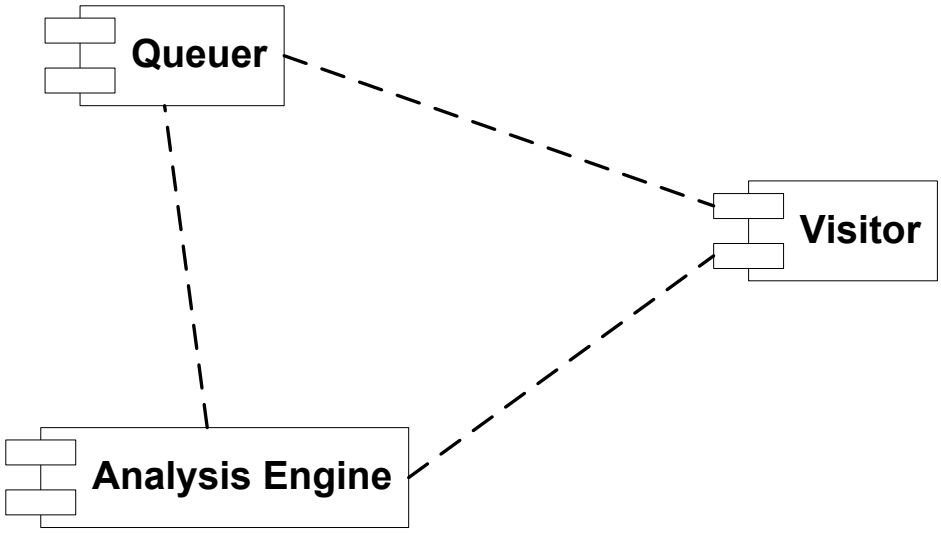


Figure 4.4: Client Honeypot Component Diagram

observed – could improve the speed of the client honeypot.

- The Visitor component retrieves and processes web pages. The visitor component poses many opportunities for speed improvements, which will primarily impact the speed of the high-interaction client honeypot with identical resources. The Visitor component is the focus of this section.

This section presents these improvements on client honeypot technology and evaluates them in the context of a client honeypot's ability to detect malicious web pages at a lower cost. Some existing algorithms are reviewed and contrasted to two new algorithms presented in this section. To evaluate the improvements, the presented true positive cost curves of high-interaction client honeypots are used.

4.2.1 Visitation Algorithms

The improvements on the Visitor component of a client honeypot will primarily impact the speed of the client honeypot with identical resources.

These speed improvements will increase the client honeypot's ability to inspect a sample of N web pages.

Four visitation algorithms – the sequential, bulk, bulk & sequential (BAS), and the divide-and-conquer (DAC) – are presented and evaluated. The sequential, bulk, and divide-and-conquer algorithms are evaluated using our high-interaction client honeypot Capture-HPC v1 and v2. The BAS algorithm is an algorithm that is implemented by HoneyMonkey [159] and not directly available. Nevertheless, it is included in the evaluation with the help of a simulator.

The algorithm used by HoneyClient v1, the first publicly available client honeypot [157], is not included in our comparison, because it does not implement a visitation algorithm that is able to specify which web page is malicious. Rather, HoneyClient v1 specifies that a malicious URL exists within a buffer of k web pages and requires additional manual analysis to determine which URL is the malicious one. While the cost of the manual analysis could be assigned to the cost of HoneyClient, the cost would certainly fall beyond economical levels even at a small scale. As a result, HoneyClient is not included in our comparison.

The speed of client honeypots to inspect a sample of N web pages is influenced by various factors. First, the underlying technology of how state changes are detected influences speed. HoneyClient v1, for example, takes snapshots of the system whereas the other implementations use event triggers that permit detection of the state changes as they occur. Independent of the mechanism to detect the state changes, there are additional factors that influence the total time t_{Algo} to inspect a sample of N web pages: the network bandwidth and average size of the request/response influence the time t_i to retrieve and t_d to render and display a web page, the classification delay t_w, the overhead of starting the client application t_s, and the overhead of resetting the client honeypot into a clean state after a malicious web page has been encountered t_r, which overall is impacted by the base rate of the web pages p. The classification delay is a purposefully

introduced waiting period after a web page has been received before a classification is made. This is introduced because some time passes before many exploits trigger. This might be due to the nature of the exploit or intentionally introduced by the attacker to avoid detection. When inspecting web pages with a client honeypot, the classification delay consumes most of the time. In addition, the time of creating a queue of URLs T_q, which is usually constant, is added to the duration to inspect web pages. $T_q, t_i, t_d, t_w, t_s, t_r$ are the six factors that we take into account for determining the speed of a client honeypot.

The client honeypot evaluation was conducted using the true positive cost curve described above. Data were collected by operating the client honeypot on an Intel Core2 Duo CPU E4500 with 2GB of RAM connected to cable Internet broadband. For the sequential, bulk, and DAC algorithms, the client honeypot, configured with Windows XP SP2 and IE6SP2 within the free VMware Server 1.x, was run against samples of 1,000 web pages that vary in base rate.

The sample of URLs for our evaluation was constructed by injecting URLs that point to manually constructed malicious web pages into a sample of randomly selected benign URLs from the Internet. Because speed is the focus of our evaluation, a realistic set of URLs that point to web pages of different size, media composition, and physical location was important to accurately capture the wide range of load and rendering times. The benign web pages were randomly sampled by issuing query terms to the Yahoo! search engine. Because these pages could also contain malicious pages, they were first inspected with a client honeypot prior to submitting them to the sample as benign pages.

Because the percentage of malicious web pages was small compared to the percentage of benign pages in the sample, obtaining malicious web pages of different size, media composition, and physical location was of lesser importance; manually constructed malicious web pages were therefore crafted using the Metasploit Framework [85]. These pages contain

an exploit that targets the MS06-014 vulnerability in the Microsoft Data Access Component [89]. Because the evaluation was assessing speed improvements resulting from optimizations on the Visitor component, a wide distribution of different attack types was of no importance, because the exploit is expected to have little impact on the performance of the Visitor component. This assumption is not likely to hold if optimizations on the Analysis Engine component were to be made. However, some malicious web pages were modified to exhibit the evasion technique of IP tracking, in which the web page initially exhibits malicious behavior, but on subsequent interactions does not. Those pages were included to create a more realistic sample. Some of the visitation algorithms require a web page to be retrieved multiple times before it can be classified as malicious, and would be negatively affected by IP tracking. However, a proxy server to cache all responses was used to neutralize the effect of IP tracking. Twenty percent of the malicious web pages in the sample employed the IP tracking evasion technique. This percentage was sufficient to include at least one malicious web page with IP tracking in the sample to illustrate that despite the IP tracking evasion technique, client honeypots – even the ones that require a web page to be retrieved multiple times before it can be classified as malicious – can successfully identify all malicious web pages in the sample.

If not otherwise stated, the client honeypot was configured in the following way: T_q is set to zero, because the time associated with creating a list of URLs is constant across visitation algorithms. The time to start the client application t_s, in other words the browser, is set to 0.5 second; the time to retrieve a web page t_i is set to 4.3 seconds; the time to render the web page t_d is set to 1.3 seconds; and the time to wait after the web page has been retrieved t_w is set to 25 seconds. The time to reset the virtual machine into a clean state t_r is set to 5 seconds. Many of the values are based on empirical evaluation on the hardware used for the evaluation.

The duration the client honeypot is running to inspect these pages is

mapped to the costs of operating a small Windows instance of Amazon EC2 at 0.125 US dollars per hour. The total cost is then divided by the number of malicious web pages identified to obtain the cost to identify one malicious web page according to Equation 4.1.

Because the HoneyMonkey system, which implements the BAS algorithm, is not publicly available, this algorithm was evaluated with a simulator. The simulator is a simple Java program that simulates the components of the client honeypot as well as the sample. It tracked the speed of the client honeypot in a global variable and, as actions of the client honeypot were conducted, the associated time was added to this global time variable. E.g., when a malicious URL was encountered, the simulator added 5 seconds to the global time variable. The simulator was calibrated using the empirical data collected for the other algorithms to accurately map to the characteristics of the physical system used in our experiments.

Sequential Algorithm

```
def find_malicious_web_pages_sequentially()
    url_queue = create_url_queue()
    while(url_queue.next?())
        url= url_queue.next()
        startClientApp()
        visit(url)
        wait()
        classification = check_state()
        closeClientApp()
        if(classification == MALICIOUS)
            //found malicious web page
            report(url)
            reset_state()
        end
    end
end
```

Figure 4.5: Sequential Algorithm

Capture-HPC v1 uses a sequential algorithm to inspect a sample of N web pages. It visits web pages sequentially and makes a classification after each web page has been retrieved. Figure 4.5 contains the pseudo code of this algorithm. After a queue of URLs has been created, each web page

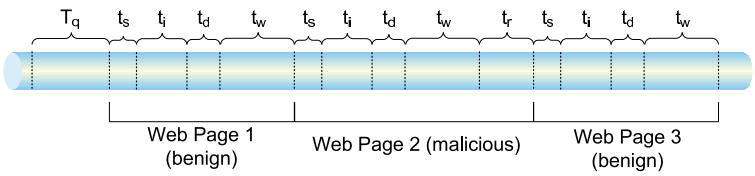


Figure 4.6: Sequential Algorithm Duration Example

is visited. After each visitation, the client honeypot waits before checking for state changes on the system to classify the web page as malicious or benign. If the web page was indeed malicious, the state of the system is reset. The computational time complexity is $O(n)$ as the time to inspect web pages increases by constant C_{Seq} with each additional web page, as shown in Figure 4.6. The total duration to inspect a sample of web pages N is given by the following equation:

$$\begin{aligned} t_{Seq} &= T_q + C_{Seq}N \\ &= T_q + (p(t_s + t_i + t_d + t_w + t_r) + (1-p)(t_s + t_i + t_d + t_w))N \end{aligned} \tag{4.3}$$

An empirical evaluation of the algorithm on the sample with various base rates showed that the client honeypot with the sequential algorithm was capable of inspecting the sample in approximately 33,000 seconds. A higher base rate resulted in a slight slowdown of the client honeypot associated with the action of resetting the state of the virtual machine. With a base rate of 0.4%, the client honeypot was capable of inspecting the sample of 1,000 web pages in 33,749 seconds; with a base rate of 4.4%, the client honeypot was capable of inspecting the sample in 34,172 seconds. While the malicious URLs did employ the evasion technique of IP tracking, all malicious web pages in the sample were successfully identified by the client honeypot because no repeated visits were conducted to identify a malicious web page. With the base rate of 0.4%, this translates to four malicious web pages identified; with the base rate of 4.4%, to 44 malicious web pages identified.

Equation 4.1 is used to calculate the cost associated with identifying

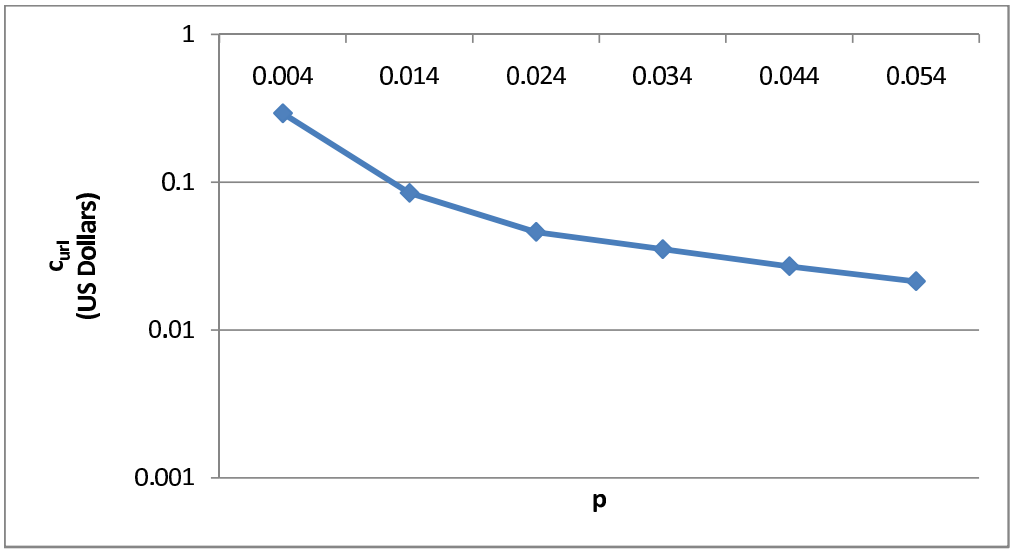


Figure 4.7: Cost per Malicious URL (Sequential Algorithm)

one malicious URL. For the base rate 0.4%, the time t_{Seq} is 33,749 seconds, or 9.37 hours, with a resource cost c_r of 0.125 US dollars per hour. The cost to identify a malicious web page c_{URL} is therefore 0.290 US dollars. As the base rate increases, the cost decreases. For the base rate 4.4%, the time t_{Seq} is 34,172 seconds, or 9.49 hours, with identical resource cost per hour. However, because the number of malicious URLs increases in the sample, the cost to identify one malicious web page decreases to 0.027 US dollars. The costs for the remaining base rates are shown in Figure 4.7.

Bulk Algorithm

The bulk algorithm was introduced with Capture-HPC v2.5. This algorithm visits a buffer of k web pages at the same time and is capable of specifying which URL in the buffer exhibited malicious behavior. Figure 4.8 contains the pseudo code for this algorithm. After a queue of URLs has been generated, a buffer of k web pages is taken from the queue and visited. After the pages have been loaded, the client honeypot waits before checking for state changes on the system to classify the web pages as malicious or benign. If a malicious web page was detected in the buffer of

```
def find_malicious_web_pages_bulk()
  url_queue = create_url_queue()
  while(url_queue.next?())
      buffer = get_buffer(url_queue, k)
      startClientApps()
      visitAllUrlsInBuffer(buffer)
      wait()
      classification = check_state()
      closeAllClientApps()
      if(classification == MALICIOUS)
          //found malicious web page in buffer
          report(url)
          reset_state()
      end
  end
end
```

Figure 4.8: Bulk Algorithm

k web pages, the state of the system is reset before the next buffer of k web pages is visited.

This algorithm is possible because of an enhanced state monitoring mechanism that was introduced with Capture-HPC v2.5. While the state monitoring mechanism of Capture-HPC v1 only reported state changes of the entire system and, as a result, was not able to link a state change to a browser process, the enhanced state monitoring mechanism reports state changes along with the process ID that is responsible for the state change. This permits inspection of web pages with several browser processes at once and, if a state change occurs, identification of the specific URL by the state change and browser process ID mapping. The bulk algorithm is only available when visitation of several web pages can occur in its own process. This is possible with Microsoft's Internet Explorer. However, Mozilla's Firefox, for instance, does not have this capability, as it optimizes all newly started browser processes into one process. While this strategy reduces resource consumption, it prevents inspection of URLs with the Mozilla Firefox web browser using the bulk algorithm.

The computational time complexity of the bulk algorithm is $O(n)$ as the time to inspect web pages increases by constant C_{Bulk} with each additional web page, as shown in Figure 4.9. As mentioned above, the web pages are inspected in a buffer of k web pages. This variable is primar-

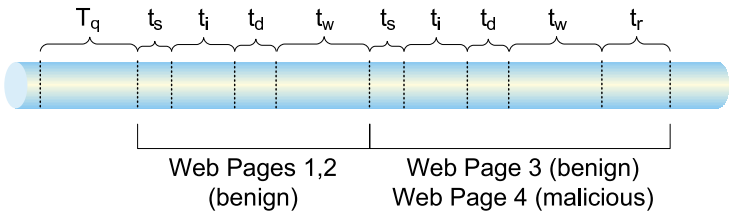


Figure 4.9: Bulk Algorithm Duration Example

ily limited by the resources of the machine and operating system. In our experiments, 54 browser processes could be opened simultaneously. Additional browser processes led to a crash of the operating system. Further, because multiple processes are opened, a lot of resources are consumed, which slows down normal operations, such as rendering a web page by a load factor. This load factor LF was experimentally determined to be a function of k: $LF(k) = \frac{4.5k}{54}$ for $k > 12$ and $LF(k) = 1$ for $k <= 12$. It is applied to determine the time to retrieve and render the page. Rendering the page is also different from the sequential algorithm, because all pages need to be rendered before the wait period is applied. As such, t_i is replaced with $max(t_i)$, which is the maximum time it takes to retrieve any web page in the buffer of k web pages. On the sample inspected, $max(t_i)$ was measured to be 46 seconds. The total duration to inspect a sample of web pages N is given by the following equation:

$$\begin{aligned} t_{Bulk} &= T_q + C_{Bulk}N \\ &= T_q + \Big(\frac{((1-(1-p)^k)(t_s k + max(t_i)LF(k) + t_d LF(k) + t_w + t_r))}{k} \\ &\quad \frac{((1-p)^k)(t_s k + max(t_i)LF(k) + t_d LF(k) + t_w)}{k}\Big)N \end{aligned} \tag{4.4}$$

An empirical evaluation of the algorithm on the sample with various base rates showed that the client honeypot with the bulk algorithm was capable of inspecting the sample in approximately 5,600 seconds. A very slight slowdown of the client honeypot associated with the action of re-

setting the state of the virtual machine is expected with higher base rates. However, because the time to reset the state is small compared to the time to visit a buffer of k web pages, the slowdown is not noticeable during our evaluation. While the malicious URLs did employ the evasion technique of IP tracking, all malicious web pages in the sample were successfully identified by the client honeypot because no repeated visits were conducted to identify a malicious web page. With the base rate of 0.4%, this translates to four malicious web pages identified; with the base rate of 4.4%, to 44 malicious web pages identified.

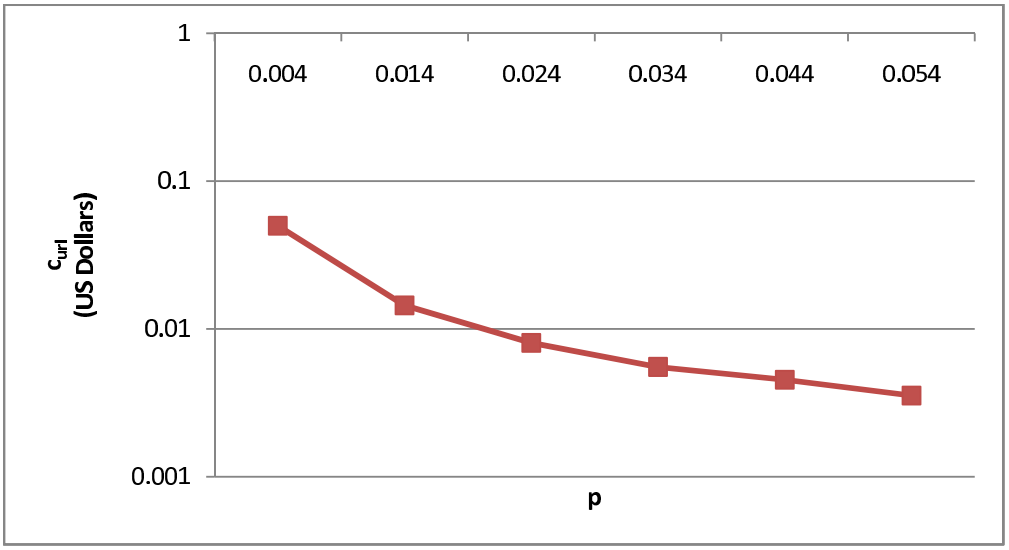


Figure 4.10: Cost per Malicious URL (Bulk Algorithm)

Equation 4.1 is used to calculate the cost associated with identifying one malicious URL. For the base rate 0.4%, the time t_{Bulk} is 5,600 seconds, or 1.59 hours, with a resource cost c_r of 0.125 US dollars per hour. The cost to identify a malicious web page c_{URL} is therefore 0.050 US dollars. As the base rate increases, the cost decreases. For a base rate 4.4%, the time t_{Bulk} is 5,600 seconds, or 1.59 hours, with identical resource cost per hour. However, because the number of malicious URLs increases in the sample, the cost to identify one malicious web page decreases to 0.005 US dollars. The costs for the remaining base rates are shown in Figure 4.10.

Bulk & Sequential Algorithm

```
def find_malicious_web_pages_bulk_and_sequential()
  url_queue = create_url_queue()
  while(url_queue.next?())
      buffer = get_buffer(url_queue, k)
      startClientApps()
      visitAllUrlsInBuffer(buffer)
      wait()
      classification = check_state()
      closeAllClientApps()
      if(classification == MALICIOUS)
          //found malicious web page in buffer
          reset_state()

          //find out which one in buffer is malicious
          //use sequential algorithm
          while(buffer.next?())
              url= buffer.next()
              startClientApp()
              visit(url)
              wait()
              classification = check_state()
              closeClientApp()
              if(classification == MALICIOUS)
                  //found malicious web page
                  report(url)
                  reset_state()
              end
          end
      end
  end
end
```

Figure 4.11: Bulk & Sequential Algorithm

The BAS algorithm is an algorithm used by the HoneyMonkey system [159]. It visits a buffer of k web pages at the same time and makes an initial classification about the entire buffer after the buffer has been inspected. This algorithm does not make use of the process ID mapping as the bulk algorithm does. As a result, after the buffer k has been inspected, the algorithm is not capable of pinpointing which URL was malicious. To determine this information, the buffer of k web pages is visited once more using the sequential algorithm. Figure 4.11 contains the pseudo code of this algorithm. After a queue of URLs has been created, each buffer of k web pages is visited. After each visitation, the client honeypot waits before checking for state changes on the system to classify the buffer as malicious or benign. As a malicious buffer is detected, the state of the system is reset and the buffer is visited once more using the sequential algorithm to determine which web page is malicious.

p	Optimum buffer size k
0.004	16
0.014	10
0.024	9
0.034	8
0.044	7
0.054	6

Table 4.1: Optimum k for Bulk & Sequential Algorithm

The size of the buffer k is dependent on the base rate p. If k is too large, the gain of using a buffer k in the first place is consumed by the expensive sequential algorithm. If k is too small, the gain of using a buffer k is minimal. As the base rate changes, the likelihood of a malicious web page in the set k increases and, as a result, the buffer k should be smaller. Table 4.1 shows the optimum values of k for various base rates. Because a buffer k is used, this poses some stress onto the system. However, because there is no need to have each browser in its own process, resources can be shared, which results in an overall lower load factor. Experiments on our test box showed that the load factor is a function of k: $LF(k) = \frac{3.3k}{54}$ for $k > 16$ and $LF(k) = 1$ for $k <= 16$ when opening a buffer of URLs with a shared browser process. Note that the load factor constant is lower than the load factor constant of the bulk algorithm. This stems from the fact that there is no need for each browser to be in its own process, which reduces the load on the system.

Revisitation of a URL, however, poses some risk to the success rate of detection. Malicious web pages, as identified by our previous work, use an anti-forensic feature designed to evade detection and make analysis of the malicious web page more difficult: IP tracking. As a client honeypot retrieves a web page multiple times, it runs the risk of failing to identify a malicious web page that employs IP tracking functionality. Alternatively, one could route all browser requests through a set of IP addresses or one could use a proxy server to cache all responses, so the second time the

browser requests a web page, it will be fetched from the local proxy cache instead of the malicious web server and is the one we adopted for the purpose of the evaluation of this algorithm. Section 4.3 demonstrates the impact of the scenario and shows how the two strategies impact costs.

Usage of a proxy has a desirable side effect. Because web pages are cached on the proxy, the time to retrieve a web page t_i is significantly reduced when retrieving the web page a second time. Instead of requesting the web page from the actual web server that is located on the Internet, the web page is rather requested from the proxy within the local network. The time t_i decreases to insignificant levels.

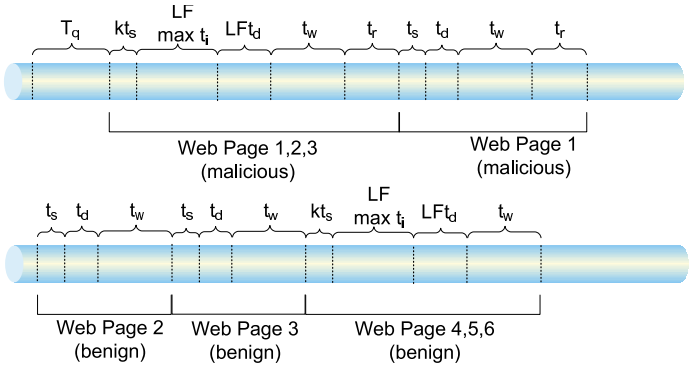


Figure 4.12: Bulk & Sequential Algorithm Duration Example

The computational time complexity is $O(n)$ as the time to inspect web pages increases by constant C_{BAS} with each additional web page, as shown in Figure 4.12. The total duration to inspect a sample of web pages N is given by the following equation:

$$t_{BAS} = T_q + C_{Bulk}N + C_{Seq}k \tag{4.5}$$

Because a system that implements this algorithm was not readily available, a simulator was used to inspect a simulated sample with various base rates. The simulator showed that the client honeypot with the BAS algorithm is capable of inspecting the sample in between 6,050 and 18,400 seconds depending on the base rate. The simulator did simulate malicious

web pages, of which 20% employed the evasion technique of IP tracking. However, since a proxy was used to cache the web pages, no repeated visits to the web server were made and, as a result, all malicious URLs in the sample were successfully identified. With the base rate of 0.4%, this translates to four malicious web pages identified; with the base rate of 4.4%, to 44 malicious web pages identified.

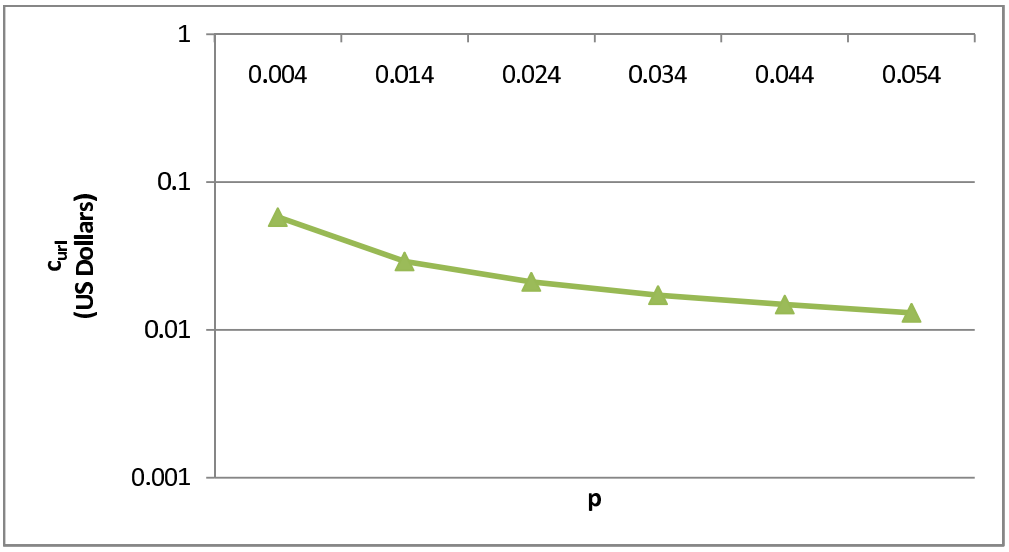


Figure 4.13: Cost per Malicious URL (Bulk & Sequential Algorithm)

Equation 4.1 is used to calculate the cost associated with identifying one malicious URL. Because web pages are retrieved twice, a proxy is employed to counter the potential IP tracking functionality, which increases the hardware cost. A proxy server is capable of serving multiple clients, but because the clients continuously retrieve content from the web, the proxy is utilized extensively. With these characteristics, a proxy server is capable of handling approximately 10 clients, which increases the resource costs c_r from 0.125 US dollars to 0.1375 US dollars. For the base rate 0.4%, the time t_{BAS} is 6,050 seconds, or 1.68 hours, with a resource cost c_r of 0.1375 US dollars per hour. The cost to identify a malicious web page c_{URL} is therefore 0.058 US dollars. As the base rate increases, the cost decreases. For a base rate 4.4%, the time t_{BAS} is 18,400 seconds, or 4.73 hours, with

identical resource cost per hour. However, because the number of malicious URLs increases in the sample, the cost to identify one malicious web page decreases to 0.015 US dollars. The costs for the remaining base rates are shown in Figure 4.13.

Divide-and-Conquer Algorithm

```
def find_malicious_web_pages_dac()
  url_queue = create_url_queue()
  while(url_queue.next?())
      buffer = get_buffer(url_queue, SIZE)
      cache_buffer(buffer)
      find_malicious_web_pages_in_buffer(buffer)
  end
end

def find_malicious_web_pages_in_buffer(buffer)
      startClientApps()
      visitAllUrlsInBuffer(buffer)
      wait()
      classification = check_state()
      closeAllClientApps()
      if(classification == MALICIOUS)
      reset_state()
      if(buffer.size == 1)
          //found malicious web page
          report(url)
      else
          first_half = buffer.get_first_half
          find_malicious_web_pages_in_buffer(first_half)
          sec_half = buffer.get_sec_half
          find_malicious_web_pages_in_buffer(sec_half)
      end
  end
end
```

Figure 4.14: Divide-and-Conquer Algorithm

The DAC algorithm was first implemented in Capture-HPC v2.1. The algorithm makes use of the well-known divide-and-conquer design paradigm. It visits a buffer of k web pages at the same time and makes a classification after the buffer has been inspected. This algorithm does not make use of the process ID mapping as the bulk algorithm does. As a result, after the buffer k has been inspected, the algorithm is not capable of pinpointing which URL was malicious. To determine this information, the buffer of k web pages is divided in two portions and recursively visited until the malicious web page or pages are identified. Figure 4.14 contains the pseudo code of this algorithm. After a queue of URLs has been created, each buffer of k web pages is visited. After each visitation, the client hon-

eypot waits before checking for state changes on the system to classify the buffer as malicious or benign. As a benign buffer is detected, all web pages in the buffer are classified as benign. As a malicious buffer is detected, the buffer is split into two and recursively visited. If the buffer k is of size 1 and no unauthorized state change is detected, the web page is classified as benign; otherwise, it is classified as malicious.

Dividing the sample of URLs N into buffers of size k will select malicious web pages according to the binomial distribution. Depending on how many malicious web pages have been selected, the number of operations to identify each malicious web page using the algorithm above differs. With the selection of zero malicious web pages, the algorithm will operate once on the buffer and exit. If one malicious web page appears in the group, the algorithm will operate $2log_2(k) + 1$ times on the buffer to identify the malicious web page. If two or more malicious web pages m appear in the group, the algorithm will traverse down the binary tree for each malicious web page m, so there will be $m(2log_2(k) + 1)$ operations. However, some operations at the top of the binary tree are shared as the branching occurs lower down the tree. The shared number of operations has to be subtracted, so the worst case total number of operations to identify malicious web pages using this approach is:

$$op(k, m) = \begin{cases} 1 & \text{if } m = 0, \\ m(2log_2(k) + 1) & \\ (2mlog_2(m) - m + 1) & \text{if } m > 0. \end{cases} \tag{4.6}$$

The number of operations executed that do not identify a malicious web page is:

$$op_b(k, m) = \begin{cases} 1 & \text{if } m = 0, \\ ((log_2(k) + 1) & \\ \quad (log_2(m) + 1))m & \text{if } m > 0. \end{cases} \tag{4.7}$$

and the number of operations executed that do identify malicious web

pages is:

$$op_m(k, m) = m(log_2(k)log_2(m) + 2)1 \tag{4.8}$$

where $m > 0$.

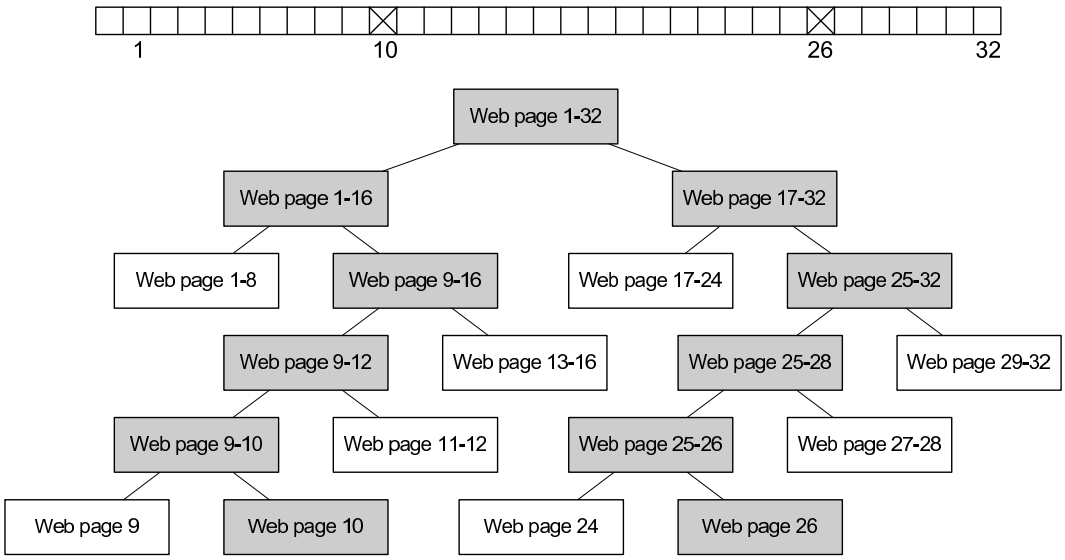


Figure 4.15: Divide-and-Conquer Algorithm Example

Figure 4.15 shows a simple example. A buffer of 32 web pages is inspected by the client honeypot using the DAC algorithm described above. First, web pages 1-32 are inspected. A malicious web page is detected in this buffer, so the buffer is divided in two and web pages 1-16 and 17-32 are inspected. Since both halves indicate that malicious web pages exist, the buffers are further divided and inspected (web pages 1-8, 9-16, 17-24, and 25-32). In the buffers with web pages 1-8 and 17-24, no malicious web pages are identified and, as such, no further investigation is made into those buffers. In the remaining buffers, however, malicious web pages are once again identified and the algorithm is applied recursively until malicious web pages 10 and 26 are identified. The tree is traversed twice to identify each web page, but branching took place after buffers with web pages 1-16 and 16-32 identified a malicious web page. A total of 19 oper-

ations are counted, of which 11 operations require the state of the client honeypot to be reset.

As described above, the number of operations to identify a malicious web page in a buffer is determined by the actual number of malicious web pages in the buffer. This number of malicious web pages m that are selected with a buffer of size k and a likelihood of selecting malicious web pages p is given by the binomial distribution:

$$f(m;k;p) = \binom{k}{m} p^m (1-p)^{k-m} \tag{4.9}$$

The binomial distribution needs to be taken into account when calculating the total time to inspect web pages:

$$\begin{aligned} t_{DAC} &= T_q + C_{DAC} N \\ &= T_q + \frac{max(t_i)k}{N} + \frac{N}{k}(f(0;k;p)(t_s k + t_d LF(k) + t_w) \\ &\quad + \sum_{0<m\leq k} f(m;k;p)(op_b(t_s k + t_d LF(k) + t_w) \\ &\quad + op_m(t_s + t_d LF(k) + t_w + t_r) \end{aligned} \tag{4.10}$$

The total time is calculated by adding the time to create the queue of URLs and to retrieve the web pages. Note that despite the fact that web pages are repeatedly retrieved, the time to retrieve all web pages is only taken into account once. This is because, similar to the BAS algorithm, the web pages are cached on a proxy server and the time to retrieve the web page from this local proxy is negligible. In addition to this time, the waiting and reset time according to the binomial distribution and number of operations necessary to identify the malicious web page(s) is added. The overall computational complexity of this algorithm is $O(n)$.

The buffer size k can be set to any value according to Equation 4.10. However, there are good and bad values for k. If k is too small, the algorithm will behave similarly to the sequential algorithm. If the buffer is too large, we could select too many malicious web pages in the buffer, leading to less efficient identification than when the buffer is split into smaller

p	Optimum buffer size k
0.004	80
0.014	27
0.024	16
0.034	11
0.044	9
0.054	7

Table 4.2: Optimum k for Divide-and-Conquer Algorithm

buffers. The optimum value of k is given by the global minimum of the function that captures the total number of operations taking into account the binominal distribution. Table 4.2 shows the optimum values of buffer size with varying values of p.

Because a buffer k is used, this poses some stress onto the system. However, because there is no need to have each browser in its own process, resources can be shared, which results in an overall lower load factor. Experiments on our test machine showed that the load factor is a function of k: $LF(k) = \frac{3.3k}{54}$ for $k > 16$ and $LF(k) = 1$ for $k <= 16$ when opening a buffer of URLs with a shared browser process. Note that the load factor constant is lower than that of the bulk algorithm. This stems from the fact that there is no need for each browser to be in its own process, which reduces the load onto the system.

The buffer size k for the bulk algorithm described in section 4.2.1 was limited to 54 due to operating system crashes. The DAC algorithm does not have this limitation because each browser is not required to be in its own process. However, to objectively compare the performance of the algorithms, we also evaluated the DAC algorithm with a max buffer size k for base rate 0.4% of 54. The duration and corresponding cost are given in parentheses and represented by the black line in the Figures below. As expected, with the un-optimized buffer size, the performance is slightly worse.

An empirical evaluation of the algorithm on the sample with various

base rates showed that the client honeypot with the DAC algorithm was capable of inspecting the sample in between approximately 4,200 (5,000) and 17,800 seconds. While the malicious URLs did employ the evasion technique of IP tracking, all malicious web pages in the sample were successfully identified by the client honeypot because the proxy cached web pages locally and, as a result, no repeated visits to the web server needed to be made. With the base rate of 0.4%, this translates to four malicious web pages identified; with the base rate of 4.4%, to 44 malicious web pages identified.

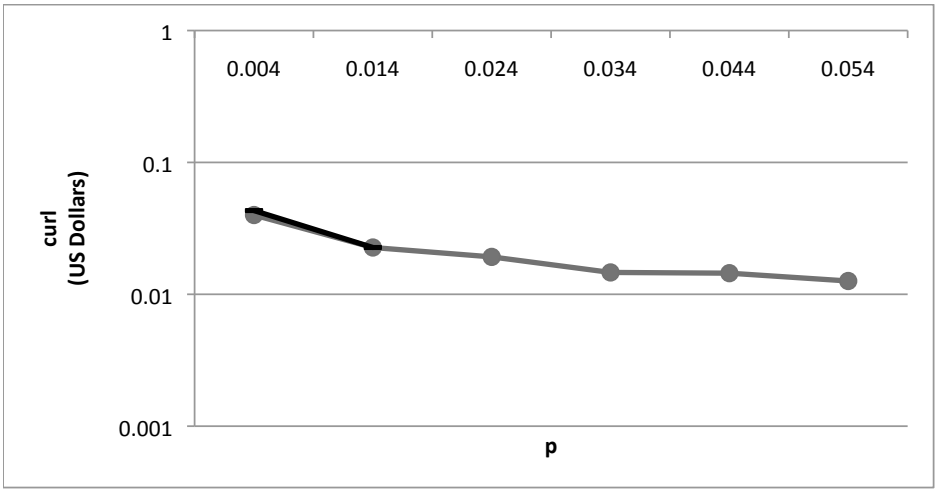


Figure 4.16: Cost per Malicious URL (Divide-and-Conquer Algorithm)

Equation 4.1 is used to calculate the cost associated with identifying one malicious URL. For the base rate 0.4%, the time t_{DAC} is 4,200 seconds (5,000 seconds,) or 1.16 hours (1.39 hours,) with a resource cost c_r of 0.1375 US dollars per hour. The cost to identify a malicious web page c_{URL} is therefore 0.040 US dollars (0.043 US dollars). As the base rate increases, the cost decreases. For the base rate 4.4%, the time t_{Seq} is 17,800 seconds, or 4.62 hours, with identical resource cost per hour. However, because the number of malicious URLs increases in the sample, the cost to identify one malicious web page decreases to 0.014 US dollars. The costs for the

remaining base rates are shown in Figure 4.16.

4.2.2 Visitation Algorithms Summary

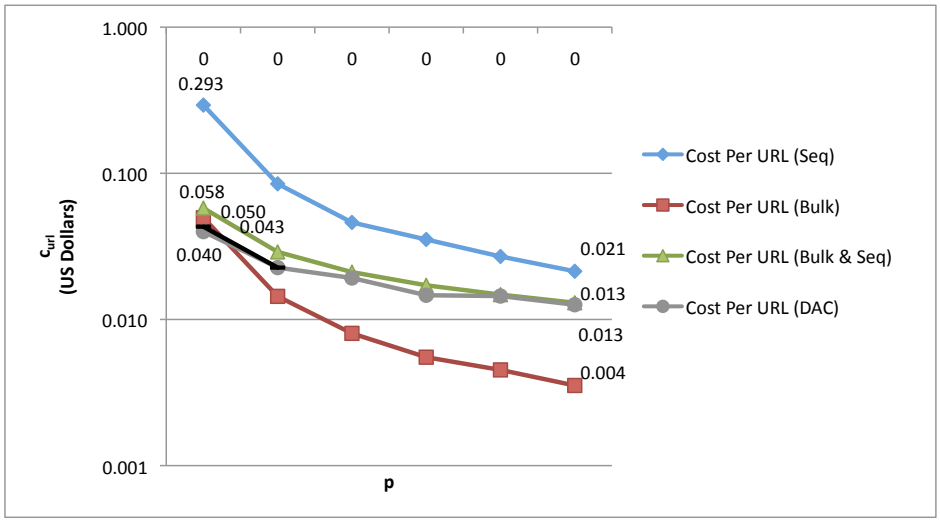


Figure 4.17: Cost per Malicious URL (All Algorithms)

In the previous section, four visitation algorithms were introduced and compared using the true positive cost curves of high-interaction client honeypots, which capture the cost associated with identifying one malicious URL. Figure 4.17 shows the true positive cost curve for the sequential, bulk, BAS, and DAC algorithms. The sequential algorithm is costliest for all base rates; the cost of identifying one URL is 0.293 US dollars for a base rate of 0.4% and 0.021 US dollars for a base rate of 5.4%. However, no one algorithm exists that performs best for all base rates. Rather, the DAC algorithm seems to perform best for very low base rates, such as 0.4%; the cost associated with finding a URL with this base rate is 0.040 US dollars (0.043 US dollars.) As the base rate rises, the bulk algorithm becomes the best performer, with a cost that stays below that of the DAC algorithm for all base rates higher than 0.4%. At a base rate of 5.4%, its cost is 0.004 US dollars compared to 0.013 US dollars for the DAC algorithm.

The BAS and the DAC algorithms both required a proxy to be incorporated into the setup to address the commonly encountered IP tracking functionality of malicious web pages that would result in false negatives. The true positive cost curves were able to incorporate the additional cost of the proxy to allow for a comparison between the client honeypot systems with different visitation algorithms and hardware setups.

The performance of the algorithms is dependent on the factors of the operating environment. In the comparison described above, the base rate was the only major factor varied, while all other factors were kept constant. From this, we conclude that the DAC algorithm performs best for low base rates, whereas the bulk algorithm performs best for higher base rates in *our operating environment*. The following section presents the impact of changes in the operating environment.

4.3 Impacts of the Characteristics of the Operating Environment on High-Interaction Client Honeypots

As mentioned in the previous section, the performance of the high-interaction client honeypot is dependent on the characteristics of the operating environment. In this section, we present two examples that demonstrate the impact of the characteristics of the operating environment on the performance of client honeypots. With the help of true positive cost curves, an operator can tune the client honeypot according to the environment. To generate the true positive cost curve for client honeypots, we utilize a simulator because the operating characteristics might be unknown or difficult to control in a real world setting.

First, the impact of time bombs is illustrated. Time bombs are exploits embedded in malicious web pages that trigger only after a few seconds have passed. Time bombs are the primary reason why a high-interaction

client honeypot waits t_w seconds after a web page has been retrieved. The value in our comparison above was set to 25 seconds, because the majority of web pages seem to launch an attack during that time frame [158]. If attackers change the trigger time of their time bombs, the ability of a client honeypot to identify malicious web pages, and therefore the cost to identify a malicious URL, is impacted.

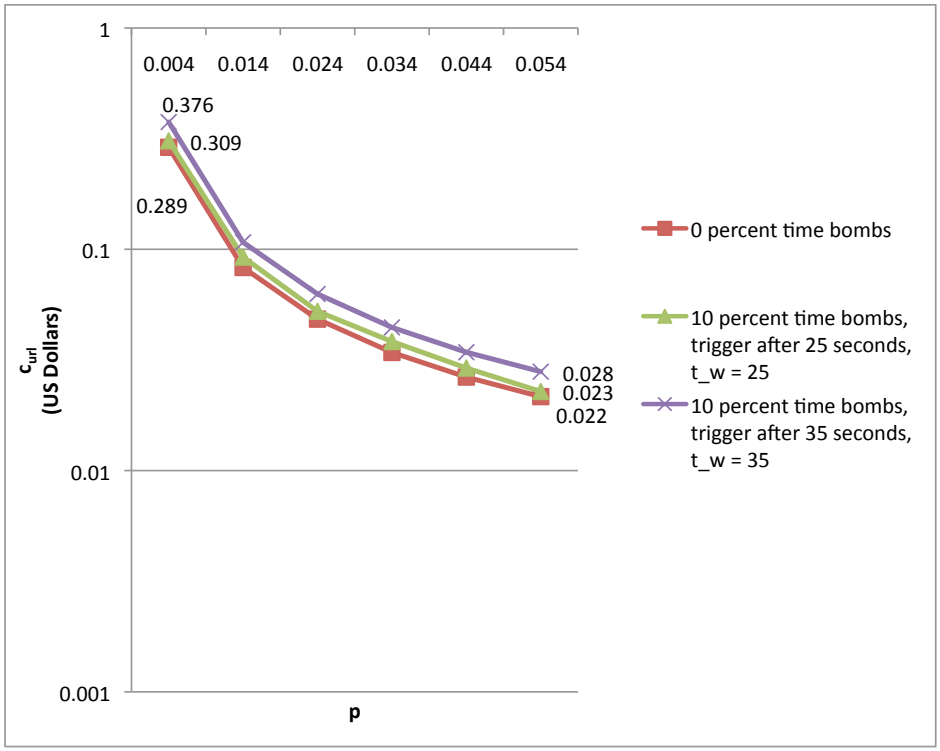


Figure 4.18: Cost per Malicious URL with Time Bombs (Sequential Algorithm)

In a hypothetical scenario, it was assumed that 10% of attackers lengthened their time bomb setting from 25 to 35 seconds. Independent of the action taken by the operator, the cost of identifying malicious web pages will increase, because the strategy employed by attackers makes detection more difficult. The true positive cost curve for client honeypots, however, assists the operator deciding whether the classification delay should

be increased to detect web pages that employ time bombs, or whether the classification delay should remain unaltered, with the result that some malicious web pages will not be detected. Figure 4.18 shows the costs of identifying malicious web pages under the various scenarios using the sequential algorithm. Cost is the greatest if the classification delay t_w is increased to counter the evasion technique employed by the malicious web pages. Therefore, if 10% of malicious web pages trigger only after 35 seconds, it is best for the operator to ignore these 10% and continue to operate the client honeypot unchanged with a classification delay t_w of 25 seconds.

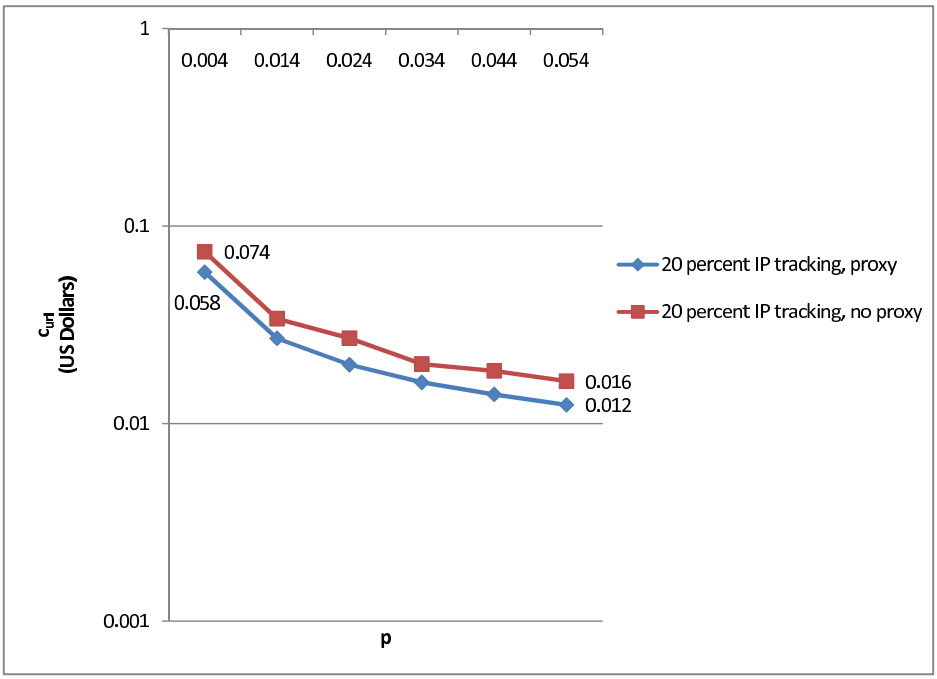


Figure 4.19: Cost per Malicious URL with IP Tracking (Bulk & Sequential Algorithm)

Another example illustrates the impact of IP tracking functionality. It is assumed that approximately 20% of malicious web pages employ the technique of IP tracking. As mentioned above, IP tracking is a technique designed to make the analysis and detection of malicious web pages more

difficult. A malicious web page that employs this technique will trigger an attack upon first visitation of a client, but ceased to do so with subsequent visitations. If an operator utilizes a visitation algorithm, such as the BAS algorithm, which visits a web page repeatedly, a potential for not detecting the web page exists. The operator has the choice of accepting this or to counter the IP tracking functionality by caching responses on a proxy, as we have chosen to do in our examples above. The proxy, however, increases the overall hardware cost. Alternatively, an operator can accept missing such web pages and, as a result, gets the benefit of the lower hardware costs. Figure 4.19 shows the effect of the two strategies on cost. For all base rates, the cost to identify a malicious URL is higher without a proxy than with a proxy. This is primarily attributed to the fact that without a proxy, the client honeypot becomes slower as no caching mechanism exists and fewer malicious web pages are successfully identified due to the IP tracking functionality. The true positive cost curve assists an operator in making this decision.

The previous two examples demonstrate that the true positive cost curves can be used to assist an operator in adjusting and configuring a client honeypot based on the operating characteristics of the operating environment. We illustrated how the phenomena of time bombs can be addressed in the optimal manner using true positive cost curves; further, we demonstrated how the true positive cost curves can be used to determine whether usage of a proxy is the more cost-effective strategy to counter IP tracking functionality of malicious web pages.

4.4 Summary

In this chapter, we have presented the true positive cost curve model for evaluating high-interaction client honeypots. The true positive cost curve is a cost-based model that directly links a client honeypot to its primary goal of identifying malicious web pages. The model accurately incorpo-

rates factors relevant to a client honeypot's performance, such as speed and detection accuracy, as well as the characteristics of the operating environment that may influence a client honeypot's performance, such as base rate and evasion techniques utilized by some malicious web pages.

The sequential, bulk, BAS, and DAC algorithms were introduced, presented, and compared. An empirical evaluation of these algorithms using the true positive cost curve demonstrated that both the bulk and DAC algorithms showed superior performance over the other two algorithms. For low base rates, the DAC algorithm, which we developed in our previous work, outperformed the bulk algorithm. For higher base rates, the bulk algorithm showed the best performance.

The effects of changing operational characteristics were also presented, showing how time bombs and IP tracking functionality influence the performance of a client honeypot in its ability to identify malicious web pages. The true positive cost curve was used to evaluate different strategies an operator can use to fine tune a client honeypot system for optimal performance within its operating environment.

Chapter 5

Experimental Design

In the previous chapter, we evaluated client honeypots within an operating environment using the true positive cost curve. As long as the operating environment stays constant, risks to the validity of such an evaluation are low. However, risks to validity arise in experimental designs that aim at identifying malicious web pages. In those settings, there is a risk to the intent of the experimental design (internal validity) and also risk about the generalizability beyond the experimental setting (external validity.)

Internal validity captures the intent of a study. If, for instance, one would like to measure the prevalence of malicious web pages on the network over a 12-month period to make a statement about trends, the internal validity is easily at risk with flaws in the experimental design. If the researcher chose to operate a client honeypot from one static IP address and a decreasing trend were observed, it would be questionable whether the trend is associated with a decreasing prevalence of malicious web pages. Rather, it is possible that the researcher's client honeypot's effectiveness in detecting malicious web pages is decreasing due to the IP tracking functionality we described in the previous chapter. External validity is the ability to generalize results beyond an experimental setting. If one, for instance, measures the prevalence of malicious web pages in search engines by inspecting the top three pages associated with popular search engine

queries, one could not generalize their findings to the entire search engine index. While the researchers have some knowledge about popular URLs, they have no information about other URLs that are served by the search engine.

In Chapter 3, we identified that results of various measurement studies [96, 159] varied widely and raised the question of whether those fluctuations were a direct result of risks to internal and external validity of the studies resulting from a failure to identify and adequately mitigate the risks. In this chapter, we present a methodology of identifying and mitigating risks in a systematic and thorough manner: the hazard and operability (HAZOP) study. Application of HAZOP to experimental design in computer science has previously been proposed by Welch et al. [166].

We apply the HAZOP to the experimental design of measurement studies of malicious web pages with client honeypots. We choose to focus on measurement studies, because, as mentioned above, existing studies appear to fail in identification and mitigation of risks. We also focus on measurement studies because the measurement is important for economic modeling, as illustrated by a variety of security-related economic studies [150, 67, 174, 62]. The business model [37, 39] of the operation behind the malicious web page can be used to devise strategies to break the business model. Accurate measurement provides the inputs for accurate business models. While we focus on measurement studies, HAZOP is not limited to such studies. In the area of identification of malicious web pages with client honeypots, whether for measurement studies or studies to develop new detection methods, many risks identified are universal. Mitigation of risks identified using the HAZOP were incorporated into our experimental designs.

This chapter is structured as follows. First, HAZOP is introduced and applied to the experimental design of a measurement study. The second part of the chapter illustrates what happens to the validity of a measurement study when risks are not appropriately mitigated. Major threats to

internal and external validity are uncontrolled variables. We illustrate the impact of uncontrolled variables on the internal and external validity of measurement studies. First, we illustrate that the URL source can greatly impact measurements; second, we show how time also has a major impact on measurement.

5.1 HAZOP

HAZOP is a systematic and thorough technique for identifying hazards and problems [70]. It originated in the chemical industry and has been generalized for use in areas as diverse as critical appraisal of proposals to release genetically modified organisms into the environment and chemical experiments. HAZOP is a generative technique that aims to discover new, unforeseen hazards through a structured process involving experts from the domain being analyzed, so the hazards can be appropriately mitigated.

HAZOP is conducted in the following manner: 1. The process to be HAZOP-ed is described in detail in the form of a flow chart. 2. Domain-specific guide words are applied to components of the process to generate possible deviations from the intended purpose of the component. The guide words are applied to the artifacts: subjects that participate in the experiment and the apparatus used to conduct the experiment, as well as the specific stimuli used during the experiment. 3. As the hazards are identified, severity and likelihood are assessed and mitigation to the hazard is described. 4. Mitigations for hazards with high severity and/or likelihood are incorporated into the experimental design.

For example, in the chemical industry, a flow chart showing the flow of materials around the system to create a chemical compound would have guide words such as MORE or LESS applied to individual pipes of the apparatus. Experts use these guidewords as prompts to help them identify a hazard, for example a LESS than flow may lead to a highly unstable chemical compound with risk of explosion. The expert considers this to be a low

likelihood, but the consequence of an explosion would be severe. Therefore, it is worth mitigating the risk by increasing the carrying capacity of the pipe.

Key to the use of HAZOP in a new domain are the development of guide words appropriate to the description of the measurement process and artifacts that influence the measurement process. These three components are described first before the HAZOP is applied to the process of measuring malicious web pages with high-interaction client honeypots.

The guide words are developed based upon our own experience in operating client honeypots to find malicious web pages on a network and guidance in the literature on good experimental design and procedure. The guide words chosen are NO, MORE, LESS, LATE, EARLY, FEWER, MORE, OUT OF ORDER, INDISTINGUISHABLE, UNRELIABLE, BIASED, and HISTORY.

We first create a flow chart to show the process under analysis. Figure 5.1 depicts how measurement data of malicious web pages is collected. It shows the various components and the process steps executed by the components to identify malicious web pages on the network using client honeypots: the apparatus (client honeypot) and external entities (web pages, DNS servers and URL store.) The apparatus itself consists of three components: the Queuer, Visitor and Analysis Engine, which are deployed across two entities: the controller and the actual client, which interacts with potentially malicious web pages. The controller contains the Queuer component responsible for feeding URLs to the client. The client contains the Visitor, which makes requests to potentially malicious web pages, as well as the Analysis Engine, which records any unauthorized state changes. The Analysis Engine also resides on the controller. It makes a classification based on the state changes and if necessary resets the client into a clean state before more URLs are fed to the client. Each step of the flow diagram is described in detail below.

1. The queue on the controller creates a list of potentially malicious

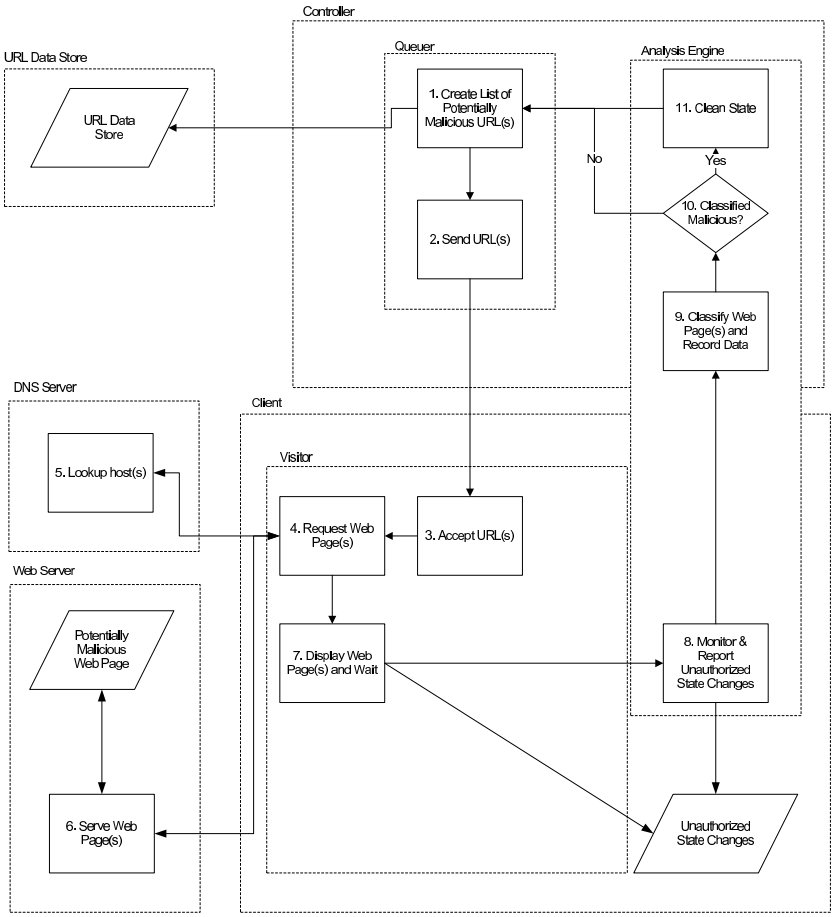


Figure 5.1: Flow Diagram of Measurement

URLs. These URLs are sourced from a URL data store, which could be a search engine, the web map of the World Wide Web, or some other means.

2. Once the URLs are sourced, they are sent to the visitor component on the client.

3. The Visitor component of the client accepts the URLs from the Queuer.

4. The Visitor proceeds to request the web pages. This involves resolving the host names into IP addresses by making a DNS lookup re-

quests to the local DNS server as well as a HTTP requests to the web servers that host the potentially malicious web pages.

5. The DNS server accepts the DNS lookup requests by the client and responds with a IP addresses for the host names contained in the DNS lookup requests.

6. The web server accepts the HTTP requests by the client and responds with the web pages requested.

7. The Visitor component displays the web pages and waits a few seconds to give the potential exploit the opportunity to launch an attack.

8. The Analysis Engine on the client reports any unauthorized state changes that may result from an attack to the Analysis Engine on the controller.

9. The Analysis Engine on the controller classifies the web pages according to the unauthorized state changes. It records this information as well as additional data collected, such as network traces, for later analysis.

10. The next processes depend on this decision point. If the web pages were classified as malicious, process 11 is executed. If the web pages were benign, process 1 is executed continuing the data collection.

11. The state of the client is dirty as the malicious web page successfully exploited the client and was able to modify the client system to its liking. As such, the client system is no longer trusted and is being reset into a clean state. Once the client has been reset into a clean state, process 1 is executed, continuing the data collection.

The description of the flow chart already explicitly mentions the apparatus with its three main components Queuer, Visitor and Analysis Engine.

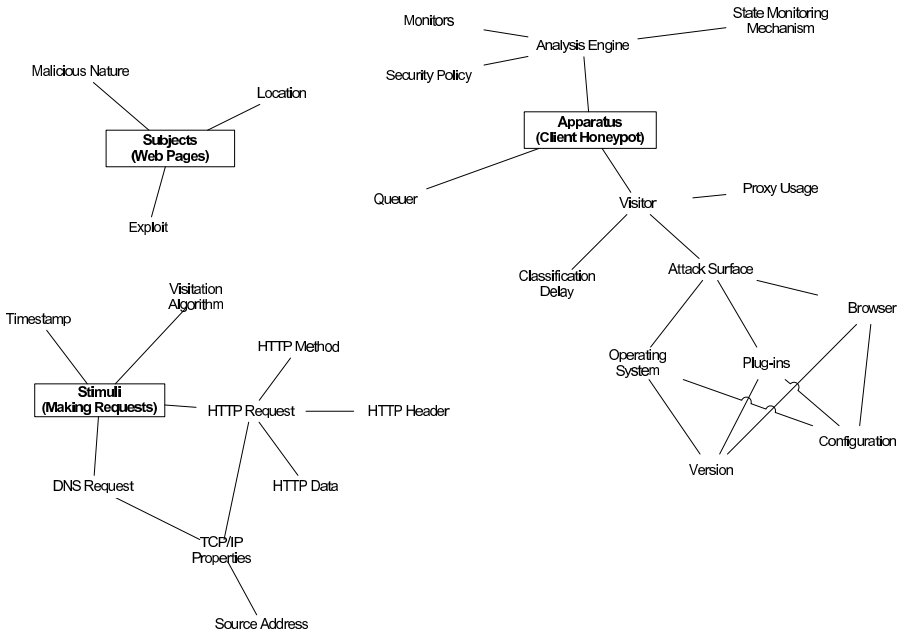


Figure 5.2: Artifacts

The Visitor has many aspects that need to be considered during the application of guide words: the classification delay, the delay that the browser waits after a page has been visited to give the attack an opportunity to trigger before the client honeypot makes a classification; the attack surface exposed by the client; and the usage of a proxy server to proxy requests. The Analysis Engine uses monitors with particular state monitoring mechanisms and security policies to classify web pages. These aspects of the apparatus need to be considered when applying the guide words.

The subjects are the second artifact. These are the web pages that the client honeypot is measuring and there are hazards that may be originating from the web pages. The web pages' location and malicious nature and the exploits used are considered when applying the guide words.

Lastly, there are the stimuli used in the experiment to obtain the measurement. In the context of measuring malicious web pages with client honeypots, the stimulus is the act of making the request to obtain the

web page to examine its malicious nature. In the process, two types of requests are made: DNS and HTTP requests, which each have specific properties, such as application and transport protocol specific properties, such as headers and header values, and source IP address. Further, DNS and HTTP requests are made at a specific time and using a specific visitation algorithm.

HAZOP was applied to the process of measuring malicious web sites on the Internet. For example, the BIASED guide word was applied to step 1 (”Create List of Potentially Malicious URL(s)”) of the flow chart from the perspective of the apparatus. The deviation that URLs selected by the Queuer introduce bias was identified. The specific consequence was a threat to external validity because bias in the URLs could threaten the generalizability of the measurement study. This is a hazard considered to be of medium severity and high likelihood. The cause for the bias may be that URLs from particular sources exhibit different malicious behavior (due to a difference in security practices, malicious intent, etc.) A possible mitigation may be to have the Queuer create a representative random sample of input URLs from the larger population to which measurements will be generalized. In the second part of this chapter, we illustrate how a lack of control in the input URL source could result in significantly different measurements. The remainder of the HAZOP is described in the HAZOP analysis work sheet in Appendix C.

Once the HAZOP was conducted, all hazards were examined with regards to severity and likelihood. All hazards of low severity and low/medium likelihood were accepted as low-risk hazards. All hazards with higher risk were addressed through mitigation. The hazards posed by the apparatus (client honeypot), the subjects (web pages), and stimuli (making the requests) as well as mitigation strategies, are summarized next.

5.1.1 Apparatus (Client Honeypot)

Several hazards were identified around the client honeypot technology used to conduct the measurement. A group of high-priority hazards was concerned with the functional aspects of correctness of the client honeypot. Questions of whether the client honeypot performs what it is designed to do were raised multiple times during the HAZOP. The mitigation strategy identified was functional testing. Functional testing can be supported when the technology is transparent and available to a larger audience, so it can be examined and tested. The open-source community does provide this level of support and is one driver of why our client honeypot Capture-HPC has been made publicly available as an open-source project. It appears that many hazards identified as part of the HAZOP are mitigated through this strategy (in 2008, for instance, Capture-HPC was downloaded 2003 times; 485 messages around installation issues, bugs, and feature requests were posted on the mailing list, and 56 bugs were filed.)

In addition to the correctness of the client honeypot, reliability was a concern. Especially in a setting in which there are several network components involved and attack code is executed, hazards that stem from low reliability emerge. Continuous monitoring and error handling appear to be mitigating these hazards and have been implemented as part of the client honeypot Capture-HPC.

Additional hazards to specific client honeypot components Queuer, Visitor, and Analysis Engine were identified. These are described in the next sections.

Queuer

The Queuer, the component of a client honeypot that consumes the URLs from the URL source and passes the URLs onto the visitor component to retrieve the corresponding web pages, appears to be particularly suscep-

tible to introducing bias into the measurement, as it is responsible for selecting potentially malicious URLs.

A sample taken from dubious sources, such as links in email spam messages and hacker web sites, can show very different characteristics in malicious web pages than in web pages sourced from search engines in a random fashion.

In addition, a Queuer could be simply selecting URLs from a URL store. An alternative strategy is to process the URLs before they are sent to the Visitor. For instance, the Queuer component could inspect the URL for specific characteristics, such as a suspicious path or query string (e.g. ”Exploit.html” or ”ms06-014”) and pass along only URLs where a match is found. The Queuer could generate URLs based on past URLs and results of the Analysis Engine. For instance, a Queuer could extract outlink URLs from web pages sent to the Analysis Engine that are deemed to be malicious.

While these are all valid strategies, they may introduce bias that threatens external validity. The mitigation strategy could be the creation of a representative random sample of input URLs from the larger population to which measurements will be generalized. Alternatively, one could simply accept the constraint and not generalize. Whatever mitigation strategy is chosen, the Queuer and its mechanism for selecting URLs need to be described in detail, so the potential hazard is appropriately disclosed.

A Queuer also controls which and when URLs are sent to the Visitor to be retrieved. Sending URLs in a fashion that is not typical human behavior (e.g., visitation of links on the page in order) could indicate to the web server that a crawler and potential client honeypot are inspecting the page and the web server could selectively choose not to send an attack, causing a false negative. Identification of crawlers and client honeypots using such a technique was described in a recent study [67]. Application of a deception model known as the deception planning loop has been proposed by the author [29]. Special steps should be taken to reduce this risk through,

for example, utilization of search engines or DNS host entries that do not exhibit crawling behavior.

Visitor

The Visitor component visits the web pages denoted by the URLs and, after the classification delay has passed, lets the Analysis Engine make an assessment of whether the web pages are malicious or benign.

The attack surface is a factor that threatens a client honeypot's ability to detect malicious web pages. The mechanism by which a client honeypot detects malicious web pages is exposing vulnerabilities and retrieving potentially malicious web pages that are capable of attacking these vulnerabilities. The state changes that result from such an attack are used as indicators that an attack has occurred. The exposed vulnerabilities need to match the attacks. As such, the operating system, the browser, and the browser plug-ins used by the client directly influence the number of vulnerabilities exposed and, therefore, the attacks used by malicious web pages that the client honeypot is capable of detecting. The configuration of these components also might influence the vulnerabilities and the ease of exploitation. A browser, for instance, could be configured with lower security settings, permitting the usage of certain ActiveX controls that otherwise would be restricted. Mitigation would utilize a mix of different versions and configurations. However, this mix quickly grows to unmanageable levels. Alternatively, one could document the configuration and not generalize beyond the particular configuration used. We have chosen to investigate malicious web pages that specifically attack Internet Explorer 6 SP2 on a Windows XP SP2 system – a configuration that is widely used and attacked.

The configuration of the operating system, browser, and plug-ins could lead to an inability to detect certain malicious web pages that do target clients at a specific location. A browser exposes a locale that, similarly to an IP address, could be used to selectively launch attacks. A composition

of plug-ins, for instance the existence of the Baofeng Storm Codec [13], a popular plug-in in the Chinese domain, could give away a client's locale, which could be used to selectively trigger attacks. This hazard could be mitigated by deploying client honeypots at different physical locations that are configured to match the locale. This greatly increases the complexity of the study. Alternatively, one could document the configuration and not generalize beyond the particular configuration used. We have chosen to investigate malicious web pages that specifically attack systems of the en-nz locale.

The configuration of a browser could also indicate to attackers whether a web crawler and potential client honeypot are accessing their site. A browser that is configured not to load images would be an example. This could preserve resources of the operator of the client honeypot, but is likely to indicate to the web server an unusual configuration, which subsequently could lead a malicious web page not launching an attack and therefore no attack would be detected by the client honeypot. Mitigation of this hazard is to configure the client honeypot identically to a system used by end users. This is a mitigation strategy we adopted.

Analysis Engine

The Analysis Engine is the component that assesses whether a web page successfully launched an attack on the client. It does so by monitoring the system for unauthorized state changes. The inner workings of the Analysis Engine influence a client honeypot's ability to detect malicious web pages. The hazards revolve around the state monitoring technique, the monitors, and the security policy.

The state monitoring technique is the way the Analysis Engine detects unauthorized state changes. There should be a high level of confidence that the data the Analysis Engine uses to make its classification is correct. It should be forensically sound. Malicious web pages could take steps to avoid detection of the state changes. If a malicious web site manipulates

the system using low-level function calls, and the state monitoring technique is monitoring high-level function calls, for instance, the technique will fail to detect them and will not truly capture the state changes on a system. Foiling the state monitoring system that runs within the client is particularly risky, because with a successful attack, the malware essentially controls the system used for monitoring.

State monitoring techniques can range from monitoring the system by determining differences between state snapshots of the system, as done by HoneyClient v1 [157], to monitoring the system in real time at the user level or kernel level. Kernel-level monitoring is preferable because a malicious web pages would have to gain control of the kernel before it could foil the state monitoring technique. Alternatively, a state monitoring technique could inspect a system from the outside. If the client runs within a virtual guest machine, the state monitoring technique could run on the host machine and use virtual machine introspection to monitor the system [45]. If the client runs on bare-metal hardware, monitoring the system state through monitoring the hardware state of memory and hard drive seems at least conceptually a possibility.

At the time of this writing, the kernel-level state monitoring technique seems to adequately mitigate the hazard of web pages evading the state monitoring technique.

Beyond the actual state monitoring technique, there are the specific monitors responsible for monitoring the state of the system. A client honeypot could monitor the file system and processes of the system. However, the hazard that malicious web pages cause state changes other than in the file system and processes of the system exists and, therefore, these web pages would go undetected. Additional monitors would reduce this risk. For instance, a client honeypot that merely monitors file and process changes would not be able to identify an attack that modifies the browser process and communicates with the attacker of the network. Additional monitors would be needed: network monitor, module (e.g., dy-

namic linked library) loading monitor, registry monitor, process threat monitor, and rootkit monitor are a few examples.

The security policy defines which events reported by the monitors will result in classification of the web page as malicious or benign. A security policy that focuses on key elements, such as the ways a malicious web page can modify the startup sequence of the operating system, will have a more narrow view than a security policy that monitors the entire system. An operator might use a narrow security policy for the sake of simplicity and ease of deployment over the more comprehensive complex security policy. Similar to the functional testing of the apparatus, the security policy needs to be tested. Being made publicly available as part of the open-source software supports this testing. The security policies used by Capture-HPC are all publicly available and community maintained.

5.1.2 Subjects (Web Pages)

The subjects, which are the web pages, also pose a variety of hazards to the experimental design of the measurement study. The primary threat is related to connectivity issues in which a web page may not be able to participate in the study because the network components, such as DNS server and/or HTTP server, are temporarily unreachable. This hazard may be addressed through retrying retrieval of the web page multiple times and logging any unsuccessful visits to the web pages. This strategy is implemented as part of Capture-HPC.

Further, a malicious web page may choose not to participate in the study. It could selectively not launch an attack as part of the study, but do so when accessed by a regular user. This may be caused by anti-forensic techniques employed by the malicious web page. The selective behavior could stem from the fact that the web page somehow identified the client honeypot. A malicious web page can identify the client honeypot primarily through the means it uses to make requests; this hazard is more closely

reviewed in the following section.

5.1.3 Stimuli (Making the Requests)

Stimuli, in our context the means of making the requests to retrieve the web page for it to participate in the measurement study, are the final area that could pose hazards to the measurement study. Since the process of making the request is part of the apparatus, hazards similar to those around functional correctness and reliability apply to making the request. Functional testing and monitoring functionality during the operation are the primary mitigation strategies for those hazards.

As mentioned in the previous section, a malicious web page may choose not to participate in the study by analyzing the way requests are made. Several characteristics of the requests may cause this hazard: location, time, deceptive nature, and history.

Location

The location from where requests are made may pose a hazard to a measurement study. The two main reasons location is of importance are the campaigns run by attackers and evasion techniques.

Attackers that operate malicious web pages do so to defraud their victims. They run campaigns that are highly sensitive to location, language, and economic trends [37, 39] as a means to increase their return on investment. As a result, the web page, exploits, and malware can be tuned according to these factors. For instance, malware found on Chinese sites might target the theft of passwords from online gamers [90, 131], whereas malware on Brazilian sites might be designed to steal bank account information [131].

For a client honeypot, this manifests itself in an inability to detect certain malicious web pages that target clients at a specific location. A client honeypot located in New Zealand accessing a malicious web page that

triggers an attack only if accessed from a client located in Germany will not be exposed to the attack and therefore will fail at identifying that malicious web page. An assessment of where a client is located is primarily made by mapping the IP address of the client to a specific location with freely available libraries, such as MaxMind Geolocation Technology [81]. Web exploitation kits, such as MPack, provide functionality to enable location-based triggering of attacks.

The geolocation-dependent triggering could easily be extended into a more fine-grained triggering mechanism as an evasion technique to avoid specific networks. Malicious web pages could create the illusion of a malicious web page in a sheep skin for entities that find and assess malicious web servers (antivirus and security companies), but could continue to exhibit malicious behavior when accessed from outside these specific networks. For the attacker, it would lead to a greatly reduced risk of detection, while at the same time increasing the likelihood of continued operation of the malicious web server, and therefore continued financial gain for the attacker.

Alternatively, the location of a client honeypot could be identified through the locale of the system. The locale consists of the language and country properties of the system and is passed along with a browser HTTP request in the form of a header value.

As a mitigation strategy, one could distribute and diversify client honeypots across network locations and locales. The goal is to emulate realistic requests as they are coming from a typical user; a user in China should have a Chinese locale, i.e., usually is located in China and accesses web pages in China. Distribution and diversification may be infeasible and may not be necessary. For a measurement study on web pages in a specific region, one may select a typical location and locale for the client honeypots conducting the measurement study, as we have done in our study on the New Zealand Internet. Since location and locale pose a hazard to the measurement study, they should be explicitly documented.

Time

Similar to location, time may be used by malicious web pages to influence their behavior. A web page with a specific location and target audience may have specific traffic patterns that are time-dependent. It may see 80% of the traffic during normal business hours. A malicious web page may, therefore, disable its malicious behavior during the non-business hours to reduce the risk of detection. This hazard may be addressed by aligning the client honeypot to these request patterns.

Deceptive Nature

If a malicious web page detects that it is part of a study, it may choose not to participate in the study. It could make this decision based on identifying that the requests were made by a client honeypot. As such, the deceptive nature of the client honeypot in making the request needs to be closely aligned with how users make requests. As mentioned above, this could be related to the time a request is made. In addition, it could be related to how the content is accessed. A visitor that requests links on a page in sequence may arouse suspicion, as might a visitor whose identifying user-agent string does not match the HTTP request headers that usually come with that user-agent string. Further, a browser that does not load images or that accesses content at high speeds may be easily identified by the malicious web page.

A particular way a malicious web page may evade detection is through time bombs. A malicious web page may simply wait a few seconds before the attack is triggered. Users usually dwell on the web page for a few seconds to read the content, whereas crawlers scour the web at high speeds and do not dwell for seconds on the page.

There are numerous ways a web page may detect a client honeypot and, to mitigate the hazard, the client honeypot needs to align itself closely to the behavior of a user who accesses the web page. In our studies, we

mitigate by utilizing links mined from search engines. The links are visited in a way that appears as if a user had typed the link in the address bar. The request is made by a real browser with no modifications. The browser displays the web page for several seconds. This makes it difficult for the malicious web site to distinguish the client honeypot from a user accessing the web page and, therefore, it will no longer have the option of opting out of the study.

History

The history of requests may pose another hazard to a study of malicious web pages. Particular malicious web pages implement a tracking functionality in which the attack is launched only once upon a target. A client honeypot requesting the identical page a second time would not lead to an attack and, therefore, the malicious web page would be missed. This is particularly problematic with visitation algorithms, such as the divide-and-conquer algorithm that need to repeatedly interact with the same web page to classify it.

There are several mitigation strategies against this hazard. One could simply choose a visitation algorithm that does not require repeated interaction (such as the sequential algorithm) or distribute repeated requests over multiple client honeypots (tracking is often done by storing the IP address of the client). Further, one could utilize a caching mechanism, such as a web proxy caching all responses.

5.1.4 Summary

Overall, HAZOP has identified many hazards to a measurement study. Existing measurement studies on malicious web pages appear not to address the threat to external and internal validity. The HAZOP technique described in this chapter provides a systematic and thorough approach to identify and address these hazards. Because hazards are plentiful, it is

recommended that the experimental design be described in much greater detail than in present studies, so the reader can assess whether threats to external and internal validity have been considered and taken into account. Currently, these aspects appear to be neglected. In the next section, we illustrate the impact of such negligence.

5.2 Impacts of Neglecting Hazards

This section presents the impacts of the hazards on the measurement of web pages that launch drive-by-download attacks using client honeypots. It is shown that the URL source can significantly impact the base rate. URL sources from different content categories (e.g., adult, forums, warez, etc.) and different top-level domains (e.g., .nz vs .com) can lead to elevated levels of web pages that launch drive-by-download attacks. Further, the impact of when the measurement is conducted is illustrated. Depending on when measurements are taken, the number of malicious web pages differs significantly.

The experiments and data presented in this section are based on two experiments we conducted in 2007 and 2008.

5.2.1 URL Source

This section shows that the URL source can significantly impact the base rate. Certain content categories can lead to elevated levels of web pages that launch drive-by-download attacks. While Moshchuk et al. also investigated how categories impact measurement [96], their study lacked a sufficient sample size, which manifests itself in unreliable numbers, and also lacked a detailed description of their apparatus, subjects, and stimuli. The work presented in this section is designed to address some of these shortcomings and confirm or dispute their observations.

Following the discussion of content categories by a presentation on

the impact of top-level domain. As with content category, some studies exist that investigate the impact of top-level domain on detection rates. McAfee's report titled "Mapping the Mal Web, Revisited" [83] shows percentages of malicious web pages for a variety of top-level domains. However, neither the apparatus, subjects, and stimuli has been disclosed as part of McAfee's report. Further, the data presented in the report lacks top-level domains of interest namely, the .nz and .au domains.

First, the impact of content categories is illustrated. Approximately 220,000 URLs were used for this study. The URLs were categorized along the content area of the web page denoted by the URL. They were sourced by issuing keywords of the specific content area to the Yahoo! search engine. The areas were:

- Adult – pages that contain adult entertainment/pornographic material
- Music – pages that contain information about popular artists and bands
- News – pages that contain current news items or news stories in sports, politics, business, technology, and entertainment
- User content – pages that contain user-generated content, such as forums and blogs
- Warez – pages that contain hacking information, including exploits, cracks, serial numbers, etc.

Approximately 220,000 URLs from approximately 100,000 hosts in these categories were sourced from the Yahoo! search engine. Table 5.1 shows the detailed breakdown for the different content areas. The URLs were grouped by content area and then inspected with our high-interaction client honeypot Capture-HPC v1.1 in the first half of May 2007.

Source	Inspected Hosts	Inspected URLs
Adult	16,375	33,999
Music	13,106	49,269
News	21,188	47,224
User Content	24,331	45,835
Warez	23,530	44,870
Total	**98,530**	**221,197**

Table 5.1: Input URLs/Hosts by Source

Using these input URLs, a total of 266 malicious URLs from 158 hosts were identified. No significant overlap of hosts or URLs existed. The percentage of malicious URLs within each source ranged from 0.0223% for music content to 0.5735% for adult content. Table 5.2 shows the breakdown of the various sources.

Source	Malicious Hosts	Malicious URLs	% Malicious URLs
Adult	102	195	0.5735
Warez	19	27	0.0602
News	15	20	0.0424
User Content	12	13	0.0284
Music	10	11	0.0223

Table 5.2: Malicious URLs/Hosts by Source

A Chi-Square test ($p < 0.01$) shows statistical significance between the adult source and any other source. The base rate of URLs is higher for adult content than any other content. Between warez, news, user content and music, no statistically significant differences were detected. Comparing these results with Moshchuk et al.'s study, some differences are observed. Moshchuk et al. also inspected adult, warez, news, and music content. However, in May 2005, they observed higher percentages for music than any other content. Five months later, they observed a higher percentage of pirated content. Neither in May nor October 2005 did adult content show elevated levels over the other categories, as shown in this

study.

As shown above, content categories impact the base rate. Next, it is demonstrated that top-level domain can have a similar effect. From the Victoria University network, 664,000 web pages from the .au, .nz, .uk, and .com domains were compared.

To compare URLs from the various domains, the URLs needed to be classified with regard to their malicious nature. The number of URLs needed to be sufficient to detect any statistically significant differences across the various domains. To achieve this, a large sample of 664,000 URLs needed to be classified. Due to resource constraints, it was not possible to classify this many URLs using a slow high-interaction client honeypot. Instead, all URLs were inspected using a hybrid system, which is described in detail in the next chapter.

The data for the comparative study was collected in January and February 2008. The URLs from the .au, .com, .nz, and .uk domains were sourced from the Yahoo! search engine [35]. Because the national language for all these domains is English, URLs could be sourced by submitting English queries to the search engine. By submitting the same queries to the search engine for each domain, it can be expected that the URLs sourced from the results page are controlled and differ only in the domain they come from. Content category bias, such as elevated percentage of adult web pages over news web pages, should be applied consistently across all four domains.

The first 1,000 URLs on the results page were used to build the list of 664,000 URLs. If less than 1,000 URLs were shown on the results page, an identical number of URLs was taken to ensure that bias from one particular query did not result in bias within the collected data set favoring one domain over the other. This also ensured that an identical number of URLs was collected across the domain.

However, due to the number of hosts in each of the domains, the URLs returned by the search engine results are hosted on a different number of

unique hosts. The number of unique host names of the input URLs per domain is shown in Figure 5.3. These numbers reflect the general notion that there are more servers in , for instance, the .com domain than the .uk domain.

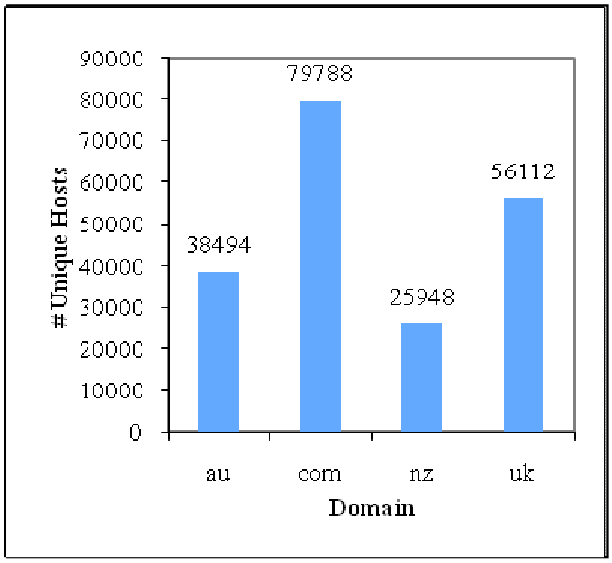


Figure 5.3: Unique Hosts of Input URLs per Domain

In total, 38 malicious URLs from 27 unique hosts were detected. Figure 5.4 shows the number of malicious URLs and hosts per domain. For example, of the 168,000 URLs per domain, 26 unique malicious URLs from 16 unique hosts were identified for the .au domain, whereas only three URLs from three unique hosts were identified for the .nz domain. The statistical Chi-Square test shows that the difference between the malicious URLs and hosts identified in the .au domain and any of the other domains is statistically very significant (URLs: $p < 0.0036$; hosts: $p < 0.0092$).

As shown, both content categories and top-level domain do influence the base rate. Measurement studies that do not control the input URLs may limit their external validity. As such, in any measurement study, it is important to disclose how the input URLs were generated and use a sample of sufficient size. Our results differ from those in Moshchuk et

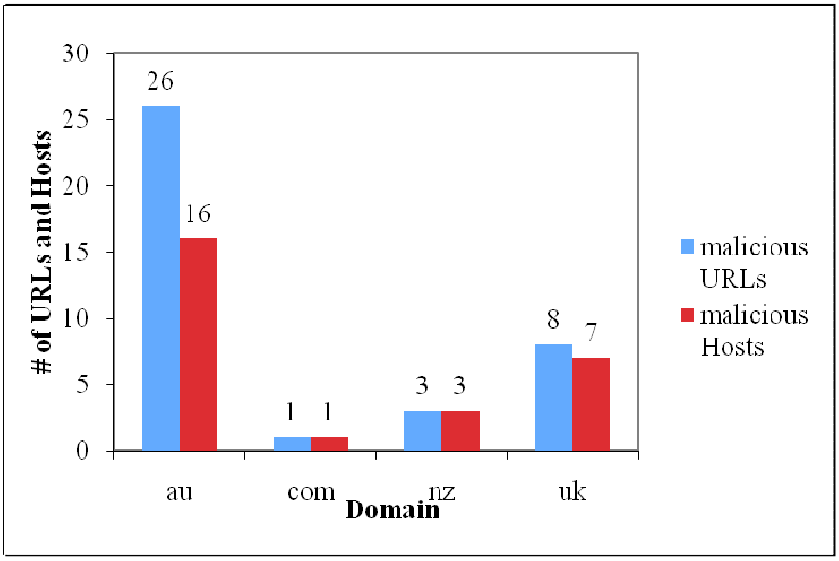


Figure 5.4: Malicious URLs and Hosts per Domain

al. The reason for these differences could not be determined, as too little information was disclosed in the published reports.

5.2.2 Time

Moshchuk at al. and our study showed different results, and the two studies were conducted at different times. Moshchuk et al. collected data in May and October 2005, whereas our data was collected in May 2007. Time could have been a factor in the differences observed. The web is highly dynamic. Just as web pages appear and disappear, the attack landscape could change. Drive-by-download attacks might shift from pages with questionable content, such as adult pages, to pages with more legitimate content, such as news pages. Time therefore is a crucial factor, as investigated in this section.

In addition, all active web servers in the .nz domain were inspected with the high-interaction client honeypot Capture-HPC v2.1 over an eight-month period. The data was collected in April 2008 and from June 2008

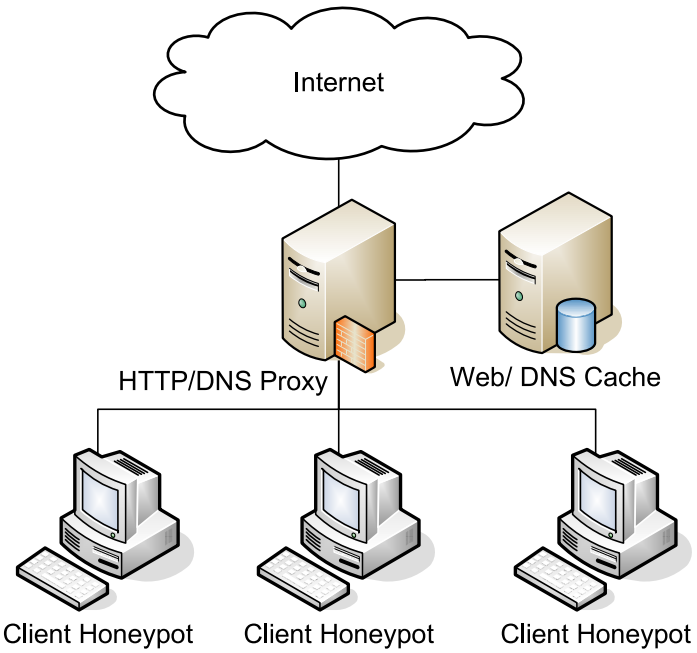


Figure 5.5: Lab Setup

to November 2008. The URLs were obtained from the .nz domain file. As URLs were inspected repeatedly, special care was taken to counter the risk of client honeypot detection, tracking, and resulting false negatives. All requests were made via a HTTP/DNS proxy server Squid v2.6 and Pdnsd 1.2.6 [168, 93], as shown in Figure 5.5. The external IP address of the system was changed with each monthly scan.

Over the eight months, a total of 291 unique malicious URLs of 247,198 input URLs, about 0.12%, were identified. Results of the monthly inspection of 247,198 URLs with the client honeypot over a period of eight month are shown in Figure 5.6. (Note that no monthly scan was conducted in May 2008.) Over the eight-month period, no increasing or decreasing trend can be detected. However, significant fluctuations between 52 (April 2008) and 97 (July 2008) malicious URLs can be observed.

This data illustrates that time seems to be an important factor and a

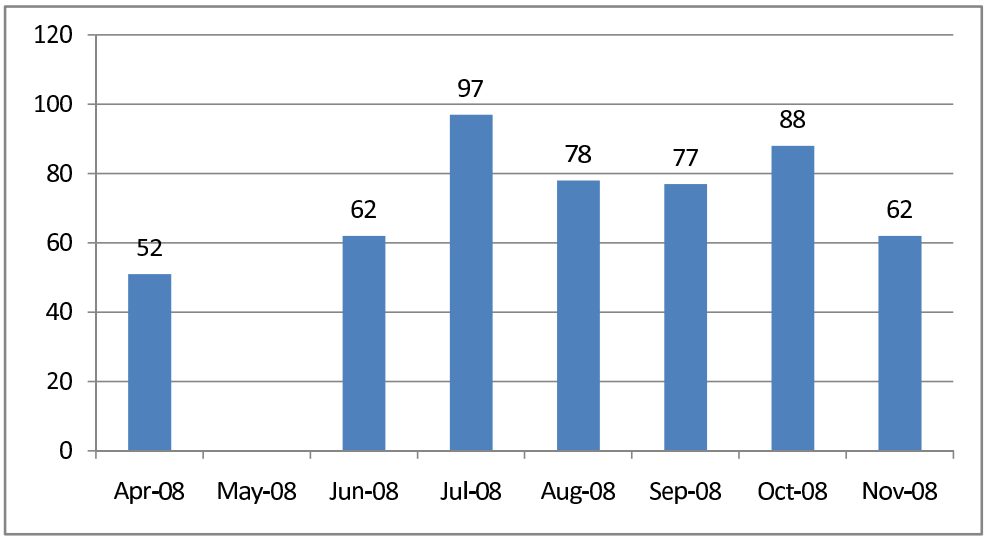


Figure 5.6: Monthly Scan Results

hazard to the validity of a measurement study. Just as web sites appear, malicious web sites might change. An increase might be related to new vulnerabilities disclosed, allowing for an infection of many malicious web sites; a decrease might be related to take-down notices of centralized exploit servers that are utilized by many thousands of web pages. As such, it is important to disclose the exact times when measurements were taken.

5.3 Summary

Weaknesses in the experimental design of the work on detection of drive-by-download attacks with client honeypots were identified in Chapter 3. A goal of our work was to stand on a strong foundation with a solid experimental design that mitigates risks to internal and external validity.

In this chapter, a methodology for identifying and mitigating risks in a systematic and thorough manner was presented: the hazard and operability (HAZOP) study. Measurement studies were used to illustrate the process of HAZOP. Uncontrolled variables were identified as major

risks. We used uncontrolled variables as examples to illustrate the impact of failure to mitigate risks appropriately. First, it was shown that the URL source can greatly impact measurements. Certain content categories, such as adult content, and top-level domains, such as the .au domain, show elevated levels of malicious web pages. If URL source is not controlled and described, the validity of a measurement study may be at risk. Second, it was shown that time can also have a major impact on measurement. Monthly measurements on the .nz domain showed significant increases and decreases in the base rate over monthly periods. As such, time is an important factor when it comes to disclosing measurement numbers. Mitigation of risks identified using the HAZOP were incorporated into our experimental designs.

Chapter 6

Low-Interaction Client Honeypots

A honeypot can be classified by its interaction level. Possible values of the interaction level are high and low. The high-interaction level denotes that the honeypot system allows for full functional interaction. An example of such a honeypot is the Honeynet [145]. A low-interaction level signifies that the functionality is limited, for example, by using emulated services. This strategy is followed by Honeyd [107].

Pouget et al. compared the interaction levels of honeypots [106] and concluded they are complementary in nature and allow for more accuracy and better utilization of resources, depending on the circumstances of deployment and goals of data collection. For example, it might be unnecessary to deploy a high-interaction honeypot on a global scale, as global data is likely to be similar; low-interaction honeypots are more suited for this situation. On the other hand, low-interaction honeypots are not suited for an in-depth investigation of attacker's actions once a honeypot has been successfully compromised. High-interaction honeypots are required to meet this goal, as they expose the full functional spectrum of a computer system for the attacker to interact with and therefore allow for collection of the desired data.

This classification can be also applied to client honeypots. While high-interaction client honeypots make use of a real dedicated vulnerable sys-

tem and usually monitor the system for unauthorized state changes to detect an attack, low-interaction client honeypots use a simulated client to interact with the potentially malicious servers. An assessment of whether an attack has occurred is done by an analysis of the server's response. Analysis techniques could apply simple signature matching, static analysis, dynamic analysis, etc.

The main differential aspect of high-interaction and low-interaction client honeypots is the Visitor component of the client honeypot component model. While a change of the Visitor component usually necessitates a change of the Analysis Engine as described, it is not a requirement for the client honeypot to be classified as a low-interaction or high-interaction client honeypot. Rather, the Visitor component is the main classifier for the interaction level.

Low-interaction client honeypots have advantages as well as disadvantages over high-interaction client honeypots. Because low-interaction client honeypots make use of a lightweight simulated client, they are usually faster than the real client of a high-interaction client honeypot. Further, deployment is usually highly simplified, because the entire client honeypot can be contained within a stand-alone application. This stands in contrast to the high-interaction client honeypot, which needs to be deployed exclusively on a dedicated system. However, at the same time, a low-interaction client honeypot can miss attacks and raise false positives.

In this chapter, we present and evaluate two classification methods that assess whether a page belongs to a malware distribution network. Malware distribution networks are responsible for the majority of malicious web pages [123, 131, 83, 164, 38, 90]. As such, any method that is able to identify whether a web page belongs to such a network will identify a majority of the malicious web pages on the Internet. These classification methods can utilize simulated clients to make a classification of a web page. As such, they can be incorporated into a low-interaction client honeypot.

First, we briefly describe malware distribution networks; that is followed by a description and evaluation of two methods that assess whether a page belongs to such a malware distribution network. The first method analyzes the network activity that is generated when a browser loads a page; the second method analyzes static characteristics of the page.

6.1 Malware Distribution Networks

A browser requests a web page through a URL. If that web page launches a drive-by-download attack, it may or may not host the exploit code directly. Often, the exploit code is fetched from other web servers, as illustrated in Figure 6.1. In such a network, several servers may be complicit in the drive-by-download attack.

- Malicious Web Page – the front-end web page the user navigates to. These pages are often legitimate web pages that have been abused by a third-party to join it to the malware distribution network.
- Redirector – the redirector may be the gateway from a malicious web page to other components of the malware distribution network. The malicious web page may contact the redirector through a server or client-side redirect, through importing content from external resources, such as iFrames or external JavaScript, etc. The redirector is the server that is responsible for pulling in the exploit from the exploit server.
- Exploit Server – the server that actually hosts the exploit that attacks the browser vulnerability.
- Malware Distribution Site – the site that hosts the malware that is pushed upon successful exploitation.

In addition to setting up complex, distributed instances of web servers to make investigation difficult, attackers also abuse DNS servers. Attack-

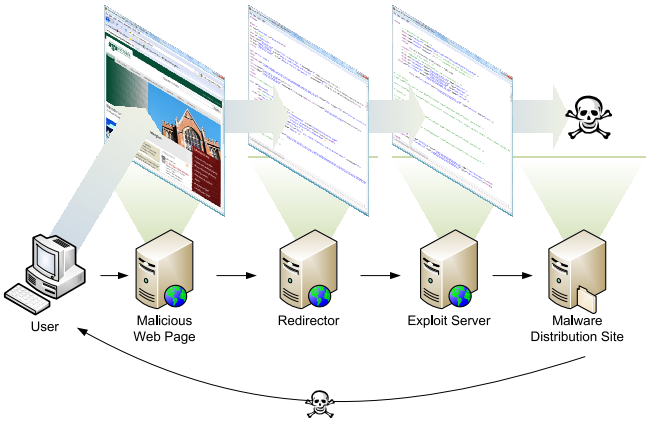


Figure 6.1: Malware Distribution Network (inspired by Figure 83 of Microsoft's SIRv6 [16])

ers make use of so-called fast-flux service networks, in which public DNS records are constantly changing [146]. For example, the host name www.foo.com might resolve to IP address 192.168.75.3 on one occasion, but to a different IP address upon the second lookup. Attackers might use this technique across international borders, e.g., in a case where a U.S.-based DNS server serves IP addresses throughout the world.

Provos and Wang mentioned that malicious web pages often belong to a malware distribution network [110, 159, 174]; white papers estimate that around 70-90% of malicious web pages belong to these networks [123, 131, 83, 164, 38, 90]. Web exploitation kits with functionality that particularly supports these network structures exist. The methods developed and presented in this chapter take advantage of the structure of malware distribution networks in determining whether a web page belongs to such a network. Because the majority of web pages appear to belong to such a network, the methods will identify the majority of the overall malicious web pages. Because they can be incorporated into a lightweight low-interaction client honeypot, they are faster than high-interaction client honeypots. Next, the two methods are presented.

6.2 Classification Method Based on Analysis of Network Traffic

Malicious web pages that belong to a malware distribution network generate abnormal network traffic by contacting several servers of the malware distribution network. Some of the structures of malware distribution networks exist to protect assets of the attacker, such as the exploit server, make the overall network failure resistant, and make the tracking and identification of the servers that compose the malware distribution network more difficult.

This is likely to be achieved through extensive redirect chains and failure-resistant network structures, such as fast-flux networks. This section introduces a novel classification method that identifies malicious web pages based on the network traffic that is generated when loading a web page. The method operates under the assumption that malicious web pages that are part of a malware distribution network will contact more servers that are involved in rendering the page from more distributed locations than will web pages that are not part of such a network.

In Section 6.2.1, the servers involved in rendering a web page are reviewed, followed by a description of the characteristics that aid in classifying malicious web pages. The methodology used in developing this new classification method and results are presented and discussed in the remainder of this section.

6.2.1 Server Relationships

When a web page is rendered, several servers are involved. These servers and their relationships to the malicious web page are described below. This information provides a foundation for understanding how malicious web pages can be identified by the classification method presented.

Two types of servers are involved in retrieving and rendering a web

page: DNS servers and web servers.

DNS Servers DNS servers are responsible for resolving the host name into routable IP addresses. For example, as shown in Figure 6.2, the DNS server resolves the host name www.mcs.vuw.ac.nz to the IP address 130.195.5.18. A client performing such a host lookup does so by sending a DNS request with the host name to a DNS server, usually located at the Internet service provider, which then sends a DNS response with the IP address of the host back to the client.

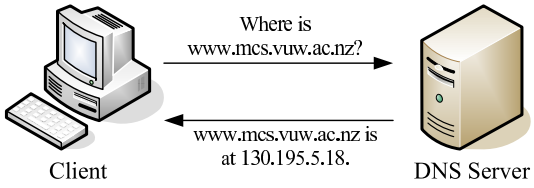


Figure 6.2: DNS Lookup

This local DNS server, however, does not necessarily know the IP address that belongs to the host name. If it does not, it goes through the steps shown in Figure 6.3. First, the local DNS server contacts one of the 13 root name servers. The root name server does not know the IP address either, but refers the local DNS server to an intermediate DNS server that might know it. (Since the example presented here deals with a host name in the New Zealand domain, it is likely that this intermediate server is the New Zealand domain name server.) If that server also does not know the IP address, it refers to another server that might, and this process iteratively continues until the responsible DNS server that does know the IP address is found. This might be a DNS server located with the hosting provider or the network itself. Once the responsible DNS server is found, it returns the requested information to the local DNS server, which in turn returns the information to the client that originally made the request.

Web Servers The other type of server involved in retrieving and rendering a web page is the web server. First is the web server that hosts the web page denoted by the URL. The web page, however, consists of

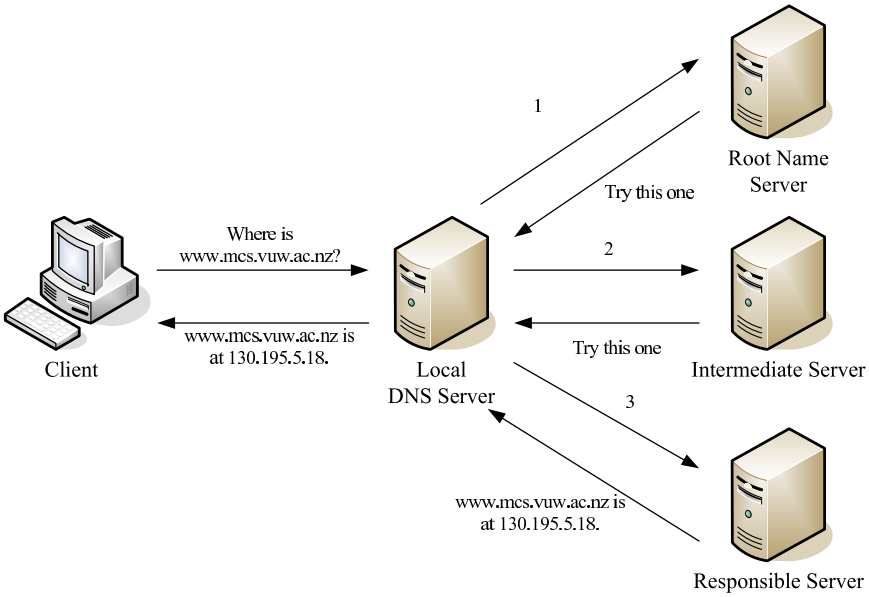


Figure 6.3: DNS Lookup by Local DNS Server

additional content beyond what is denoted by the URL, including such things as images, JavaScript, Flash content, style sheets, and embedded web pages, which might or might not be hosted by the same server that hosts the original web page. HTML, the language of web pages, supports these constructs to provide a mechanism for a reusable and modular design. For instance, it allows a web page author to include centralized elements, such as advertisements and counters, and highly dynamic content, such as news feeds.

In addition, a web page might also employ redirect directives that instruct the browser to fetch a web page located at a different URL. Server-side redirect directives are part of the HTTP protocol and are delivered as part of the server response, such as a 302 HTTP response. Client-side redirect directives are embedded in the web page, such as the JavaScript windows.location property of the document object model, which can be set to a new location. As mentioned before, these mechanisms have legitimate uses. For instance, if a web site moves from one domain to another, the redirect directives allow the author to redirect the user from the old

site to the new site, so the user is presented with up-to-date content.

The classification method presented in the next section identifies characteristics of malicious web pages related to abuse of DNS, HTTP, and HTML that would indicate more servers involved in rendering the page from more distributed locations are contacted than web pages that are not part of such a network.

6.2.2 Methodology

This section describes how the new classification method can take advantage of network traffic that is generated when loading a malicious web page. Using the high-interaction client honeypot Capture-HPC v2.1, several thousand English-language malicious web pages were identified. The network traffic generated while retrieving both malicious and benign web pages was recorded, and various attributes were extracted. The extracted attributes were fed to a J4.8 machine learning algorithm to assess their predictive nature. The methodology is described below; results are presented in the following section.

In October and November 2007, several thousand malicious web pages were inspected using the high-interaction client honeypot. After configuring the client honeypot with a clean installation of Windows XP SP2, several thousand potentially malicious web pages were inspected with the Internet Explorer 6 SP2 web browser. The list of potentially malicious web pages was generated using known bad sites, such as the MVP's hosts file [98]. As the client honeypot inspected potentially malicious web pages, the network traffic was recorded. In cases where a web page was indeed malicious, the web page was marked as such and the corresponding network traffic was saved.

Similarly, network traffic was recorded when interacting with benign web pages using an identically configured system. To collect benign web pages, English 5 N-grams (an N-gram is a selection of n words from a

string) randomly selected from the corpus of web pages linked by the DMOZ Open Directory Project [101] were issued to the Yahoo! search engine [35], and the first 50 URLs on the results page were visited with the client honeypot. The web pages were marked as benign and the corresponding network traffic was saved.

Some characteristics of web pages might be associated with the countries where the pages originate or the languages used on the web pages. For instance, Zhuge et al. observed that a large percentage of malicious web pages exist in the Chinese domain [174]. Since only benign pages were collected using English 5 N-grams, all malicious and benign pages were filtered to exclusively contain English-language web pages. The tool TextCat [153] was used to perform this filtering. TextCat is based on the text categorization algorithm presented by Cavnar and Trenkle [22].

Once the network traffic was collected and filtered, attributes that may characterize whether a web page belongs to a malware distribution network and as a result contacts more servers from more distributed locations than benign web pages do were extracted. Because some data around the DNS lookups were not contained in the network traffic, DIG, a DNS query tool, was used to obtain this information. Similarly, geo location information, in which IP addresses are mapped to specific countries, was also obtained using an additional tool, MaxMind's GeoLite Country Technology [81]. The attributes extracted are described in Table 6.1.

All the extracted attributes served as input for the machine learning algorithm. The extracted attributes were fed into the J4.8 decision tree learning algorithm implementation of the Waikato University's Weka Machine Learning Library [170]. J4.8 builds decision trees using the C4.5, revision 8, decision tree machine learning algorithm. The decision tree built by this algorithm is a predictive model that can assess whether a web page is malicious or not, which is represented by the value of the leaves. The values of the remaining attribute nodes determine the path to the child node and ultimately to the leaf node with the final classification. Decision trees, as

Attribute	Description
Number of Unique HTTP Servers	The number of unique HTTP servers. Obtained through counting the IP addresses of packets originating on ports 80, 8080, 8088, 3128, and 443.
Number of Redirects	The number of 301, 302, and 303 redirects. Obtained by inspecting the response code of web pages returned by any HTTP server.
Number of Redirects to Different Country	The number of 301, 302, and 303 redirects in which the server that issues the redirect response is located in a different country than the server to which the browser is being forwarded.
Number of Redirects to Same Country	The number of 301, 302, and 303 redirects in which the server that issues the redirect response is located in the same country as the server the browser is being forwarded to.
Number of Domain Name Extensions	The number of domain name extensions of all host names that operate a web server.
Number of Unique DNS Servers	The number of DNS servers involved in making a DNS lookup. The DIG tool is used to count the number of responsible DNS servers for each host name encountered.

Table 6.1: Extracted Dynamic Attributes

opposed to neural nets, explicitly present the acquired knowledge, which allows an expert to reason about and interpret the data.

The results of this work are presented in the next section.

6.2.3 Results

For this study, 2,623 instances of malicious web pages and 16,809 instances of benign web pages were input into the machine learning algorithm. Although a ratio of roughly 99.5% benign to 0.5% malicious web pages exists in the ”real world”, this ratio was not applied to the data input into the machine learning algorithm because that information would skew the results. In order to weight the instances of the malicious web pages more heavily, data from the malicious web pages was amplified using a ratio of approximately 1 malicious to 6.5 benign.

The data were analyzed using the J4.8 machine learning algorithm. A stratified ten-fold cross validation was performed to assess the accuracy of the acquired knowledge. This validation splits the data into ten partitions and uses each for testing and the remainder for training. The ten resulting error estimates are averaged. As shown in Table 6.2, malicious web pages would be correctly identified as malicious 74.5% of the time and

missed (i.e., false negatives) 25.5% of the time, while benign web pages would be correctly identified 97.4% of the time and incorrectly classified as malicious (i.e., false positives) 2.6% of the time.

False Negative Rate	False Positive Rate	True Positive Rate	True Negative Rate
0.255	0.026	0.745	0.974

Table 6.2: Detection Accuracy

However, if just reviewing these numbers, one might fall into the base-rate fallacy when trying to assess absolute errors [10]. Because the underlying data is not evenly distributed, the false positive rate of benign pages has a much larger impact on absolute numbers than one might initially assess. For instance, assume a distribution of benign to malicious web pages of 99.5 to 0.5 (i.e., in a set of 10,000 pages, 9,950 are benign and 50 are malicious). Using these distributions yields the absolute errors shown in Table 6.3. This table demonstrates that the impact of incorrect classification is much larger for benign web pages despite the low false positive rate (2.6%) in comparison to the false negative rate (25.5%): 2599 benign pages would be incorrectly classified as malicious, but only 13 malicious pages would be classified as benign.

99950	benign pages	0.974 True negative rate	97351	correct classifications
99950	benign pages	0.026 False positive rate	2599	incorrect classifications
50	malicious pages	0.745 True positive rate	37	correct classifications
50	malicious pages	0.255 False negative rate	13	incorrect classifications

Table 6.3: Detection Accuracy - Absolute Error Rates Example

The decision tree being generated is shown in Figure 6.4. The attribute "Number of Countries" was removed because it created a model with a slightly higher error estimate. With six attributes included in the training data set, the machine learning algorithm selected only two as relevant in the classification: Number of Domain Name Extensions and Number

of Unique DNS Servers. Using these two attributes, the decision tree can make classifications in accord with the classification accuracies presented above. For example, if (1) a URL causes web content to be retrieved from hosts having three unique domain name extensions (i.e., the number of domain name extensions is greater than two) and (2) six DNS servers are involved to resolve all host names (i.e., the number of unique DNS servers is greater than five), the web page would be classified as malicious. If, however, a URL causes web content to be retrieved from a host with only one unique domain name extension (i.e., the number of domain name extensions is two or less), the web page would be classified as benign.

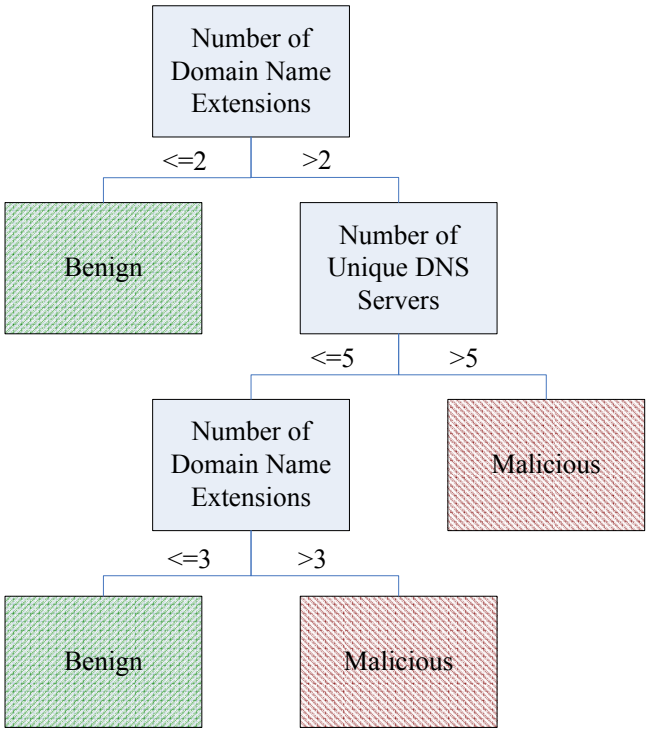


Figure 6.4: Decision Tree (confidence 25%, minimum object number of 75, and number of countries removed)

6.2.4 Discussion

The knowledge acquired by the machine learning algorithm presents a simple method for assessing whether a web page is malicious, because only the host names contacted and the DNS servers involved in the resolution of those host names need to be collected in order to assess the malicious nature of the web page.

The extracted knowledge depicted in the decision tree meets our expectations. As described above, a malicious web page that is part of a malware distribution network will contact many web servers from different locations. As implied by the constructed decision tree, this structure is rather uncommon on benign pages. While benign pages might include web components from a different domain (e.g., a New Zealand web page [domain co.nz] might include advertisements from an international corporation [domain .com]), the web page usually does not contain content from more than two different domain extensions.

A downside of the new classification method is that it could be easily evaded by attackers. Instead of using iFrames or similar methods that instruct the browser to retrieve content from a specific location, attackers could use server-side includes in which the various components are first aggregated in one web page before the web page is served to the client in its entirety. When retrieving such a web page, it might appear to be coming from only one source despite the various sources aggregated on the web server itself. This could explain the high percentage of false negatives shown by the model. However, this evasion technique places a higher burden on the attacker in terms of setting up and maintaining the attack page.

The method could be incorporated into a low-interaction client honeypot component. However, because the web page with all its components needs to be loaded in its entirety, and redirects need to be followed similar to how a real browser behaves, the performance gains of this method compared to high-interaction client honeypots will be small. Similar to

how exploits themselves could trigger only upon artificially introduced delays by the attacker (aka time bombs), redirect chains could be delayed in a similar manner. As such, the low-interaction client honeypot would have to disable such delays in its processing of the server response or accept and process the delays. The result would be performance similar to a high-interaction client honeypot. If we assume the bulk visitation algorithm is applied, a low-interaction client honeypot that would incorporate this classification method would be able to process a web page in approximately 2.95 seconds.

6.2.5 Summary

In this section, we presented a novel classification method for assessing whether a web page belongs to a malware distribution network. The classification method takes advantage of the fact that malware distribution networks are composed of many servers and usually make an effort to distribute these servers over national boundaries. It analyzes the network activity that is generated when a browser loads a page and makes an assessment of whether the page belongs to a malware distribution network by taking into account how many DNS servers were contacted and how many different top-level domains the web page components are sourced from. A 10-fold cross validation estimates the following error rates in classifying unseen web pages: a false positive rate of 2.6% and a false negative rate of 25.5%. A low-interaction client honeypot that incorporates such a method would have a service time of approximately 3 seconds.

6.3 Classification Method Based on Static Attributes on the Web Page

In this section, we present the classification method that statically analyzes the initial HTTP response denoted by the URL for characteristics that as-

sess whether the page belongs to a malware distribution network. This method, in contrast to the method described in the previous section, does not need to retrieve the web page, and the elements it contains, in its entirety nor does it need to follow redirects. Rather, the classification method is able to extract all information from the page denoted by the URL.

The classification method assesses whether the page belongs to a malware distribution network through analysis of three core elements contained on a malicious web page: the exploit itself, the delivery mechanism that joins the web page to the malware distribution network, and mechanisms to hide the exploit or the delivery mechanism from detection.

Exploit The exploit is the central part of the malicious web page and the core element that must be present for a web page to be considered malicious. The exploit is the attack code that targets a specific vulnerability of the browser, its plug-ins, or underlying operating system. It is specific to the vulnerability it is targeting and can make use of a variety of techniques. Less obvious exploits have been found in images. The most common exploits target vulnerabilities in scriptable ActiveX components. For example, a popular web exploitation kit, called IcePack, primarily targets vulnerabilities in ActiveX components with 75% of the supported exploits being related to ActiveX components.

Exploit Delivery Mechanism While the exploit is the central part of a malicious web page, the web page might not contain the exploit directly. Exploits might be ”imported” from a different server of the malware distribution network. There are two types of imports: direct includes of resources and redirects.

Direct includes of resources are a feature naturally supported by HTML. The src attribute, which exists on several HTML tags, is able to import resources from local and remote web servers. Even if a tag does not support the src attribute, scripts are able to effectively source any HTML element remotely, because scripts can arbitrarily modify an HTML page via the document object model (DOM) and so import whole HTML elements from

remote sources.

Alternatively, instead of importing an exploit, an attacker might instruct the browser to fetch a new page from a new location altogether. Redirects can be used to instruct the browser to perform this action. There are server and client-side redirects. Server-side redirects instruct the browser to fetch a page from a different location via the HTTP response code (3xx) and the location header field. Client-side redirects instruct the browser to fetch a page from a new location via HTML or JavaScript. Client-side redirects trigger after an HTML page is loaded.

Obfuscation Hiding the exploit or the exploit delivery mechanism through obfuscation is a common mechanism used by malicious web pages. Script code is provided in obfuscated form alongside a custom de-obfuscation function, which can convert the obfuscated code snippet into its clear form. Once converted, the code can be executed. Alternatively, hiding functionality that is naturally supported by HTML elements, such as the hidden style attribute of iFrames, could be utilized to hide a malicious component on the page.

All the elements described above have their legitimate purposes, but attackers also use them. As a result, a web page cannot be classified as malicious if one merely observes these elements contained in an HTTP response. A more able mechanism is needed, which is described next.

6.3.1 Methodology

In this section, characteristics of HTTP responses and how the contained HTML page can be taken advantage of to classify whether a web page belongs to a malware distribution network are presented. In October and November 2007, several thousand potentially malicious web pages were inspected using the high-interaction client honeypot Capture-HPC v2.1 configured with a clean installation of Windows XP SP2 and running the Internet Explorer 6 SP2 web browser. As the client honeypot inspected

potentially malicious web pages, the network traffic was recorded. Web pages identified as malicious were marked as such and the corresponding HTTP response was saved.

Similarly, the HTTP response was recorded when interacting with benign web pages using an identically configured system. To collect benign web pages, English 5 N-grams were randomly selected from the corpus of web pages linked by the DMOZ Open Directory Project [101]. Those N-grams were issued to the Yahoo! search engine [35], and the first 50 URLs on the results page were visited. The web pages that were not classified as malicious by the client honeypot were marked as benign and the corresponding HTTP response was saved.

Once the HTTP responses were collected, attributes of the HTTP response and embedded HTML code that aim at capturing the characteristics of a malicious web page were extracted. The attributes include characteristics that capture indications of potential exploits, exploit delivery mechanism, and obfuscation attempts. The attributes extracted are described in Table 6.4.

Category	Attributes	Description
Exploit	Plug-ins	Count of the number of applet and object tags.
	Script Tags	Count of script tags.
	XML Processing Instructions	Count of XML processing instructions. Includes special XML processing instructions, such as VML.
Exploit Delivery Mechanism	Frames	Count of frames and iFrames including information about the source.
	Redirects	Indications of redirects. Includes response code, meta-refresh tags, and JavaScript code.
	Script Tags	Count of script tags including information about the source.
Hiding	Script Obfuscation	Functions and elements that indicate script obfuscation, such as encoded string values, decoding functions, etc.
	Frames	Information about the visibility and size of iFrames.

Table 6.4: Extracted Static Attributes

All the extracted attributes were fed into the J4.8 decision tree learning algorithm implementation of the Waikato University's Weka Machine Learning Library [170]. The predictive value of the generated classifier

was evaluated on new web pages that were not used in the learning phase. First, a sample of 61,000 URLs randomly selected using the method described above was used. All URLs were classified with a low-interaction client honeypot that incorporated the classification method presented in this section as well as with the high-interaction client honeypot Capture-HPC v2.1. The false negative and positive rates were determined. In addition, the classification method was evaluated with a set of 500,000 URLs provided by HauteSecure, a leader in web-based threat protection [60]. These URLs had already been analyzed for drive-by-download attacks by HauteSecure's technology. Evaluation using these URLs reduced the risk of potential bias introduced by the high-interaction client honeypot technology and sampling method.

The sample of 61,000 URLs was also used to assess the performance gain of the presented classification method using an Amazon EC2 instance with 1.7GB of RAM, which is equivalent to a CPU capacity of a 1.0-1.2 GHz 2007 Xeon processor, on a 250Mbps connection. The duration of classifying the sample with the low-interaction client honeypot that incorporated the classification method was compared to the high-interaction client honeypot on another test machine with similar specifications (assuming a divide-and-conquer algorithm, a service time of approximately 10 seconds, and the ability to run three client honeypot instances).

6.3.2 Results

For this study, 5,678 instances of malicious and 16,006 instances of benign web pages were input into the machine learning algorithm. The generated classifier was used to classify a new sample. Of the 61,000 URLs included in the sample, 3,590 URLs were marked as malicious by the presented classification method. Inspection by a high-interaction client honeypot determined the false positive and false negative rates of the presented classification method on the sample. Seven malicious URLs were detected in

the 3,590 URLs; six malicious URLs were detected in the remaining URLs marked as benign. This amounts to a false positive rate of 5.88% and a false negative rate of 46.15% for the new classification method. The evaluation against the 500,000 URLs provided by HauteSecure resulted in similar metrics (a false positive rate of 9.7% and a false negative rate of 44.4%.)

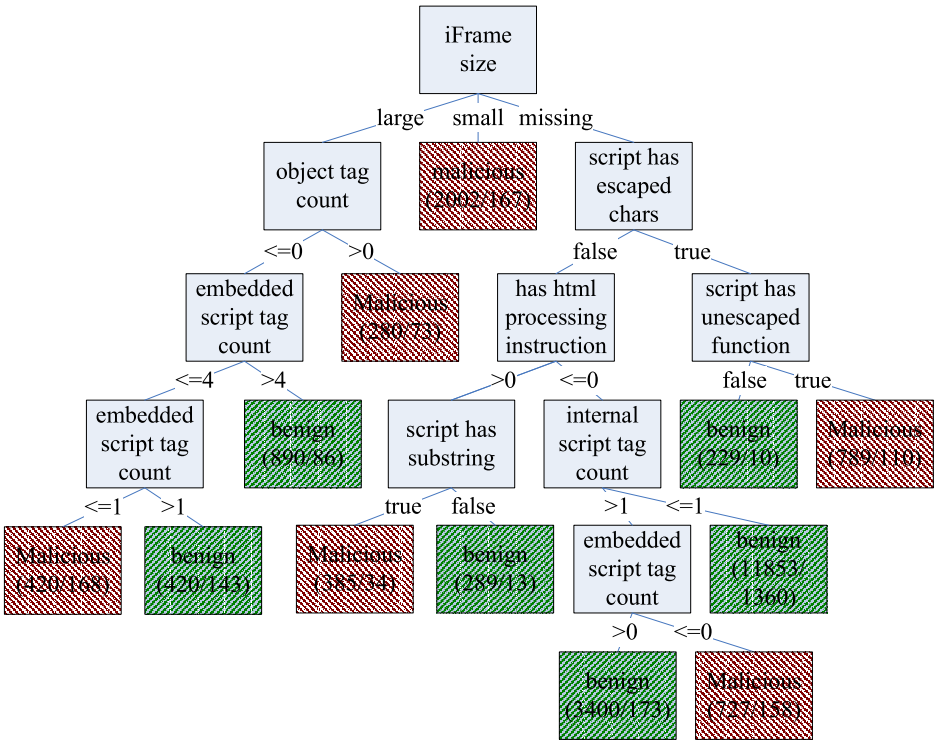


Figure 6.5: Decision Tree

The decision tree generated is shown in Figure 6.3.2. The decision tree can be used to classify unseen pages. Starting from the root node, the value of the attribute shown on the tree node is evaluated, which leads to a specific child node. Attributes are recursively evaluated until a classification node that specifies whether a page is malicious or benign is reached.

The decision tree shows that the existence of iFrames is a good classifier

on the malicious nature of a web page. Existence of a small iFrame on the web page causes a malicious classification. When the iFrame is missing, additional attributes are evaluated. The presence of escaped characters in JavaScript code and the existence of the unescape function are other good classifiers of a malicious page. These are attributes that directly link to the exploit delivery mechanism and obfuscation, which exist on pages that are part of a malware distribution network. Features that capture the existence of the exploit itself do not appear on the tree. This is likely to be the case because a majority of the web pages are part of malware distribution networks. Pages that contain exploits on the front-end web page are rare occurrences that the machine learning algorithm is likely to ignore.

Performance in classifying an HTTP response with a low-interaction client honeypot that incorporates the classification method is greatly increased over the traditional high-interaction client honeypot. The test machine was able to retrieve and classify 61,000 URLs in 49 minutes. This is equivalent to 1.79 million web pages a day (approximately a service time of 0.05 seconds per URL). In contrast, the high-interaction client honeypot classified 996 URLs in the same period of 49 minutes; this is equivalent to approximately 29,270 URLs a day (approximately a service time of 2.95 seconds per URL). The presented method is able to inspect 61 times as many URLs as high-interaction client honeypots in the same period.

6.3.3 Summary

We presented a simple yet effective classification method for assessing whether a web page belongs to a malware distribution network. The method requires assessing attributes of only the initial HTTP response. Evaluation of the classification method on a sample of 61,000 URLs resulted in a false positive rate of 5.88% and false negative rate of 46.15%. Evaluation with 500,000 URLs provided by HauteSecure resulted in simi-

lar metrics (a false positive rate of 9.7% and a false negative rate of 44.4%.) A higher false positive rate on data provided by HauteSecure may be related to the fact that HauteSecure selected their URLs from a suspicious pool of URLs. If that is the case, our method would select more URLs from that suspicious pool, which is supported by the lower false negative rate. The extracted knowledge in the form of the decision tree captures two common concepts of malware distribution networks: modular design and obfuscation. It supports our observation that exploits are usually not clearly visible on the web page denoted by the URL but are rather delivered through malware distribution networks; this is an observation supported by our in-depth analysis of malicious web pages in Appendix D. Because this classification method is based on *common* attributes of malicious pages, attackers could structure malicious pages to evade detection by this method. They merely need to make use of uncommon features. For instance, an exploit that is not imported via an iFrame and does not make use of JavaScript could evade detection. However, if attackers commonly adopt such an approach, the knowledge acquisition, if reapplied, would potentially adjust itself to capture these common attributes. The necessity and required frequency of new knowledge acquisition will be explored as part of future work.

Speed is the big advantage of the presented method. There are two reasons for the speed increase. First, the presented classification method does not require all components of a web page to be downloaded nor is support for rich functionality, such as JavaScript, required before an assessment can be made. Second, the presented classification method can be implemented as a threaded stand-alone application. This stands in contrast to the requirements of a high-interaction client honeypot, which requires several seconds to classify a page while the presented method only requires a fraction of that time for classification.

The presented classification method shows better performance over high-interaction client honeypots. At the same time, the classification method

produces false positives and misses attacks. The usefulness of the approach becomes apparent if combined in a hybrid system, which is presented in the next chapter.

6.4 Summary

In this chapter, we presented and evaluated two novel classification methods that assess whether a web page belongs to a malware distribution network. The first method takes advantage of the fact that malware distribution networks are composed of many servers and usually make an effort to distribute these servers over national boundaries. It analyzes the network activity that is generated when a browser loads a page and makes an assessment of whether the page belongs to a malware distribution network by taking into account how many DNS servers were contacted and how many different top-level domains the web page components are sourced from. A 10-fold cross validation estimates the following error rates in classifying unseen web pages: a false positive rate of 2.6% and a false negative rate of 25.5%. A low-interaction client honeypot that incorporates such a method would have a service time of approximately 2.95 seconds.

The second method analyzes static characteristics of the page to assess whether a page belongs to a malware distribution network. It captures characteristics on the page that indicate whether a link to a redirector exists and whether indicators of obfuscation exist. Obfuscation is one method that attempts to hide malicious elements on the page from signature-based approaches. An evaluation of this method on an unseen sample of 61,000 pages resulted in a false positive rate of 5.88% and a false negative rate of 46.15%. While the detection accuracy is worse than in the first method, the service time of a low-interaction client honeypot is much lower. On an Amazon EC2 instance with 1.7GB of RAM, which is equivalent to a CPU capacity of a 1.0-1.2 GHz 2007 Xeon processor, on a 250Mbps connection, the service time was about 0.05 seconds.

Both methods used machine learning – in particular decision trees – as the basis for developing a classifier that is able to assess the malicious nature of web pages based on the extracted features. Besides detection accuracy, machine learning has some additional drawbacks: staleness, evasion, and brittleness. First, the methods suffer from staleness in that they are based on the behavior and characteristics of malicious web pages at a point in time. As time passes, malicious web pages may adopt new techniques that change their behavior and characteristics. The detection accuracy of the method, as a result, may decay over time. Second, the classifiers capture uncommon behavior of web pages that are common for malicious web pages. Once this is known, malicious web pages may adjust their behavior and characteristics to blend into the crowd of all web pages. This is likely to be the case as exhibited by the fairly high false negative rate. Third, the methods are brittle. The decision tree that is used by the method can quickly lead to incorrect classification if a wrong decision is made at the top. We chose to use decision trees because they allow for expert evaluation of the extracted knowledge. According to the detection accuracy, it appears they are suitable to identify malicious web pages. However, as they become more stale, brittleness may become an issue.

Despite these shortcomings, both methods appear to be successful in determining whether a page belongs to a malware distribution network. However, due to the low base rate, the methods produce a large amount of false alerts despite low false positive rates. When combining the methods into a hybrid system using a low-interaction client honeypot with a high-interaction client honeypot, the complementary nature of the client honeypots produces a more cost-effective system. Such a hybrid client honeypot system is presented next. In the future work section, some ideas on how to address the shortcomings mentioned above will be explored.

Chapter 7

Hybrid Client Honeypot

In Chapter 4, we presented several visitation algorithms that reduce the overall cost of identifying malicious web pages with high-interaction client honeypots. The cost reduction was primarily achieved in speeding up the high-interaction client honeypot's ability to visit potentially malicious web pages. However, the overall cost remains high when trying to inspect a larger set of web pages. With a base rate of $p = 0.004$, the cost to identify one malicious web page is approximately 0.025 US dollars. If 20,000,000,000 web pages are assumed to exist today, identifying the 80,000,000 malicious web pages would cost approximately 2,030,000 US dollars. As demonstrated in Chapter 5, the web is highly dynamic, necessitating frequent scans to keep the list of malicious web sites identified current. The cost of identifying malicious web pages increases with each scan. A more cost-efficient solution is needed.

Provos and Wang mentioned that malicious web pages often belong to a malware distribution network [110, 159]; white papers estimate that around 70-90% of malicious web pages belong to these networks [123, 131, 83, 164, 38, 90]. In Chapter 6, we introduced client honeypots that are capable of taking advantage of this fact to detect these malicious web pages. These client honeypots make use of a simulated client and, as a result, are a new category of client honeypot: low-interaction client honeypots. They

can be faster and are capable of detecting malicious web pages that belong to malware distribution networks.

Unfortunately, low-interaction client honeypots do not have the favorable detection characteristics that high-interaction client honeypots do: They do produce false positives. However, when low-interaction client honeypots are combined with high-interaction client honeypots into a hybrid system, the hybrid system is able to reclaim the favorable detection characteristics. In a hybrid system, web pages are first inspected by the low-interaction client honeypot and any positive classifications are forwarded to a high-interaction client honeypot to filter out false positives, so the actual malicious web pages remain. In Section 7.1, a model is presented that illustrates how speed and detection accuracy of the low- and high-interaction client honeypot components are combined in such a hybrid system.

In Section 7.2, a specific hybrid client honeypot implementation is presented combining the low-interaction client honeypot with a low-interaction client honeypot that incorporates the static analysis method presented in Section 6.3. We use the true positive cost curve to show the positive impact of the hybrid client honeypot system on cost.

7.1 Hybrid Client Honeypot System Model

In the previous chapter, the concept of low-interaction client honeypots was introduced. Two different low-interaction client honeypot systems were presented: dynamic and static analysis. The low-interaction client honeypots stand out in their ability to make a fast classification, but at the same time, they produce false positives and miss attacks. Because of the ratio of malicious to benign web pages, even a low percentage of false positives will lead the client honeypot's classifications to consist of primarily false malicious classifications. However, at the same time, the client honeypot will correctly classify the majority of benign web pages as benign.

The overall percentage of malicious web pages of all web pages for which the low-interaction client honeypot raised an alert will be higher than the input to the low-interaction client honeypot.

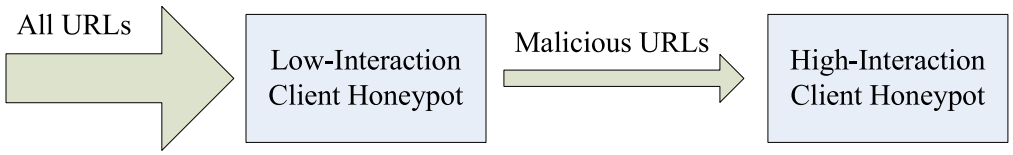


Figure 7.1: Hybrid System

The usefulness becomes apparent when combining the low- and high-interaction client honeypots into a hybrid system as shown in Figure 7.1. At the front stands a low-interaction client honeypot that initially retrieves the URLs and classifies them. Since the false positive rate is high, all URLs that have been classified as malicious are forwarded to the second part of the hybrid system, the high-interaction client honeypot. It retrieves the page once again and makes a final classification. Since the high-interaction client honeypot has a negligible false positive rate, the false positives will be filtered out.

The overall effect of the combination of low- and high-interaction client honeypots will be an increase in speed while maintaining favorable detection accuracy. In this section, we present a model that captures these characteristics. Because of the performance characteristics of low- and high-interaction client honeypots and the requirement to process a different number of URLs, the allocation needs of low- and high-interaction client honeypots in a hybrid system are different. To determine these allocation needs, the hybrid system is modeled in the form of a queue, as presented in Section 7.1.1. In Sections 7.1.2 and 7.1.3, the speed and detection accuracy of the system are modeled. In Section 7.2, the hybrid system is empirically evaluated.

7.1.1 Queues

A hybrid client honeypot system consists of low- and high-interaction client honeypots. These systems can have very different performance characteristics and processing needs. The low-interaction client honeypot is fast, but needs to process all URLs in the sample because it is at the front of the hybrid system. The high-interaction client honeypot, on the other hand, is slow, but needs to process only a fraction of URLs: the URLs that were classified as malicious by the low-interaction client honeypot. In this section, we model the hybrid system in the form of a network of multiserver queues. The model will allow us to optimally allocate resources of low- and high-interaction client honeypots. First, however, we model a homogeneous client honeypot system as a multiserver queue and then expand into the hybrid system.

Homogeneous Client Honeypot System

We model a homogeneous client honeypot system as a multiserver queue. This model serves as the basis for our model of the hybrid client honeypot system. The homogeneous client honeypot system consists exclusively of N_{Total} homogeneous client honeypot nodes, as shown in Figure 7.2. This multiserver queue ($M/M/N$) is filled with rate λ responses per time period t. The rate λ is limited by the bandwidth and byte size of responses to retrieve. Each node classifies a response in service time T, which is a function of the percentage of malicious web pages p: $T(p)$. The theoretical maximum processing capacity is:

$$\lambda_{max} = \frac{N_{Total}}{T(p)} \tag{7.1}$$

For example, a homogeneous client honeypot system consists of 30 nodes, $N_{Total} = 30$. Each node is able to service a response in 2.95 seconds, $T(p) = 2.95$. Assuming no limitations on bandwidth exist, λ_{max} is equal to 10.17 according to Equation 7.1. A homogeneous client honey-

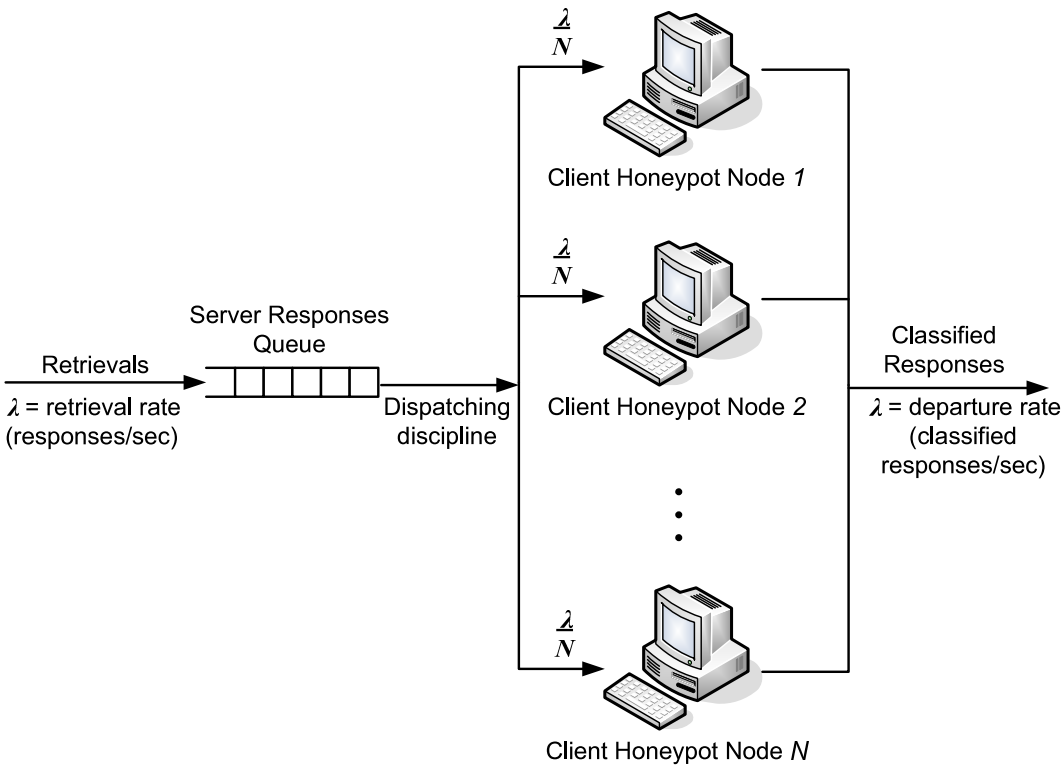


Figure 7.2: Homogeneous Client Honeypot Queue

pot system of 30 nodes and a service time of 2.95 seconds is capable of processing 10.17 web pages a second.

Hybrid Client Honeypot System

In this section, we expand the model to model the hybrid client honeypot system as a network of multiserver queues with a total of N_{Total} nodes, as shown in Figure 7.3. The hybrid system consists of two multiserver queues, homogeneous low- and high-interaction client honeypot queues that are connected in tandem. The low-interaction client honeypot queue consists of N_L nodes and each node has a service time of $T_L(p)$ per response. The high-interaction client honeypot queue consists of N_H nodes and each node has a service time of $T_H(p_{mf})$ per response. Responses are first classified by the low-interaction client honeypot queue. Depending on that classification, the traffic is partitioned and only a portion of the traffic with probability p_{mf} is forwarded to the high-interaction client honeypot queue for the final classification.

A response is able to be processed exclusively by the low-interaction client honeypot queue or by the low- and high-interaction client honeypot queues. Assuming no bottlenecks exist within the system, the theoretical maximum rate at which the system is able to classify responses is the input rate to the system λ_{Lmax}. To prevent bottlenecks in the system, a certain number of high-interaction client honeypot nodes need to exist to process the forwarded responses. The correct ratio of nodes can be determined by considering the percentage of responses forwarded to the high-interaction client honeypot as well as the service times of the individual nodes. The maximum theoretical input rate of each queue is:

$$\lambda_{Lmax} = \frac{N_L}{T_L(p)} \tag{7.2}$$

$$\lambda_{Hmax} = \frac{N_H}{T_H(p_{mf})} \tag{7.3}$$

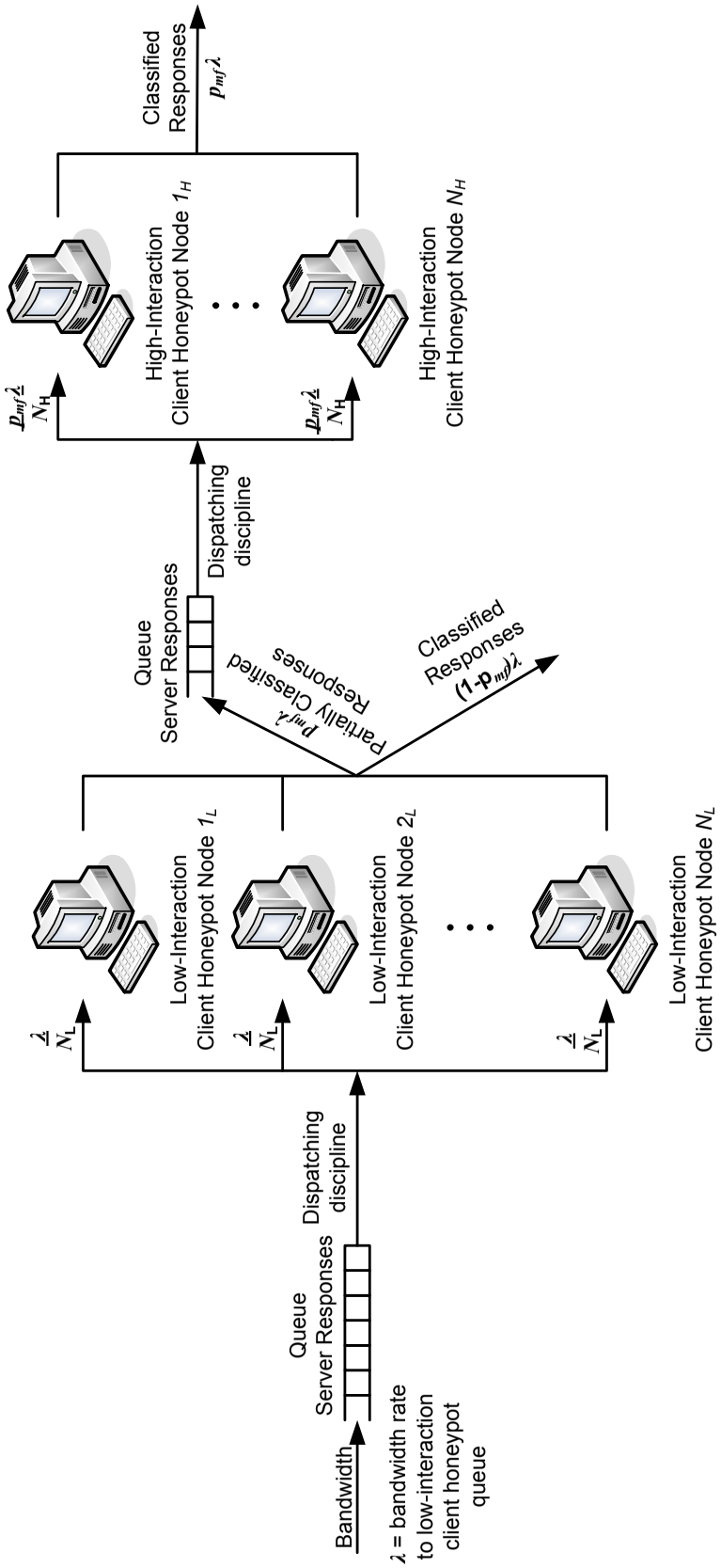


Figure 7.3: Hybrid Client Honeypot Queue

Considering only a percentage of responses is forwarded to the high-interaction client honeypot

$$\lambda_{Hmax} = p_{mf}\lambda_{Lmax} \tag{7.4}$$

follows:

$$\frac{N_H}{T_H(p_{mf})} = \frac{p_{mf}N_L}{T_L(p)} \tag{7.5}$$

Equation 7.5 represents the ratio of nodes required to prevent bottlenecks within the hybrid client honeypot system.

Presented with the number of low- and high-interaction client honeypot nodes for a hybrid client honeypot system, one can evaluate whether the number of high-interaction client honeypot nodes represents a bottleneck within the system by the following expression:

$$\frac{p_{mf}N_L}{T_L(P)} > \frac{N_H}{T_H(p_{mf})} \tag{7.6}$$

If this expression evaluates to true, the high-interaction client honeypot nodes have problems processing the responses forwarded by the low-interaction client honeypot nodes. The system would contain a bottleneck, reducing overall performance. If the expression evaluates to false, the system does not contain a bottleneck and throughput is not constrained by the composition of the queues.

Provided with the number of total nodes N_{Total} and the service times of each node, Equation 7.5 can determine the ratio of low- and high-interaction client honeypot nodes that will not lead to a bottleneck and in which $\lambda_{max} = \lambda_{Lmax}$:

$$N_L = \frac{N_{Total}T_L(p)}{p_{mf}T_H(p_{mf}) + T_L(p)} \tag{7.7}$$

$$N_H = \frac{p_{mf}N_{Total}T_H(p_{mf})}{p_{mf}T_H(p_{mf}) + T_L(p)} \tag{7.8}$$

For example, 100 nodes ($N_{Total} = 100$), a high-interaction client honeypot that is capable of servicing a response in 5.6 seconds ($T_H(p_{mf}) = 5.6$) and a low-interaction client honeypot that is capable of servicing a response in 0.05 seconds ($T_L(p) = 0.05$) are available. The low-interaction

client honeypot will forward approximately 5% of all web pages to the high-interaction client honeypot to inspect ($p_{mf} = 0.05$). The equations above allow determination of the best allocation of the nodes for low- and high-interaction client honeypots. According to Equation 7.7 and 7.8, one would need 15 low-interaction and 85 high-interaction client honeypots. With a sense of allocation needs, we turn to more closely model how speed and detection accuracy will be affected by a hybrid system.

7.1.2 Detection Speed

The detection speed of a hybrid system is greatly influenced by the performance of the individual components. However, another factor that influences speed is the number of malicious classifications coming out of the low-interaction client honeypot component, because all malicious classifications are forwarded to the high-interaction client honeypot for a second inspection. Since the high-interaction client honeypot is quite slow, this will have a large impact on the hybrid system.

$$N_M = p * N \tag{7.9}$$

$$N_B = N - N_M \tag{7.10}$$

$$alerts_L = FP_L N_B + TP_L N_M \tag{7.11}$$

$$p_{mf} = \frac{alerts_L}{N} \tag{7.12}$$

The number of malicious classifications of the low-interaction client honeypot can be calculated as follows: If we know p , the percentage of malicious pages in the set that is being inspected N, then the number of malicious pages N_M and benign pages N_B can be determined (Equations 7.9 and 7.10). Taking into account the false positive rate FP_L and true

positive rate TP_L of our low-interaction client honeypot, we can calculate the number of malicious classifications with Equation 7.11. For example, assume the $p = 0.4\%$ and a true positive rate of 50% and false positive rate of 5%. If we inspect 10,000 URLs with the low-interaction client honeypot, 680 malicious classifications are reported. Expressed in percentage p_{mf} according to Equation 7.12, 6.80% of URLs raise an alert.

$$T_{Hy}(p) = T_L(p) + p_{mf}T_H(p_{mf}) \tag{7.13}$$

The average service time of the hybrid system can then be determined by adding the service time of the low-interaction client honeypot to the service time of the high-interaction client honeypot times the percentage of URLs for which an alert will be raised. According to Equation 7.13, the average service time of the hybrid system T_{Hy} for the current example ($T_L = 0.05$ seconds, $p_{mf} = 0.068$, $T_H = 5.6$ seconds) is 0.43 seconds.

$$TTotal_{Hy}(p) = \frac{T_L(p)N + p_{mf}NT_H(p_{mf})}{N_{Total}} \tag{7.14}$$

The actual time it takes to inspect the sample of N web pages is dependent on the number of URLs in the sample and the number of nodes of the hybrid client honeypot system, as shown in Equation 7.14. This is the lower bound of the processing time, assuming all the nodes are fully utilized during the processing time. Assuming one node is available in which the hybrid system URLs are first inspected by the low-interaction client honeypot and then all URLs for which alerts have been raised are inspected by the high-interaction client honeypot system, the total time to inspect the URLs is approximately 1 hour and 12 minutes ($T_L = 0.05$ seconds, $N = 10,000$, $p_{mf} = 0.068$, $T_H = 5.6$ seconds and $N_{Total} = 1$.) This stands in contrast to the 15 hours and 33 minutes it would take to inspect with a single high-interaction client honeypot ($N = 10,000$, $T_H = 5.6$ seconds and $N_{Total} = 1$).

But as illustrated in Section 7.2, speed is not the only factor that makes for a better client honeypot. Detection accuracy is the other major factor,

Scenario Name	Classification by Low-Interaction Client Honeypot Node	Classification by High-Interaction Client Honeypot Node
Scenario 1	Not malicious	Not malicious
Scenario 2	Not malicious	Malicious
Scenario 3	Malicious	Not malicious
Scenario 4	Malicious	Malicious

Table 7.1: Classification Scenarios

as modeled next.

7.1.3 Detection Accuracy

Detection accuracy is described as the overall false positive and false negative rates of the system. The goal of the hybrid client honeypot system is to combine the detection accuracy of the individual components into an overall favorable detection accuracy. A similar model in the context of anomaly and misuse IDS has been presented by Tombini et al. [151]. The strength of the negligible false positive rate of the high-interaction client honeypot nodes within the hybrid system should be emphasized. In addition, while the false negative rate of the hybrid system is likely to surpass the false negative rate of the individual components, it remains low, so more malicious web pages can be identified overall compared to a high-interaction client honeypot system.

We first present a simplified model and continue to refine the model throughout this section, with the final model presented at the end of this section. The simplified model consists of a hybrid client honeypot system that is composed of low- and high-interaction client honeypot nodes that both classify responses. Classification by each node leads to four scenarios, as shown in Table 7.1. The nodes agree in their classifications in Scenarios 1 and 4, but disagree in Scenarios 2 and 3.

Placing the classification in the context of a malicious response allows

Malicious Response		
Scenario Name	**Classification by Low-Interaction Client Honeypot Node**	**Classification by High-Interaction Client Honeypot Node**
MR Scenario 1	False Negative	False Negative
MR Scenario 2	False Negative	True Positive
MR Scenario 3	True Positive	False Negative
MR Scenario 4	True Positive	True Positive

Table 7.2: Malicious Response Scenarios

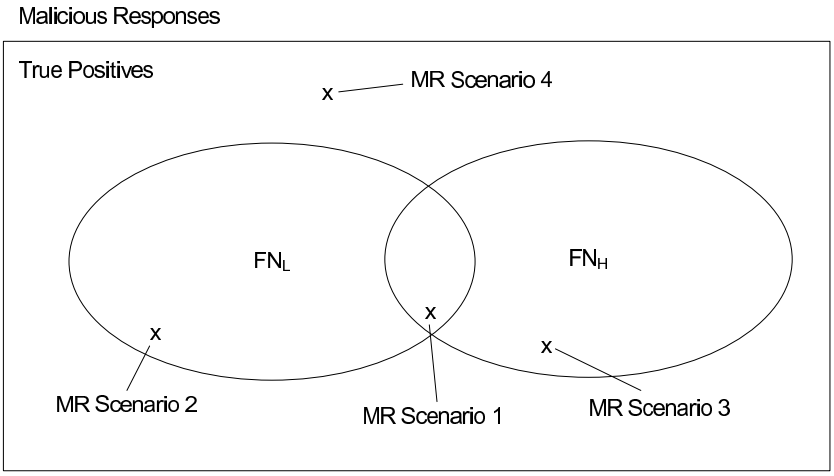


Figure 7.4: Venn Diagram - Malicious Response Scenarios

Benign Response		
Scenario Name	**Classification by Low-Interaction Client Honeypot Node**	**Classification by High-Interaction Client Honeypot Node**
BR Scenario 1	True Negative	True Negative
BR Scenario 2	True Negative	False Positive
BR Scenario 3	False Positive	True Negative
BR Scenario 4	False Positive	False Positive

Table 7.3: Benign Response Scenarios

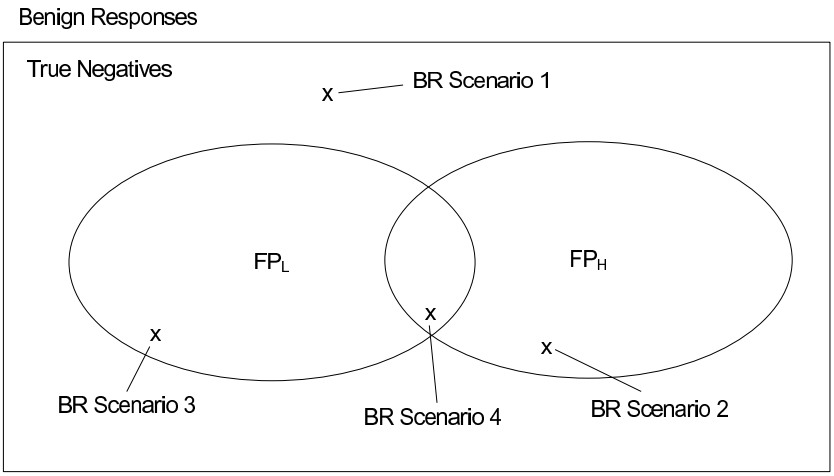


Figure 7.5: Venn Diagram - Benign Response Scenarios

us to assess the correctness of the classification, also known as false negatives, as shown in Table 7.2. Figure 7.4 graphically illustrates these classifications as a Venn diagram. All malicious servers are represented by the entire space. The set on the left represents the false negatives of the low-interaction client honeypot node (FN_L) and the set on the right represents the false negatives of the high-interaction client honeypot node (FN_H). The intersection FN_L, FN_H represents the set where both nodes raise false negatives, whereas the complement of the two sets represents the set of true positives. (Table 7.3 and Figure 7.5 show the corresponding scenarios for benign responses.) These tables and figures, however, do not communicate the false negative or false positive rates of the hybrid client honeypot system. We discuss this next.

To determine the false positive and false negative rates of the hybrid client honeypot system, there needs to be an agreement on how to treat conflicting classifications. Because the high-interaction client honeypot has a negligible false positive rate, more trust is put in the classification made by the high-interaction client honeypot. As such, only if both client honeypots raise an alert, an alert is accepted. Tables 7.4 and 7.5 show the outcome of these classifications on the overall classification of the hybrid

Malicious Response	
Scenario Name	**$FN_L \cup FN_H$**
MR Scenario 1	False Negative
MR Scenario 2	False Negative
MR Scenario 3	False Negative
MR Scenario 4	True Positive

Table 7.4: Basic Hybrid Client Honeypot Malicious Response Classification

Benign Response	
Scenario Name	**$FP_L \cap FP_H$**
BR Scenario 1	True Negative
BR Scenario 2	True Negative
BR Scenario 3	True Negative
BR Scenario 4	False Positive

Table 7.5: Basic Hybrid Client Honeypot Benign Response Classification

client honeypot system.

Figure 7.6 shows the effect on the overall false negative and false positive rates of the hybrid client honeypot system. The shaded areas in these diagrams represent the false negative and false positive rates of the overall hybrid client honeypot system. It results in a low false positive rate, but a high false negative rate, in that the false negative rate of the low- and high-interaction systems are combined. The overall false positive rate and false negative rate are described by Equations 7.15 and 7.16.

$$FN_{Hy} = FN_L \cup FN_H = FN_L + FN_H - (FN_L, FN_H) \tag{7.15}$$

$$FP_{Hy} = FP_L \cap FP_H = FP_L, FP_H \tag{7.16}$$

Now that the basic overall false positive and false negative rates of our hybrid client honeypot system are determined, we proceed to refine the model. As mentioned above, the model holds when all responses are classified. Such an approach, however, would diminish the performance gains

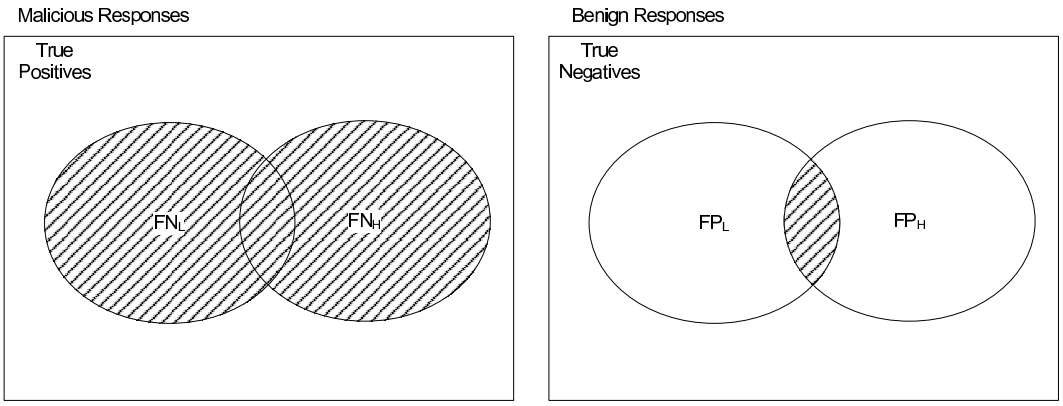


Figure 7.6: Venn Diagrams - Basic Hybrid Classification

Malicious Response			
Scenario Name	**Classification by Low-Interaction Client Honeypot Node**	**Classification by High-Interaction Client Honeypot Node**	**Classification by Hybrid Client Honeypot System**
MR Scenario 1	False Negative	n/a	False Negative
MR Scenario 2	False Negative	n/a	False Negative
MR Scenario 3	True Positive	False Negative	False Negative
MR Scenario 4	True Positive	True Positive	True Positive

Table 7.6: Hybrid Client Honeypot Malicious Response Classification - Single Classification Input for Scenarios 1 and 2

Benign Response			
Scenario Name	**Classification by Low-Interaction Client Honeypot Node**	**Classification by High-Interaction Client Honeypot Node**	**Classification by Hybrid Client Honeypot System**
BR Scenario 1	True Negative	n/a	True Negative
BR Scenario 2	True Negative	n/a	True Negative
BR Scenario 3	False Positive	True Negative	True Negative
BR Scenario 4	False Positive	False Positive	False Positive

Table 7.7: Hybrid Client Honeypot Benign Response Classification - Single Classification Input for Scenarios 1 and 2

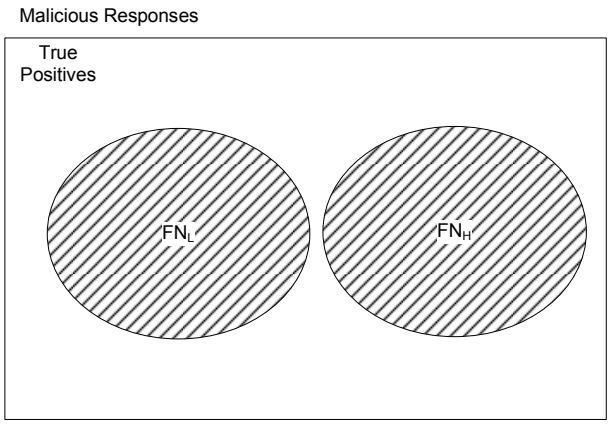


Figure 7.7: Venn Diagrams - Final Hybrid Classification – False Negatives

that are the main driver for introducing the hybrid client honeypot system. As part of the model refinement, we halt the evaluation of responses for which no alert was raised by the initial low-interaction client honeypot node. The resulting overall false negative and false positive rates will be not affected by this adjustment, as illustrated in Tables 7.6 and 7.7 and Figure 7.6. However, because the high-interaction client honeypot uses the output of the low-interaction client honeypot as input, there will be no intersection between the FN_L and FN_H as shown in Figure 7.7. Therefore, Equation 7.15 can further be simplified:

$$FN_{Hy} = FN_L \cup FN_H = FN_L + FN_H \tag{7.17}$$

In the hybrid client honeypot system, only a fraction of responses need to be evaluated by both client honeypot nodes. The percentage of responses to be forwarded would be the number of responses for which the low-interaction client honeypot node raises an alert, p_{mf}:

$$P_{mf} = TP_L * p + FP_L * (1 - p) \tag{7.18}$$

Note that p_{mf} equals p_{mf} of our performance model.

In this section, the detection accuracy of the hybrid system was presented. The overall false negative and false positive rates are given by

Equations 7.15 and 7.16. The false positive rate will be negligible for the hybrid system, similar to how it is negligible for the high-interaction client honeypot. However, combining the false negative rates of the low- and high-interaction client honeypot has increased the false negative rate overall.

7.2 Hybrid Client Honeypot System Evaluation

In the previous section, we presented a hybrid client honeypot system model. It combines a low- and high-interaction systems into a hybrid client honeypot system that can exhibit favorable detection characteristics. When combined with the fast static analysis method presented in Chapter 6, Section 6.3, the hybrid system model predicts that the system is capable of identifying more malicious web pages than a corresponding high-interaction client honeypot system with identical resources and time. In this section, we closely examine an actual hybrid client honeypot system that combines the static analysis method and the high-interaction client honeypot Capture-HPC. First, we inspect a set of web pages with a hybrid and high-interaction client honeypot system, respectively. This provides initial support of the model presented in the previous section. In the second part of this section, we introduce a hybrid client honeypot simulator that allows us to vary the dependent variables of speed, detection accuracy, and base rate of the evaluation. The simulator allows us to generate a true cost positive curve to investigate the behavior in different operating environments and configurations.

7.2.1 Evaluation of Hybrid Client Honeypot System with TPCC

When evaluating the hybrid client honeypot with the true positive cost curve, the factors of detection speed and accuracy are being evaluated.

The hybrid client honeypot system incorporates a favorable speed characteristics; however, at the same time, the number of malicious web pages in a sample is decreased. As long as the hybrid system can identify more malicious web pages overall than a high-interaction client honeypot given identical resources, the true positive cost curve will show a favorable evaluation.

The number of malicious web pages identified by a hybrid client honeypot system for a given time frame needs to exceed the number of malicious web pages identified by the high-interaction client honeypot system in an identical time frame, as shown by Equation 7.19.

$$p\lambda_{Hy}(1 - FN_{Hy}) > p\lambda_H(1 - FN_H) \tag{7.19}$$

Equation 7.20 shows the identical expression in terms of service time:

$$p\frac{1}{T_{Hy}(p)}(1 - FN_{Hy}) > p\frac{1}{T_H(p)}(1 - FN_H) \tag{7.20}$$

Note that we use $T_H(p)$ and not $T_H(p_{mf}$ because in this comparison, the high-interaction client honeypot system is exposed to a sample with the original base rate.

If those conditions are met, the hybrid client honeypot will identify more malicious web pages given identical resources than a high-interaction client honeypot system.

This equation allows one to determine whether a low-interaction system incorporated into a hybrid client honeypot system can be useful. In Chapter 6 we introduced two different approaches that could be incorporated into a low-interaction client honeypot: a method based on analyzing network traffic and a method based on analyzing static elements on the page. The speed and detection accuracy varied greatly and the usefulness of a low-interaction client honeypot was unclear. In the next few paragraphs, we illustrate the effect of low-interaction client honeypots on a hypothetical hybrid client honeypot system with the model presented in the previous section.

For the purpose of this illustration, we assume a percentage of malicious web pages $p = 0.001$. Further, we assume that the high-interaction client honeypot has a service time of approximately $T_H(p) = 2.95$ seconds and a false positive and false negative rates of $FP_H = 0$ and $FN_H = 0$, respectively. The right portion of Equation 7.20 therefore evaluates to 0.000339. For a low-interaction client honeypot to have a positive effect, the left portion of Equation 7.20 needs to evaluate to a higher value.

First, we turn to a low-interaction client honeypot that incorporates the method that analyzes network traffic. A 10-fold cross validation estimates a false positive rate of $FP_L = 0.0260$ and a false negative rate of $FN_L = 0.2550$. As no system that implements this classification method was built, the service time was not recorded. However, for the dynamic behavior classification method to be successful, a web page needs to be retrieved in its entirety. This can be accomplished with a regular browser. As a result, we assume a service time similar to that of a high-interaction client honeypot: approximately $T_L = 2.95$ seconds. The resulting service time of the hybrid system would be approximately $T_{Hy}(p) = 3.02$ ($p_{mf} = 0.0267$) with a false negative rate of $FN_{Hy} = 0.255$. The left portion of Equation 7.20 evaluates to 0.000246, which is lower than the value for the high-interaction client honeypot system. As such, low-interaction client honeypots that incorporate the method that analyzes network traffic seem unsuitable to be combined into a hybrid client honeypot system.

Second, we look at a low-interaction client honeypot that incorporates the method that analyzes static elements on a page. Its detection accuracy is not as favorable as the method described in the previous paragraph, with false positive and false negatives rate of $FP_L = 0.0588$ and $FN_L = 0.4615$. However, at the same time, an implementation showed a service time of $T_L(p) = 0.05$ seconds. The resulting service time of the hybrid system would be approximately $T_{Hy}(p) = 1.79$ ($p_{mf} = 0.5385$) with a false negative rate of $FN_{Hy} = 0.4615$. The left portion of Equation 7.20 evaluates to 0.002395, which is larger than 0.000339. As such, the static analysis

method incorporated into a low-interaction client honeypot as part of a hybrid system appears to bring benefits.

The values of Equation 7.20 express the number of malicious web pages detected for a given time period and correspond directly to the values of the true positive cost curve, which we investigate in greater detail toward the end of this chapter.

7.2.2 Initial Evaluation with a Hybrid Client Honeypot System

In this section, we inspect a set of web pages with an actual hybrid client honeypot system. First, we present the implementation of the hybrid system, then evaluate the hybrid client honeypot system with a set of randomly selected URLs. The performance of the hybrid system will be compared against the model presented in the previous section. In addition, the hybrid system's performance will be compared to a high-interaction client honeypot system.

Hybrid Client Honeypot System Implementation

Our implementation of the hybrid client honeypot consists of a low-interaction client honeypot that analyzes static elements on a page to make a classification about a page, as described in Section 6.3, and high-interaction client honeypot Capture-HPC. URLs are first inspected by the low-interaction client honeypot; if deemed malicious, are forwarded to high-interaction client honeypot Capture-HPC for a final classification.

The low-interaction client honeypot is a simple emulated browser written in Java. It uses the Apache HTTPClient [41] with a spoofed Internet Explorer user-agent header to retrieve web pages at high speed. The low-interaction client honeypot retrieves only the page denoted by the URL, so no embedded images, JavaScript, iFrames, etc. are retrieved. The low-interaction client honeypot also does not follow redirects. Once retrieved,

the HTTP response is parsed using HTMLParser [103] and the features are extracted and forwarded to the Weka Machine Learning Library [170] for classification using the tree that was derived from our classified training set in Section 6.3.

Capture-HPC is the high-interaction client honeypot introduced in previous chapters. It was configured with a stock installation of Windows XP SP2 and Internet Explorer 6.0 and with a classification delay of 10 seconds.

The low-interaction client honeypot was deployed on an Amazon EC2 instance with 1.7GB of RAM, which is equivalent to a CPU capacity of a 1.0-1.2 GHz 2007 Xeon processor, on a 250Mbps connection. The low-interaction client honeypot exhibited a service time of approximately 0.05 seconds. On hardware with similar specifications, we deployed three instances of our high-interaction client honeypot Capture-HPC. The three high-interaction client honeypots were capable of inspecting approximately 29,270 URLs a day, which is equivalent to a service time of 2.95 seconds per URL.

The hybrid client honeypot system was presented with 61,000 unclassified URLs. These URLs were randomly selected by issuing English 5 N-grams to the Yahoo! search engine. Each English 5 N-gram was randomly selected from the corpus of web pages linked by the DMOZ Open Directory Project [101]. The first 50 URLs on the results page were used for an initial evaluation of the hybrid client honeypot system.

First, the 61,000 URLs were inspected with the low-interaction client honeypot. Each URL classified as malicious was also inspected by the high-interaction client honeypot. To compare the speed and detection accuracy of the hybrid system, all URLs were also inspected by the high-interaction client honeypot system. The classifications and duration to inspect the URLs were recorded.

Results

Of the 61,000 URLs included in the sample, 3,590 URLs were marked as malicious by the low-interaction client honeypot. Inspection of those 3,590 URLs by a high-interaction client honeypot identified seven malicious URLs; the high-interaction client honeypot identified 13 malicious URLs in the entire sample of 61,000 URLs. If the false positive and negative rates of the high-interaction client honeypot are assumed to be zero, the hybrid client honeypot system produced zero false positives, but missed approximately 46.15% of malicious web pages. The low-interaction client honeypot produced 3,583 false positives or 5.88% ($N_B = 61,000 - 13 = 60987$, $FP_L = \frac{3583}{60987}$).

A comparison to the model presented in the previous section shows similar numbers. We determined that $p = \frac{13}{61000} = 0.000213$. The model predicts the hybrid client honeypot's false positive rate to be equal to the false positive rate of the high-interaction client honeypot component, in our case zero, and the false negative rate to be equal to the union of the false negative rate of the low- and high-interaction client honeypot components. In our case, this was 46.15%. As a result, the model predicts that the hybrid system will identify approximately six malicious URLs, which is what occurred.

The speed of the hybrid system also matches the predictions of the model: The low-interaction client honeypot inspected 61,000 URLs in 49 minutes. The 3,590 URLs for which the low-interaction client honeypot raised an alert were inspected in 2 hours and 56 minutes. The hybrid client honeypot therefore spent 3 hours and 45 minutes to inspect 61,000 URLs. In contrast, the high-interaction client honeypot system spent 49 hours and 59 minutes to inspect all 61,000 URLs.

The number of malicious URLs needs to be taken into account to determine the cost of identifying each malicious URL using a hybrid or high-interaction client honeypot system. If a cost of 0.125 US dollars per hour is assumed, the hybrid client honeypot cost 0.368 US dollars to identify

seven malicious URLs or 0.053 US dollars per URL. This stands in contrast to a cost of 0.480 US dollars per malicious URL for the high-interaction client honeypot system, which identified 13 URLs in approximately 50 hours (total cost of 6.24 US dollars).

Conclusion

Our initial experiment has revealed that under certain conditions, the hybrid client honeypot system appears to have favorable characteristics in the detection of malicious web sites. At $p = 0.00213$, the hybrid client honeypot system was able to identify malicious URLs at a cost of 0.053 US dollars; this is approximately one-tenth the cost of identifying malicious URLs with a high-interaction client honeypot system.

The model presented in the previous section appears to predict performance well. The data collected with a hybrid client honeypot implementation consisting of the low-interaction client honeypot and Capture-HPC matches the predictions of the model.

However, to more comprehensively evaluate the hybrid client honeypot system, dependent variables need to be controlled to better understand the conditions in which the hybrid client honeypot system exhibits favorable characteristics over the high-interaction client honeypot system. Because the dependent variables, such as detection accuracy, are difficult to control, we chose to utilize a simulator to explore the hybrid client honeypot system more comprehensively. The simulator and the evaluation of the hybrid client honeypot system using the true positive cost curve are presented next.

7.2.3 Evaluation with a Simulator

As mentioned in the previous section, the effectiveness of a hybrid system is dependent on the speed and detection accuracy of the components as well as the base rate of the operating environment. In this section, we

explore the effects of these dependent variables on the hybrid client honeypot system using a simple simulator.

While the dependent variables can be varied using the model introduced in the previous section, the simulator provides us with another means of validation of the model. Similar to a real hybrid client honeypot system, we can use it to validate against the model. However, with the simulator, we are also getting greater flexibility. The simulator is a rapid prototyping platform, which would allow us to quickly change aspects of the system that may not be exposed by the model. For instance, we could use it to modify the client honeypot node composition to introduce bottlenecks. Further, we could assess how the performance and detection accuracy of the system changes if we were to chain multiple low-interaction client honeypots together. The simulator provides us with this flexibility.

Simulator

The simulator is a simple Java program that models a hybrid client honeypot system, which allows us to vary the dependent variables of speed and detection accuracy of the components as well as the base rate of the operating environment. The Java program simulates the hybrid system with web pages (with a base rate) and low- and high-interaction client honeypot components (with speed and detection accuracy) capable of classifying the web pages. As the simulator runs, the malicious classifications and total time spent making these classifications are recorded and can be retrieved. Because the service time is merely tallied to the total time spent, and not to time actually passed, the simulator is capable of determining classifications and total time spent instantaneously, which allows us to explore a wide range of hybrid client honeypot systems in varying conditions. In addition of being able to simulate a hybrid system, the simulator is also capable of simulating a system that exclusively consists of high-interaction client honeypots.

The class diagram of the simulator is shown in Figure 7.8. The simu-

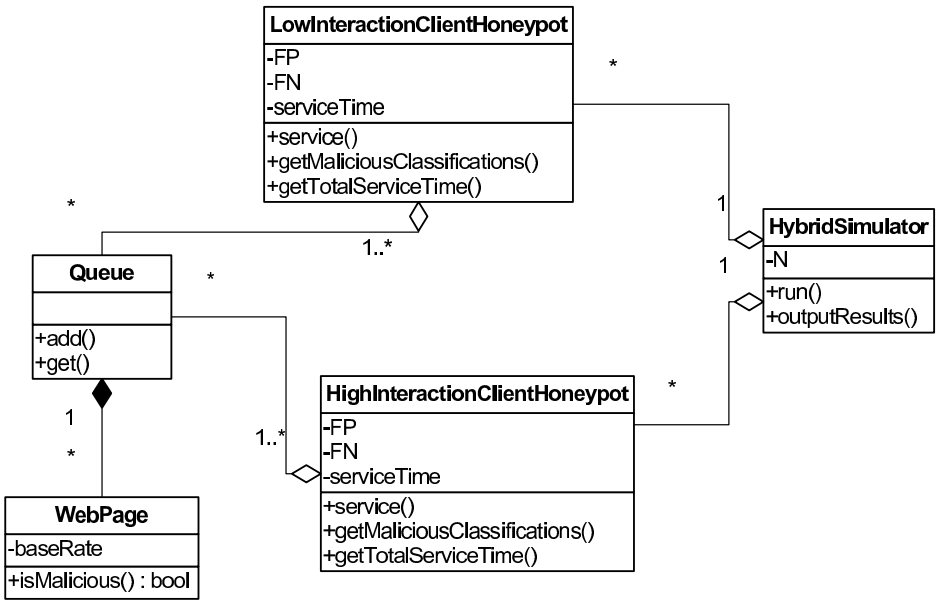


Figure 7.8: Hybrid Client Honeypot Simulator Class Diagram

lator consists of N low- and high-interaction client honeypots with a detection accuracy expressed in false negative and false positive rates and service time. Each client honeypot has access to a queue that consists of multiple web pages that, based on the base rate p, are either malicious or benign, which is randomly assigned to each web page as it is generated. The low-interaction client honeypot accesses a queue that consists of all web pages to be classified, whereas the high-interaction client honeypot accesses a queue that consists only of web pages that the low-interaction client honeypot classified as malicious. Classification occurs through application of the false positive and false negative rates on the malicious or benign web page. While a classification is made, the total time is incremented by the service time of the client honeypot. Once all web pages are processed by the hybrid system, the simulator outputs the results in the form of malicious classifications made and total time spent to make those classifications. When simulating a system that consists of high-interaction client honeypots, the web pages are simply placed in the queue to which

the high-interaction client honeypot has access without going through the low-interaction client honeypot.

The simulator was functionally tested for correctness. In addition, it was run with parameters identical to those in our implementation presented in the previous section. With a false negative rate of 46.15%, a false positive rate of 5.86%, and a service time of 0.05 seconds for the low-interaction client honeypot, and a false negative rate of 0%, a false positive rate of 0%, and a service time of 2.95seconds for the high-interaction client honeypot, and a base rate of $p = 0.00213$, the simulator classified eight malicious web pages in 3 hours and 51 minutes (this compares to seven malicious classifications in 3 hours and 45 minutes for our actual implementation) for the hybrid system. The simulator classified 13 malicious web pages in 49 hours and 59 minutes with a system that consisted exclusively of high-interaction client honeypots (this compares to 13 malicious classifications in 49 hours and 59 minutes for our actual implementation). The slight discrepancies are explained by the randomness introduced when generating web pages based on the base rate and the client honeypot's classification based on the false negative and false positive rates.

Results

With the simulator available, the base rate could be manipulated and the performance of the hybrid system compared to the high-interaction client honeypot using the true positive cost curve. The true positive cost curve, while it was specifically developed for evaluation of high-interaction client honeypots, can be used for evaluation of the hybrid system, because the hybrid system exhibits similar characteristics in detection accuracy: It exhibits the same negligible false positive rate that high-interaction client honeypots exhibit. As a result, the true positive cost curve is appropriate for evaluation of the hybrid system.

First, the hybrid system is compared to the high-interaction client honeypot system with identical parameters. The true positive cost curve is

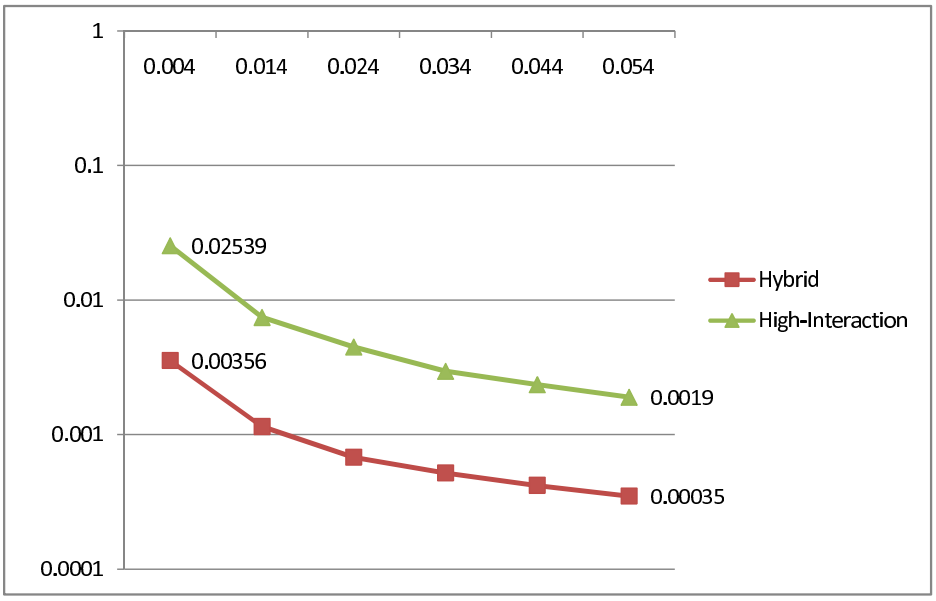


Figure 7.9: Hybrid Client Honeypot vs. High-interaction Client Honeypot True Positive Cost Curve

created by plotting the cost to identify one malicious URL on varying base rates for the hybrid system as well as the high-interaction client honeypot, as shown in Figure 7.9. This true positive cost curve shows a lower cost to identify malicious URLs for the hybrid system across all base rates. For a base rate of $p = 0.004$, the cost of the hybrid system is 0.00356 US dollars compared to 0.02539 US dollars for the high-interaction client honeypot. This equates to a factor of 7.13, i.e., the high-interaction client honeypot system is 7.13 times as expensive as the hybrid system. With increasing base rate, the cost decreases. At a base rate of $p = 0.054$, the cost of the hybrid system is 0.00035 US dollars compared to 0.0019 US dollars for the high-interaction client honeypot. This equates to a factor of 5.43. Note that the factor decreases with increasing base rate. This is related to the fact that the low-interaction client honeypot's filtering of benign pages for inspection by the high-interaction client honeypot decreases in effectiveness, because the number of benign pages is decreasing.

The cost of the hybrid system, however, is not always lower than the cost of the high-interaction client honeypot. Equation 7.20 needs to hold true for the hybrid client honeypot system to perform better than the high-interaction client honeypot. The dependent variables are the service time and the detection accuracy of the hybrid system, which is determined by the individual components. If a service time of 2.95 seconds and a false negative rate of zero is assumed for the high-interaction client honeypot, Equation 7.20 will evaluate to false if the service time of the hybrid system drops below 1.589 seconds (equivalent to a service time of 1.415 seconds for the low-interaction client honeypot) with a constant false negative rate of 0.4615 or if the false negative rate increases to 92.30% with a constant service time of 0.227 seconds (equivalent to a service time of 0.05 seconds for our low-interaction client honeypot).

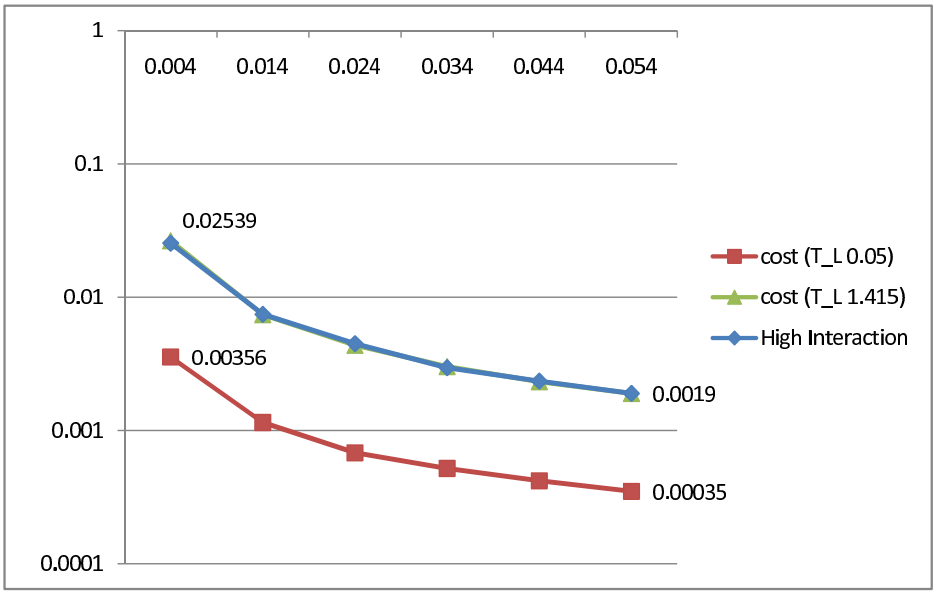


Figure 7.10: Hybrid Client Honeypot with Varying Service Times vs. High-interaction Client Honeypot True Positive Cost Curve

The simulator's data supports the model. As shown in Figure 7.10, the hybrid client honeypot system performs identically to the high-interaction

client honeypot as soon as the service time of the hybrid system exceeds 1.589 seconds (or 1.415 seconds of service time for the low-interaction client honeypot). As such, if the low-interaction client honeypot component is faster than 1.415 seconds with our high-interaction client honeypot, it will be beneficial to combine them into a hybrid system.

Note that these service times are not absolute service times, but are dependent on the performance of the high-interaction client honeypot. If the high-interaction client honeypot were to show service times other than 2.95 seconds, the service times of the low-interaction client honeypot would also change for a hybrid system to be beneficial. The model presented above can find these varied service times.

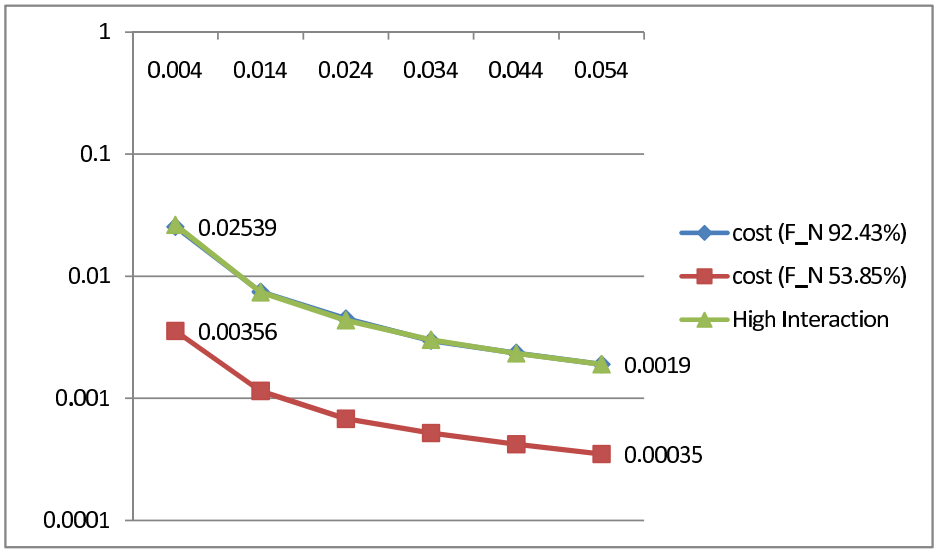


Figure 7.11: Hybrid Client Honeypot with Varying False Negative Rates vs. High-interaction Client Honeypot True Positive Cost Curve

Similar to service time, the simulator's data also supports the model in respect to false negative rate. As shown in Figure 7.11, the hybrid client honeypot system performs identically to the high-interaction client honeypot as soon as the false negative rate of the hybrid system exceeds 92.43%. As such, if the low-interaction client honeypot component exhibits a false

negative rate of 92.43% or lower, it will be beneficial to combine it with our high-interaction client honeypot into a hybrid system. Note that these rates may change depending on the characteristics of the individual components, and the model can be used to obtain the exact rates at which it is beneficial to combine low- and high-interaction client honeypots into a hybrid system. In our case, even a low-interaction client honeypot that is fast but misses almost all malicious pages would be beneficial in a hybrid system over a high-interaction client honeypot system.

Conclusion

To conclude, the simulator allowed us to vary a wide range of dependent variables of the hybrid client honeypot system, such as detection speed and accuracy of the individual components as well as the base rate of the operating environment. The data collected with the simulator supports the model of the hybrid client honeypot presented in the previous section. It showed that combining a low-interaction client honeypot based on the static analysis detection method with our high-interaction client honeypot Capture-HPC is beneficial compared to a high-interaction client honeypot system with identical resources. The true positive cost curve showed that the hybrid system evaluates better across all base rates.

However, a hybrid client honeypot is not beneficial under all conditions. Equation 7.20 needs to hold for a low-interaction client honeypot to contribute positively to a hybrid system. The simulator's data supports this equation. We showed that once the low-interaction client honeypot's speed drops below 1.415 seconds or its false negative rate exceeds 92.30% combining a low-interaction client honeypot into a hybrid system would not be beneficial.

7.3 Conclusion

In this chapter, we illustrated the usefulness of low-interaction client honeypots when combined into a hybrid client honeypot system in which classifications of the low-interaction client honeypot are confirmed by the high-interaction client honeypot. We presented such a hybrid system and showed the effect of such a system on performance and detection accuracy using a multiqueue model.

Evaluation on a sample of web pages with an actual hybrid client honeypot system that combined a low-interaction client honeypot that identifies malicious web pages using a static analysis method with our high-interaction client honeypot Capture-HPC is indeed favorable over a high-interaction client honeypot system with identical resources. With a base rate of $p = 0.004$, the hybrid system was capable of identifying a malicious web page at a cost of 0.053 US dollars compared to a high-interaction client honeypot system that was able to identify a malicious web page at a cost of 0.480 US dollars. This equates to a factor of nine.

However, the hybrid client honeypot model predicted that combination of a low-interaction client honeypot into a hybrid system may not always be beneficial. If the detection accuracy is low or the low-interaction client honeypot is too slow, it may actually hurt performance when combined into a hybrid system. The model showed that a low-interaction client honeypot that incorporates the method of analyzing network traffic when loading a page would not be beneficial.

This model was supported by data collected using a simple simulator. We used it to illustrate the effect of a hybrid system on a varying base rate. It allowed us to evaluate the hybrid system against the high-interaction client honeypot using the true positive cost curve. Across all base rates, the true positive cost curve showed favorable costs for the hybrid system over the high-interaction client honeypot.

The performance of the hybrid system, however, was highly depen-

dent on the performance of the individual components. We explored the effect of speed and false negative rate of the low-interaction client honeypot on the hybrid system. We showed that with a false negative rate of approximately 46.15%, the service time of the low-interaction client honeypot needs to stay below 1.415 seconds to evaluate better over a high-interaction client honeypot with identical resources. Similarly, if the service time stays constant at 0.05 seconds, but the false negative rate is increased, a hybrid system would evaluate better over a high-interaction client honeypot system as long as the false negative rate stays below 92.43%.

Initially one would question the usefulness of this system in light of the false negative rates. As we mentioned in chapter 3, antivirus software exhibits a false negative rate of approximately 70%. While this is the case, the antivirus software – in a crawling detection scenario - would behave similarly to a high-interaction client honeypot. As the antivirus software primarily focuses on detecting the malicious binary pushed as a result of a successful attack, the antivirus would require a vulnerable browser and a classification to give the attack an opportunity to succeed. Only if it succeedsdoes the antivirus software have the opportunity to detect the attack - with a false negative rate of approximately 70%. If malware is indeed pushed, the high-interaction client honeypot is likely to outperform antivirus software as it does not work with signatures. Nevertheless, a hybrid client honeypot system – even with a false negative rate of 46.15% – is more cost effective in detection of malicious web pages than antivirus software, just as hybrid client honeypots are more cost effective than high-interaction client honeypots.

At the beginning of this chapter, we mentioned that the cost to identify 80,000,000 malicious web pages embedded in 20,000,000,000 web pages assuming a base rate of $p = 0.004$ would be approximately 2,030,000 US dollars. With the introduction of the hybrid system, this cost can be reduced to approximately 285,000 US dollars, a significant reduction. The hybrid client honeypot system made frequent, comprehensive inspection

of web pages on the Internet feasible.

Chapter 8

Conclusions

Drive-by-download attacks are a serious threat to the confidentiality, integrity, and availability of computers and networks. Client honeypots are devices that are capable of detecting and subsequently protecting against these attacks, but they are faced with challenges of their own: They are slow and are known to miss attacks. A deployment of this detection technology on a large scale to detect drive-by-download attacks given the billions of web pages is prohibitively expensive and infeasible.

As part of this work, we developed a hybrid client honeypot system that is more cost-effective in detecting web pages that launch drive-by-download attacks than existing client honeypots. We have shown that the hybrid client honeypot system exhibits characteristics that allow for deployment on a large scale to inspect large portions of the Internet.

Our conclusions are summarized by chapter:

- *Chapter 2 Background* The thesis is concerned with the detection of client-side attacks in a web context. We began this thesis by reviewing the general approaches of detecting attacks and defining the terminology on attack detection that we adopted. We categorized client-side attacks in a web-based context according to the three impacts an attack can have on computer systems: confidentiality, availability, and integrity. We analyzed the vulnerability landscape,

which resulted in defining the focus of this work: Detection of the most prevalent type of web-side, client-side attack: drive-by-downloads on the primary target: Microsoft's Internet Explorer 6.0.

- *Chapter 3 Related Work* We then reviewed the related work that concerned itself with the detection of and protection from drive-by-download attacks. In particular, we concentrated on a review of a particular detection technology, client honeypots, and analyzed the gaps that exist in effectively detecting drive-by-download attacks with client honeypots. We identified four main gaps: lack of a publically and freely available client honeypot research platform; lack of evaluation techniques, which hinders research and the evolution of client honeypots; a general weakness in the experimental design of the work on detection of drive-by-download attacks with client honeypots, which fails to mitigate the risk to internal and external validity of the work; and the gap of low-interaction client honeypots able to assess whether a web page belongs to a malware distribution network. Filling these gaps would allow us to model, develop, and evaluate a hybrid client honeypot system that is able to detect malicious web pages in a more cost-effective manner.

- *Chapter 4 True Positive Cost Curve* We developed a method for high-interaction client honeypots that can be used to evaluate client honeypots against their primary purpose of identification of malicious web pages: the true positive cost curve (TPCC.) The TPCC borrows on existing evaluation methods of intrusion detection systems, but incorporates the unique factors of client honeypot technology: a negligible false positive rate and the fact that client honeypots are active devices that are tasked with finding malicious web pages; resource costs are therefore of much greater importance. The feasibility and usefulness of the TPCC are illustrated through newly developed improvements on the visitation algorithm of high-interaction client

honeypots. Further, it is illustrated how the TPCC can evaluate different configuration settings of a client honeypot for optimizations in a specific operating environment.

- *Chapter 5 Experimental Design* Weaknesses in the experimental design of earlier work on detection of drive-by-download attacks with client honeypots was identified in Chapter 3. A goal of our work was to stand on a solid foundation with a strong experimental design that mitigates risks to internal and external validity. In Chapter 5, a methodology of identifying and mitigating risks in a systematic and thorough manner is presented: the hazard and operability (HAZOP) study. Measurement studies are used to illustrate the process of HAZOP. A major risk identified is uncontrolled variables. We use uncontrolled variables as an example to illustrate the impact of failure to mitigate risks appropriately. First, it is shown that the URL source can greatly impact measurements; second, it is shown that time can also have a major impact on measurement. Mitigation of risks identified using the HAZOP were incorporated into our experimental designs.

- *Chapter 6* Low-Interaction Client Honeypots We developed and evaluated several new detection techniques that assessed whether a web page belongs to a malware distribution network. These methods could be incorporated into a lightweight low-interaction client honeypot that is generally faster than high-interaction client honeypots at finding malicious web pages on a network. However, at the same time, they produce false positives and therefore would not be suitable as stand-alone systems to detect malicious web pages.

- *Chapter 7 Hybrid Client Honeypot* The usefulness of these low-interaction client honeypots when they are combined with high-interaction client honeypots into a hybrid client honeypot system was illustrated. A model of a hybrid client honeypot system was developed to assess

the impact of the hybrid client honeypot system's ability to identify malicious web pages based on the speed and detection accuracy of the underlying low- and high-interaction client honeypot components. The model showed that not all low-interaction client honeypots are suitable to produce hybrid client honeypot systems to detect more malicious web pages.

A candidate low-interaction client honeypot system, based on the developed statistical static analysis of initial HTTP responses, was used to evaluate a model of a hybrid client honeypot system. As the hybrid client honeypot system exhibited the same characteristics as a high-interaction client honeypot with a negligible false positive rate, the TPCC could be used to evaluate the system. It showed that a hybrid client honeypot system is more cost-effective and could be used as part of a large-scale deployment to monitor the Internet.

8.1 Contributions

In carrying out this research, we have made five main contributions to the client honeypot literature:

8.1.1 High-Interaction Client Honeypot

The design and implementation of a high-interaction client honeypot Capture-HPC, a client honeypot research platform, is a main contribution of this thesis. While three high-interaction client honeypots existed when this work commenced [96, 157, 159], these systems either were not publicly available or did not meet the resource and forensic requirements necessary to conduct research on malicious web servers in a cost-effective manner. As a result, based on forensic requirements we have developed, a new open-source high-interaction client honeypot, named Capture-HPC, was created, which allows researchers and security professionals conduct

research on malicious web pages and client honeypots. Capture-HPC has been incorporated into the other available open-source high-interaction client honeypot, HoneyClient [157], and is being used in numerous research and commercial projects [135, 160, 46, 27, 171].

Based on our client honeypot implementation and analysis of existing client honeypots, we developed a component model of client honeypots. We identified three core components of a client honeypot as shown in Figure 4.4: Queuer, Visitor, Analysis Engine. This model allows researchers to agree on the object of study, allows for focus of specific areas within the object of study, and provides a framework for communication of research around client honeypots. As increased understanding results from this model, it allows for improved design and development of client honeypot technology. This model has been accepted as a client honeypot model by the research community [116, 169, 33, 142].

8.1.2 True Positive Cost Curve

The true positive cost curve (TPCC) is the second main contribution of this thesis. The TPCC is a method that takes into account the unique characteristics of client honeypots – speed, detection accuracy, and resource cost – and provides a simple, cost-based mechanism for evaluating and comparing client honeypots in an operating environment. As such, the TPCC provides a foundation for improving client honeypot technology.

The applicability of the TPCC in evaluating client honeypots is demonstrated through improvements to client honeypot visitation algorithms developed by us and Wang et al. [159]: the bulk, bulk & sequential, and divide-and-conquer algorithms. The TPCC showed that the performance of the bulk algorithm is generally more cost-effective than the other visitation algorithms; however, under certain conditions, namely a low base rate, the divide-and-conquer algorithm outperforms all other visitation algorithms.

TPCC evaluates a client honeypot in an operating environment. As such, the TPCC may also be used by an operator to evaluate different configurations and settings of the client honeypot within a specific operating environment; in other words, the TPCC can be used to tune a client honeypot in a specific operating environment. Application of the TPCC in such a way is demonstrated by tuning a client honeypot in an operating environment with malicious web pages that employ time bombs or IP tracking functionality.

8.1.3 Mitigation of Risks to the Experimental Design with HAZOP

Mitigation of risks to internal and external validity on the experimental design using hazard and operability (HAZOP) study is the third main contribution of this thesis. This methodology addresses risks to intent (internal validity) as well as generalizability of results beyond the experimental setting (external validity) in a systematic and thorough manner.

Measurement studies are used to illustrate the process of HAZOP. A major risk identified is uncontrolled variables. We use uncontrolled variables as an example to illustrate the impact of failure to mitigate risks appropriately. First, it is shown that the URL source can greatly impact measurements; second, it is shown that time can also have a major impact on measurement.

8.1.4 Low-Interaction Client Honeypots

Malicious web pages are usually part of a malware distribution network that consists of several servers that are involved as part of the drive-by-download attack. Development and evaluation of classification methods that can be incorporated into a low-interaction client honeypot network is the fourth main contribution. These methods are used to assess whether a web page is part of a malware distribution network. In contrast to the

high-interaction client honeypot, one would not have to load the web pages in a dedicated system nor monitor the system for unauthorized state changes. Rather, a simulated client could be used to retrieve the web page and the server response analyzed directly. The two methods are based on analyzing the dynamic behavior when loading a web page and statistical analysis of elements found on the page. As shown in this thesis, the methods can be used to identify malicious web pages quickly; however, at the same time, many false alerts would be generated.

8.1.5 Hybrid Client Honeypot System

The fifth main contribution of this thesis is the hybrid client honeypot system. A model is developed that is capable of optimizing resources and estimating cost and detection accuracy of a hybrid client honeypot system based on the underlying low- and high-interaction client honeypot components. The hybrid client honeypot system is capable of identifying malicious web pages in a cost-effective way on a large scale. The hybrid client honeypot system outperforms a high-interaction client honeypot with identical resources and identical false positive rate.

The model allows assessment of whether low-interaction client honeypots can be beneficial when combined into a hybrid client honeypot system. Two candidate low-interaction client honeypots, based on statistical static and dynamic behavioral methods, are evaluated. The hybrid client honeypot model is used to identify a low-interaction client honeypot component that could be combined into a beneficial hybrid client honeypot system.

The model is evaluated with an actual implementation of a hybrid client honeypot system.

8.1.6 Publications

Part of the research discussed in this thesis has appeared in the following publications:

- C. Seifert, P. Komisarczuk, and I. Welch, "True Positive Cost Curve: A Cost-Based Evaluation Method for High-Interaction Client Honeypots," in SECURWARE, Athens, 2009.
- C. Seifert, P. Komisarczuk, and I. Welch, "Application of divide-and-conquer algorithm paradigm to improve the detection speed of high interaction client honeypots," in 23rd Annual ACM Symposium on Applied Computing Ceara, Brazil, 2008.
- C. Seifert, B. Endicott-Popovsky, D. Frincke, P. Komisarczuk, R. Muschevici, and I. Welch, "Justifying the Need for Forensically Ready Protocols: A Case Study of Identifying Malicious Web Servers Using Client Honeypots," in 4th Annual IFIP WG 11.9 International Conference on Digital Forensics, Kyoto, 2008.
- C. Seifert, B. Endicott-Popovsky, D. Frincke, P. Komisarczuk, R. Muschevici, and I. Welch, "Identifying and Analyzing Web Server Attacks," in Advances in Digital Forensics IV, I. Ray and S. Shenoi, Eds. New York: Springer, 2008, pp. 151-162.
- C. Seifert, R. Steenson, I. Welch, P. Komisarczuk, and B. Endicott-Popovsky, "Capture - A Behavioral Analysis Tool for Applications and Documents," in 7th Digital Forensics Research Workshop Conference, Pittsburgh, 2007.
- C. Seifert, R. Steenson, T. Holz, Y. Bing, and M. A. Davis, "Know Your Enemy: Malicious Web Servers," The Honeynet Project, 2007, p. available from http://www.honeynet.org/papers/mws/; accessed on 25 September 2007.

- C. Seifert, V. Delwadia, P. Komisarczuk, D. Stirling, and I. Welch, "Measurement Study on Malicious Web Servers in the .nz Domain," in 14th Australasian Conference on Information Security and Privacy (ACISP), Brisbane, 2009.

- C. Seifert, I. Welch, and P. Komisarczuk, "Taxonomy of Honeypots," Wellington: Victoria University of Wellington, 2006, pp. available from http://www.mcs.vuw.ac.nz/comp/Publications/index-byyear-06.html; accessed on 14 July 2006.

- C. Seifert, I. Welch, and P. Komisarczuk, "HoneyC - The Low-Interaction Client Honeypot," in NZCSRCS, Hamilton, 2007, pp. available from http://www.mcs.vuw.ac.nz/ cseifert/blog/images/seifert-honeyc.pdf; accessed on 10 September 2006.

- C. Seifert, I. Welch, P. Komisarczuk, C. Aval, and B. Endicott-Popovsky, "Identification of Malicious Web Pages Through Analysis of Underlying DNS and Web Server Relationships," in 3rd IEEE Conference on Local Computer Networks, Montreal, 2008.

- C. Seifert, P. Komisarczuk, and I. Welch, "Identification of Malicious Web Pages with Static Heuristics," in Austalasian Telecommunication Networks and Applications Conference, Adelaide, 2008.

8.2 Delimitations and Limitations of the Study

The generalizability of our results is limited in a number of ways.

Attack Type The work focuses on the identification of malicious web pages that launch drive-by-download attacks. The identification utilizes low-interaction client honeypots that use some characteristic of the page to assess whether it is malicious. If a different type of client-side attack does not exhibit a common characteristic that can be observed with a low-interaction client honeypot, the detection accuracy of the low-interaction

client honeypot would fall and it would invalidate the inclusion of such a component into a hybrid system. As such, our work on the hybrid client honeypot can only be generalized to attacks that exhibit some common characteristics that can be observed with low-interaction client honeypots.

Attack Target The work focuses on the identification of malicious web pages that launch drive-by-download attacks that target installations of Microsoft's Internet Explorer 6.0. While we believe, as a result of our analysis in Chapter 2, that drive-by-downloads and attacks on Microsoft's Internet Explorer 6.0 are most prevalent and therefore lent themselves as a subject of study, our approaches may not be generalizable to different types of client and even different versions of Internet Explorer. For instance, attacks that target clients that are under a different patch strategy may be transient and not reside on web pages to be detected. Rather, they could reside on routers and randomly inject attack code. A crawling strategy to detect such attacks would fail.

8.3 Future Work

The research represents a variety of aspects that deal with the evaluation of client honeypots, improvement of client honeypots, and study of malicious web pages. Much scope for future work remains. Here we discuss a few ways this research could be continued.

8.3.1 Evaluation of Antivirus Software

Antivirus software has traditionally focused on identification of malicious binaries. Just in recent years has the antivirus industry started to include functionality that is aimed at detection of client-side attacks, such as drive-by-download attacks, by instrumenting the browser and evaluating internal client state to determine whether an attack occurs. An evaluation and comparison of these advanced antivirus solutions is left for future work.

8.3.2 Extensions to the TPCC

The TPCC evaluates client honeypots based on the premise that a client honeypot performs better if it detects more malicious web pages with identical resources. As part of future work, the TPCC can be expanded to incorporate the value of a malicious web page, as not all malicious web pages are of equal value. For instance, it is much more valuable to detect a malicious web page that is capable of launching a zero-day attack than a malicious web page that launches an attack on an older vulnerability that most users have already patched.

The value of a page could incorporate several factors:

- Attack success probability - the probability that an attack is successful; this value is based on the distribution of vulnerable clients paired with the exploits available to the specific web page. For example, a web page that launches a zero-day vulnerability has a higher attack success probability because all clients are vulnerable.

- Interaction probability - the probability that a randomly selected client interacts with the malicious web page; a web page that is popular, for instance, will be of higher value than an unpopular page, because the likelihood that the popular web page is visited is higher.

- Trigger probability - the probability that a randomly selected malicious web page launches an attack; some malicious web pages may employ a strategy to evade detection by only occasionally launching attacks; these web pages may be of lower value as the likelihood that the page will trigger itself is lower.

To incorporate the notion of the value of a malicious web page, more data about clients and malicious web pages needs to be detected. For instance, to assess attack success probability, the vulnerabilities of clients need to be mapped to exploits the malicious web pages are capable of

launching. Both HoneyMonkey and PhoneyC are capable of conducting such mapping to some extent.

8.3.3 Additional Visitation Algorithms

Four visitation algorithms were presented and evaluated. These visitation algorithms all followed a synchronous push model in which the client inspected a set of k web pages before returning results and continuing to inspect another set of k web pages. This leads to wasted resources. For instance, if two URLs are to be inspected by a client honeypot, and one can be retrieved and classified in 10 seconds, the other in 50 seconds, the client honeypot would waste 40 seconds of the client honeypot's resources waiting for the second URL to be retrieved and classified. A pull model in which URLs are fetched from the queue as resources become available are expected to further increase the speed of client honeypots and will be explored as part of our future work.

8.3.4 Investigation of False Negatives

False negatives are threats to the external validity of measurement studies. False negatives of high-interaction client honeypots have been observed, but little research has been done to establish the reason or the magnitude of the problem. Especially in cases where measurement studies would indicate a decline in attacks, the question of false negatives needs to be further examined. A measurement study could conclude there is a decline in attacks, whereas in reality an increase in false negatives is being measured.

Two types of false negatives are of particular interest, as we have linked specific web exploitation frameworks to incorporating such functionality that would manifest itself in false negatives: IP tracking and network-dependent triggering. Recall that in IP tracking, a malicious web page may launch an attack only once and cease to do so on subsequent attacks;

network-dependent triggering assesses the network of the client and triggers only if the network does not reside in a malicious web page's blacklist. These false negatives may be investigated through the deployment of a globally distributed client honeypot network or distributed client honeynet.

Further, false negatives may be investigated through case studies that examine web exploitation kits and reverse engineer malicious web pages.

8.3.5 Expand Client Honeypot Research Platform

Capture-HPC is an open-source, publically and freely available client honeypot research platform. It has been incorporated into the other available open-source high-interaction client honeypot HoneyClient [157] and is being used in numerous research and commercial projects [135, 160, 46, 27, 171]. It is primarily used for data collection.

A central data storage and a distributed analysis platform are two components that are currently missing from the research platform that would further the research and successes in this area. The data storage component needs to be able to store large amounts of data from various sources and make this data available through a distributed analysis platform that is capable of conducting data mining experiments that can be used to develop new classifiers, characterize malicious web pages, visualize trends, etc.

8.3.6 Improvements on Low-Interaction Client Honeypots

The low-interaction client honeypots described in Chapter 6 were making decisions about the malicious nature of a web pages based on a classifier developed. The classifier – in form of a decision tree - had various drawbacks: staleness, evasion, and brittleness.

A lot of room for improvement exists to address these shortcomings. First, we would like to explore the staleness aspect of the methods and

answer the question as to whether and how much their detection accuracy decays over time. Repeated evaluations of the method with high-interaction client honeypots would need to be conducted over several months to answer these questions. If staleness is observed, an automated retraining may be initiated based on data collected by a high-interaction client honeypot. This could be based on the existing features, but mechanisms to explore additional feature sets in an automated way may need to be developed. Depending on the decay rate of the classifiers, the retraining rate could be optimized to keep staleness at a minimum.

Second, malicious web pages may evade a low-interaction client honeypot's malicious classification by adjusting its structure and behavior to blend into the mix of benign web pages. For instance, it may move the exploit to an internal script. We would like to explore the robustness of evasion by combination of several low-interaction client honeypot methods together. While it may be possible to evade one technique, it becomes increasingly difficult as new classification methods are added. A voting scheme could be used to result in a final classification.

Third, brittleness could be addressed by exploration of more flexible machine learning models. We chose the decision trees because they nicely allow for expert evaluation of the extracted knowledge. However, if a node makes an incorrect decision towards the root of the tree, it may result in an incorrect classification. Alternative classifiers, such as Bayesian classifier and support vector machine classifier, are classifiers we would like to explore as part of future work.

8.3.7 Improvements on High-Interaction Client Honeypots

High-interaction client honeypots are crucial to data collection and it is likely that they will continue to be of importance even with new types of detection technologies, such as low-interaction client honeypots, appearing. In Chapter 4, we introduced the TPCC and made and evaluated sev-

eral improvements to the visitation algorithm of the Visitor component. Additional opportunities to improvements exist:

- Visitor - Evasion techniques that a malicious web page employs are rooted in the fact that the malicious web page detects a client honeypot and chooses not to launch an attack. The Visitor component is the component that interacts with the web page; deceptive techniques could be used to counter evasion attempts and further improve client honeypot detection of drive-by-download attacks. A deception model known as the deception planning loop has been applied to identify the current status of client honeypots [29].

- Queuer - Opportunities exist to improve client honeypots through improvements on the Queuer component. In Chapter 5, we illustrated that URL source can have an impact on the base rate. Further exploration of how to keep the base rate high through adjustment of the Queuer component will overall improve client honeypots according to the TPCC. A Queuer component, for instance, could be a crawler that conducts scoring of links prior to adding the link to the queue to be visited. Input of such a link scoring mechanism may be the category of the current page, classification of the current page, classification of the current site, etc.

- Analysis Engine - As client honeypot technology becomes more widely adopted, attackers will further adjust their attacks to evade detection. High-interaction client honeypots monitor visible state changes on the system to assess whether an attack occurs. If an attack were to reside completely within the client process without causing state changes, it may go undetected. Further development of the analysis engine to monitor memory and process state may reveal such attacks.

8.3.8 Assessment of Malware Distribution Network Membership

The low-interaction client honeypots described in Chapter 6 assess whether a page belongs to a malware distribution network through examination of static elements of the HTTP response and dynamic characteristics of the servers contacted when loading a page. Additional information could be collected and utilized to make an assessment of malware distribution network membership, such as specific requests made when loading a page, malware hashes, malware behavior, etc. With additional information, the method of assessing malware distribution network membership could be expanded to assess the membership of a specific malware distribution network. Behavior and data collected about a malicious web page could be mined to generate malware distribution fingerprints or signatures that would allow such an assessment to be made.

8.4 Summary

Drive-by-download attacks are a serious threat to the confidentiality, integrity, and availability of computers and networks. Client honeypots are devices that are capable of detecting and subsequently protecting against these attacks, but they are faced with challenges of their own: they are slow and are known to miss attacks. A deployment of this detection technology on a large scale to detect drive-by-download attacks given the billions of web pages is prohibitively expensive and infeasible.

As part of this work, we developed a hybrid client honeypot system that is more cost-effective in detecting web pages that launch drive-by-download attacks than existing client honeypots. We have shown that the hybrid client honeypot system exhibits characteristics that allow for a deployment on a scale to inspect large portions of the Internet today.

We developed an evaluation method for client honeypots, applied the

HAZOP to mitigate risks of internal and external validity to our experimental design, and developed and evaluated a hybrid client honeypot that is able to detect malicious web pages more cost-effectively.

Appendix A

Glossary

- ActiveX component - A proprietary web browser plug-in for a Microsoft web browser that can be placed inside and distributed as part of a document [87].
- Activity - An event or a sequence of events in a given context.
- Alarm - A report of an error that may lead to security failure, optionally including indications whether the error led to security failure. The report may include diagnostic information about the fault, i.e., the activity that threatens the security policy that led to the generation of the report. Alarms are also referred to as alerts.
- Attack - A malicious activity threatening the security policy. It is therefore a malicious external fault.
- Attack vector - A generalization of the collection of possible attacks.
- Base rate - The percentage of attacks in a set of events.
- Client - A computer or program that requests and consumes services from other computers via a network. It is actively initiating requests to servers.

- Denial-of-service - Failure in the assurance of timely and reliable access to data.
- Domain name - A textual representation of an Internet address that permits location and communication to servers on the Internet.
- Downloader - A program that is designed to download and execute malware. Downloaders are usually small programs. Downloaders are required in a drive-by-download attack when the available memory does not hold the malware payload.
- Error - The part of the system state that is liable to lead to failure.
- Event - Something that happens or takes place.
- Exploit - A program or a piece of code that encapsulates an attack.
- Exploit server - A web server that exists exclusively to serve exploits. These exploits are imported from front-end URLs. As such, no front-end URL to exploit servers exist.
- Failure - A deviation of a delivered service to fulfill its intended function.
- False negative - An event corresponding to the occurrence of an attack that is not detected as such. This means that no alarm is raised due to either a lack of coverage or to excessive latency; also called a miss.
- False positive - An event corresponding to an alarm generated in the absence of an attack, i.e., a false alarm.
- Fault - The adjudged or hypothesized cause of an error.
- Front-end URL - A URL that a user navigates to by either clicking a link or typing the URL in the navigation bar of the web browser.

- HyperText Markup Language (HTML) - A markup language for authoring static web pages [154]. It consists of plain text with instructions for the browser to render the text in a particular way.
- HyperText Transfer Protocol (HTTP/ HTTPS) - Encapsulation of a set of rules that allow data to be exchanged between a web browser and a web server.
- Information disclosure - Failure in the assurance that data is protected and not disclosed to an unauthorized party.
- Intrusion - A malicious activity threatening the security policy that leads to a security failure, i.e., to a security policy violation. It is therefore a malicious external fault that leads to failure.
- Intrusion detection system - A piece of software and/or hardware designed to detect and alert to attacks that occur on a computer system it is monitoring.
- Landing Page - A URL that a user navigates to by either clicking a link or typing the URL in the navigation bar of the web browser.
- Loss of integrity - Failure in the assurance that data is unaltered by an unauthorized party.
- Malicious web page - A web page that directly launches a web-based client-side attack.
- Malicious web site - A web site that contains at least one malicious web page.
- Malicious web server - A web server that hosts at least one malicious web site.
- Malware - A malicious program that performs the intended tasks of an adversary. In a drive-by-download attack, the goal of the attacker is to push and execute malware on the client machine.

- Patch - A correction of a vulnerability.
- Payload - The initial piece of code that executes once the attack code successfully exploits a vulnerability. A payload can also be referred to as shellcode. In a drive-by-download attack, the payload could be actual malware or code that pulls and executes the malware. The available memory for the payload could be limited as per the properties of the vulnerability being exploited.
- Remote code execution vulnerability - A vulnerability that if exploited successfully, allows an attacker to execute arbitrary code. Integrity is impacted by attacks that target these vulnerabilities. A drive-by-download attack exploits remote code execution vulnerabilities of the operating system, web browser, and/or a web browser plug-in.
- Security failure - A failure against a defined security policy. In the context of a client, such as a web browser, the impact of the security failure could be information disclosure, denial of service, and/or loss of integrity.
- Server - A computer that delivers services to other computers linked by a network. It is passively awaiting requests for services from clients.
- Targeted attack - An attack that is aimed at a specific user or organization. Targeted attacks are not widespread, but rather are designed to attack a specific target.
- True negative - Correct decision to not rate a non-malicious event as an attack.
- True positive - Correct generation of an alarm. This means that an attack has been correctly detected, recognized, and reported.

- URL - A short string that identifies resources on the World Wide Web, such as documents, images, downloadable files, services, electronic mailboxes, and other resources [15]. URLs can consist of a protocol, a destination port, a server location, a path to the resource, and optional parameters.

- Vulnerability - An accidental fault or a malicious or non-malicious intentional fault in the requirements, the specification, the design, or the configuration of the system, or in the way it is used. The presence of a vulnerability may enable an error to lead to security failure.

- Web browser - A client that can request web content from web servers.

- Web browser plug-in - A software component that can be added to a web browser to enhance its capabilities. Web pages can utilize the additional functionality provided by web browser plug-ins.

- Web exploitation kits - Software that executes on malicious web servers. The software bundles a variety of exploits that are capable of launching a drive-by-download attack. In addition, these kits provide a variety of operational services to attackers, for example, the ability to track the number of users that have visited the malicious web server.

- Web page - A document created with HTML. It is a subset of the web content that a web server can deliver to a web browser. A web browser can render such a document on the screen. It usually consists of text, images and embedded multimedia components and programs.

- Web server - A server that delivers web content to web browsers.

- Web site - A collection of web pages on a web server that are served under the umbrella of a specific domain name.

- Web-based client-side attack - Attacks launched by malicious web servers that aim at compromising the integrity, availability, and confidentiality of the user, operating system, or web browser or its plug-ins.
- Zero-day attack - An attack that targets a vulnerability for which no patch is available.

Appendix B

Symbols

- p - The base rate of malicious web pages in a sample of N web pages. $p = 0.004$ in a sample of $N = 1000$ web pages denotes that 4 of these web pages are malicious.
- c_{URL} - The cost in US dollars to identify one malicious web page.
- t_{Algo} - The time in seconds required for a client honeypot to inspect a sample of N web pages.
- c_r - The resource costs, such as hardware costs, per time period t in US dollars.
- c_{MA} - The costs associated with manually analyzing a web page per time period t in US dollars.
- k - The number of web pages in a buffer.
- t_i - The average time in seconds required to retrieve a web page over the network.
- t_d - The average time in seconds required to render a web page.
- t_w - The classification delay in seconds introduced to give an exploit the opportunity to trigger.

- t_s - The average time in seconds required start the client application.
- t_r - The average time in seconds required reset a virtual machine.
- T_q - The average time in seconds required to generate a list of N URLs to be inspected by a client honeypot.
- $LF(k)$ - The load factor on the client application when classifying a set of k web pages.
- p_{mf} - The base rate of malicious web pages in a sample of N web pages that were classified as malicious by a low-interaction client honeypot. $p_{mf} = 0.04$ in a sample of $N = 1000$ web pages denotes that 40 of these web pages are malicious.
- N - The sample size of web pages.
- N_M - The number of malicious pages in the sample N.
- N_B - The number of benign pages in the sample N.
- λ - The rate responses can be processed by a system per time period t.
- λ_{max} - The maximum rate responses can be processed by a system per time period t.
- λ_{Lmax} - The maximum rate responses can be processed by a system per time period t of low-interaction client honeypot nodes.
- λ_{Hmax} - The maximum rate responses can be processed by a system per time period t of high-interaction client honeypot nodes.
- $T(p)$ - The average service time a response can be processed by a node within a system in seconds.
- $T_H(p_{mf})$ - The service time a response can be processed by a high-interaction client honeypot node within a system in seconds.

- $T_L(p)$ - The service time a response can be processed by a low-interaction client honeypot node within a system in seconds.
- $T_{Hy}(p)$ - The average service time a response can be processed by a hybrid client honeypot system in seconds.
- $TTotal_{Hy}(p)$ - The total service time a set of N responses can be processed by a hybrid client honeypot system in seconds.
- N_{Total} - The total number of nodes in system modeled as a multiserver queue.
- N_H - The total number of high-interaction client honeypot nodes in system modeled as a multiserver queue.
- N_L - The total number of low-interaction client honeypot nodes in system modeled as a multiserver queue.
- $alerts_L$ - The number of malicious web pages identified by a low-interaction client honeypot.
- FP - The false positive rate of a client honeypot.
- TP - The true positive rate of a client honeypot.
- FN - The false negative rate of a client honeypot.
- TN - The true negative rate of a client honeypot.
- FP_L - The false positive rate of the low-interaction client honeypot.
- TP_L - The true positive rate of the low-interaction client honeypot.
- FN_L - The false negative rate of the low-interaction client honeypot.
- TN_L - The true negative rate of the low-interaction client honeypot.
- FP_H - The false positive rate of the high-interaction client honeypot.

- TP_H - The true positive rate of the high-interaction client honeypot.
- FN_H - The false negative rate of the high-interaction client honeypot.
- TN_H - The true negative rate of the high-interaction client honeypot.

Appendix C

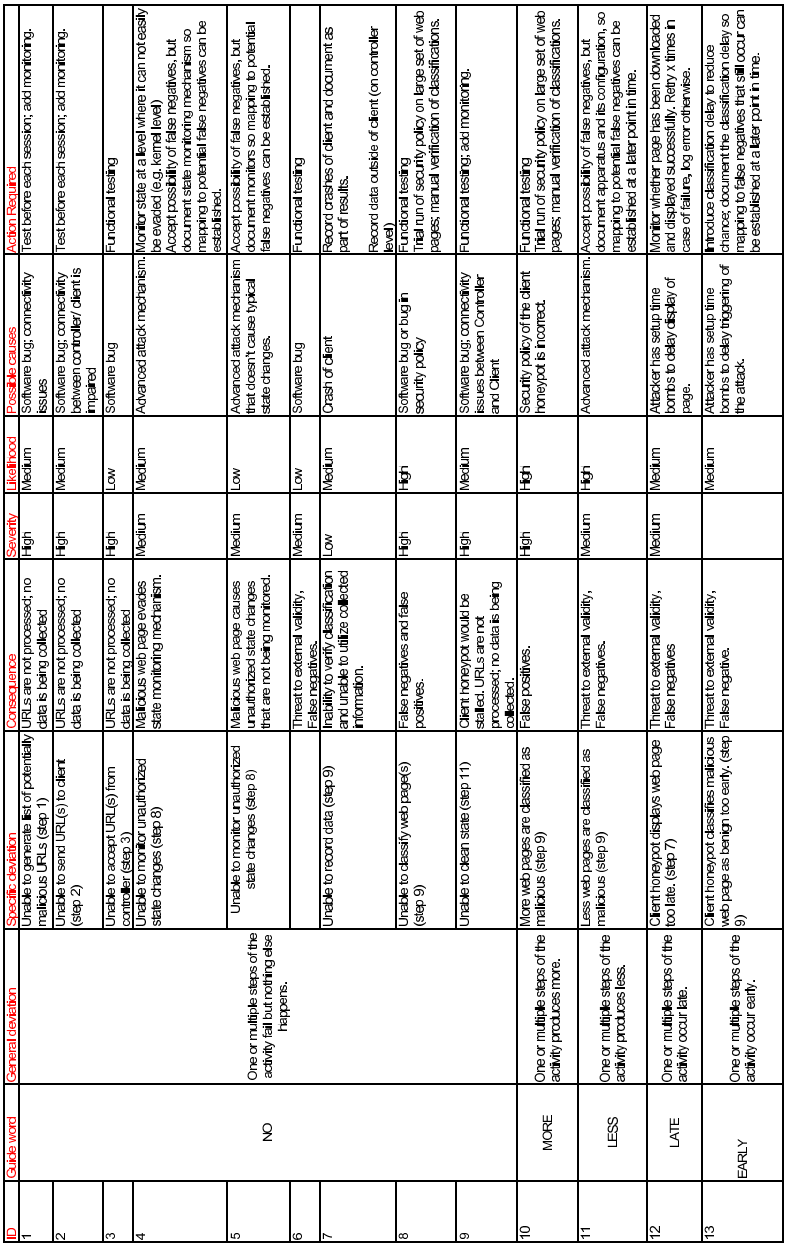
HAZOP

This appendix includes the analysis worksheets of the HAZOP analysis. Three analysis worksheets are included: The worksheet on the apparatus (the client honeypot) in Figures C.1 and C.2; the worksheet on the subjects (web pages) in Figure C.3; and the worksheet on the stimuli (making the request) in Figure C.4.

Each worksheet contains the following information:

- ID An identifier of the specific risk.
- Guide word the guide word that is applied to components to generate possible deviations from the intended purpose of the component.
- General deviation a general description of the deviation that could occur when applying the guide word.
- Specific deviation a specific description of the deviation that could occur when applying the guide word to components at a specific step in the process.
- Consequence a possible consequence of a specific deviation.
- Severity an assessment of the impact of the consequence on the experiment. Possible values are High, Medium, and Low.

- Likelihood the likelihood that the deviation occurs. Possible values are High, Medium, and Low.
- Possible causes a description of why a specific deviation could occur.
- Action required action that could mitigate the deviation from occurring; action that could control the deviation.

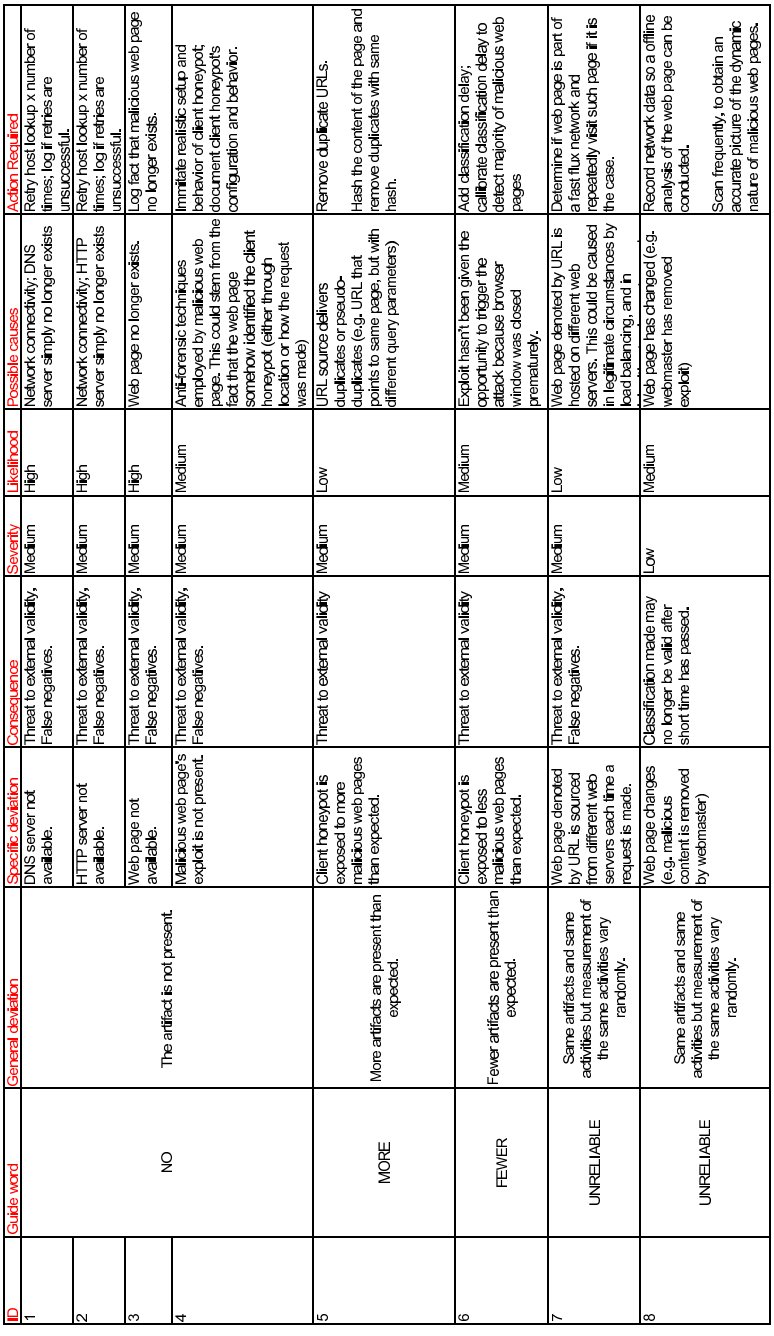


ID	Guide word	General deviation	Specific deviation	Consequence	Severity	Likelihood	Possible causes	Action Required
1	NO	One or multiple steps of the activity fail but nothing else happens.	Unable to generate list of potentially malicious URLs (step 1)	URLs are not processed; no data is being collected	High	Medium	Software bug; connectivity issues	Test before each session; add monitoring.
2			Unable to send URL(s) to client (step 2)	URLs are not processed; no data is being collected	High	Medium	Software bug; connectivity between controller/ client is impaired	Test before each session; add monitoring.
3			Unable to accept URL(s) from controller (step 3)	URLs are not processed; no data is being collected	High	Low	Software bug	Functional testing
4			Unable to monitor unauthorized state changes (step 8)	Malicious web page evades state monitoring mechanism.	Medium	Medium	Advanced attack mechanism	Monitor state at a level where it can not easily be evaded (e.g. kernel level) Accept possibility of false negatives, but document state monitoring mechanism so mapping to potential false negatives can be established.
5			Unable to monitor unauthorized state changes (step 8)	Malicious web page causes unauthorized state changes that are not being monitored.	Medium	Low	Advanced attack mechanism that doesn't cause typical state changes.	Accept possibility of false negatives, but document monitors so mapping to potential false negatives can be established.
6				Threat to external validity, False negatives.	Medium	Low	Software bug	Functional testing
7			Unable to record data (step 9)	Inability to verify classification and unable to utilize collected information.	Low	Medium	Crash of client	Record crashes of client and document as part of results. Record data outside of client (on controller level)
8			Unable to classify web page(s) (step 9)	False negatives and false positives.	High	High	Software bug or bug in security policy	Functional testing Trial run of security policy on large set of web pages; manual verification of classifications.
9			Unable to clean state (step 11)	Client honeypot would be stalled. URLs are not processed; no data is being collected.	High	Medium	Software bug; connectivity issues between Controller and Client	Functional testing; add monitoring.
10	MORE	One or multiple steps of the activity produces more.	More web pages are classified as malicious (step 9)	False positives.	High	High	Security policy of the client honeypot is incorrect.	Functional testing Trial run of security policy on large set of web pages; manual verification of classifications.
11	LESS	One or multiple steps of the activity produces less.	Less web pages are classified as malicious (step 9)	Threat to external validity, False negatives.	Medium	High	Advanced attack mechanism.	Accept possibility of false negatives, but document apparatus and its configuration, so mapping to potential false negatives can be established at a later point in time.
12	LATE	One or multiple steps of the activity occur late.	Client honeypot displays web page too late. (step 7)	Threat to external validity, False negatives	Medium	Medium	Attacker has setup time bombs to delay display of page.	Monitor whether page has been downloaded and displayed successfully. Retry x times in case of failure, log error otherwise.
13	EARLY	One or multiple steps of the activity occur early.	Client honeypot classifies malicious web page as benign too early. (step 9)	Threat to external validity, False negative.		Medium	Attacker has setup time bombs to delay triggering of the attack.	Introduce classification delay to reduce chance; document the classification delay so mapping to false negatives that still occur can be established at a later point in time.

Figure C.1: HAZOP Analysis Worksheet Apparatus 1 of 2

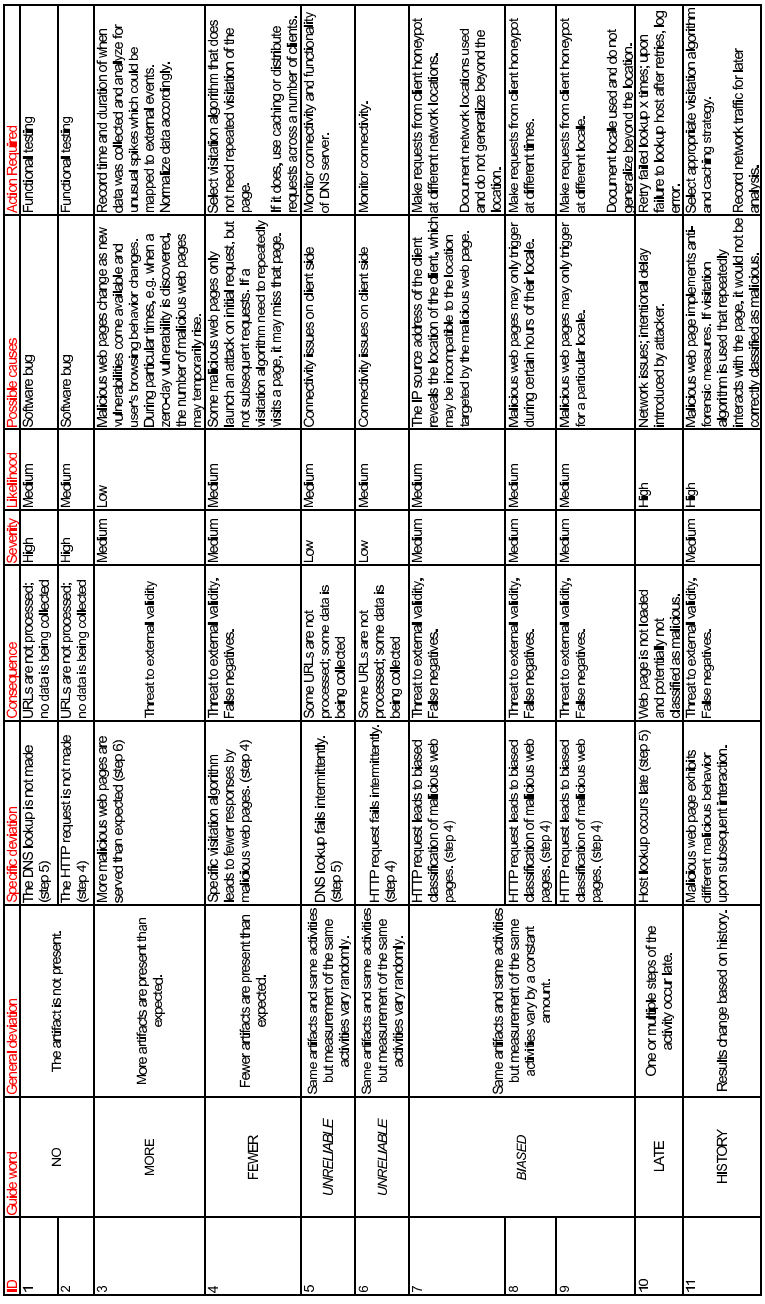
ID	Guide word	General deviation	Specific deviation	Consequence	Severity	Likelihood	Possible causes	Action Required
14	EARLY	One or multiple steps of the activity occur early.	Client honeypot cleans state of client too early.	While not all state changes would be recorded, the correct classification would still be made.	Low	High	Attacker has setup time bombs to delay triggering of the attack.	Record time stamps along with state changes. These will allow one to assess when the state changes occur and if necessary adjust the classification delay.
15	*OUT OF ORDER*	One or multiple steps of the activity occur out of order.	Client honeypot reports state changes out of order.	While the state changes would be out of order and an analysis of the attack would be more difficult, the correct classification would still be made.	Low	Low	Software bug	Record time stamps along with state changes.
16	*INDISTINGUISHABLE*	One or multiple steps of the activity occur out of order.	When visiting multiple web pages simultaneously, state changes can not be attributed to a specific web page.	Threat to external validity, False negatives.	Medium	Medium	Software bug	Functional testing
17			When visiting multiple web pages simultaneously, state changes can not be attributed to a specific web page.	Threat to external validity, False negatives.	Medium	Medium	Incompatible state monitoring and visitation algorithm (e.g. bulk and kernel level monitoring without process ID).	Select appropriate visitation algorithm and state monitoring technique that is capable to perform such mapping.
18	*UNRELIABLE*	One or multiple steps of the activity do not work as designed upon repeated execution	Unable to reliably generate list of potentially malicious URLs (step 1)	URLs are not processed; no data is being collected	High	Medium	Software bug; connectivity issues	Test before each session; add monitoring.
19			Unable to reliably send URL(s) to client (step 2)	URLs are not processed; no data is being collected	High	Medium	Software bug; connectivity between controller/client is impaired	Test before each session; add monitoring.
20			Unable to clean state (step 11)	Client honeypot would be stalled. URLs are not processed; no data is being collected.	High	Medium	Software bug; connectivity issues between Controller and Client	Functional testing; add monitoring.
21	*BIASED*	One or multiple steps of the activity introduces bias into the data collected	URLs selected by queuer introduce bias. (step 1)	Threat to external validity	Medium	High	Particular type of URLs are selected that exhibit different malicious behavior	Representative random sampling of input URLs from the larger population that measurement shall be generalized to.
22			Operating system, browser and browser plugins will bias detection towards malicious web pages that attack the specific operating system, browsers. For instance, if Firefox is used, a malicious web page that only attacks Internet Explorer will be missed.	Threat to external validity, false negatives.	Medium	High	Particular type of URLs are selected that exhibit different malicious behavior	Document operating system, browser and plug-ins used, and do not generalize beyond that configuration
23			Configuration of operating system, browser and browser plugins will bias detection towards malicious web pages that attack the specific configuration. For instance, if JavaScript of browser is disabled, all malicious web pages that rely on JavaScript could not be detected.	Threat to external validity, false negatives.	Medium	High	Particular type of URLs are selected that exhibit different malicious behavior	Document configuration of operating system, browser and plug-ins used, and do not generalize beyond that configuration.

Figure C.2: HAZOP Analysis Worksheet Apparatus 2 of 2



ID	Guide word	General deviation	Specific deviation	Consequence	Severity	Likelihood	Possible causes	Action Required
1	NO	The artifact is not present.	DNS server not available.	Threat to external validity, False negatives.	Medium	High	Network connectivity; DNS server simply no longer exists	Retry host lookup x number of times; log if retries are unsuccessful.
2			HTTP server not available.	Threat to external validity, False negatives.	Medium	High	Network connectivity; HTTP server simply no longer exists	Retry host lookup x number of times; log if retries are unsuccessful.
3			Web page not available.	Threat to external validity, False negatives.	Medium	High	Web page no longer exists.	Log fact that malicious web page no longer exists.
4			Malicious web page's exploit is not present.	Threat to external validity, False negatives.	Medium	Medium	Anti-forensic techniques employed by malicious web page. This could stem from the fact that the web page somehow identified the client honeypot (either through location or how the request was made)	Immitate realistic setup and behavior of client honeypot; document client honeypot's configuration and behavior.
5	MORE	More artifacts are present than expected.	Client honeypot is exposed to more malicious web pages than expected.	Threat to external validity	Medium	Low	URL source delivers duplicates or pseudo-duplicates (e.g. URL that points to same page, but with different query parameters)	Remove duplicate URLs. Hash the content of the page and remove duplicates with same hash.
6	FEWER	Fewer artifacts are present than expected.	Client honeypot is exposed to less malicious web pages than expected.	Threat to external validity	Medium	Medium	Exploit hasn't been given the opportunity to trigger the attack because browser window was closed prematurely.	Add classification delay; calibrate classification delay to detect majority of malicious web pages
7	UNRELIABLE	Same artifacts and same activities but measurement of the same activities vary randomly.	Web page denoted by URL is sourced from different web servers each time a request is made.	Threat to external validity, False negatives.	Medium	Low	Web page denoted by URL is hosted on different web servers. This could be caused in legitimate circumstances by load balancing, and in	Determine if web page is part of a fast flux network and repeatedly visit such page if it is the case.
8	UNRELIABLE	Same artifacts and same activities but measurement of the same activities vary randomly.	Web page changes (e.g. malicious content is removed by webmaster)	Classification made may no longer be valid after short time has passed.	Low	Medium	Web page has changed (e.g. webmaster has removed exploit)	Record network data so a offline analysis of the web page can be conducted. Scan frequently, to obtain an accurate picture of the dynamic nature of malicious web pages.

Figure C.3: HAZOP Analysis Worksheet Subjects 1 of 1



ID	Guide word	General deviation	Specific deviation	Consequence	Severity	Likelihood	Possible causes	Action Required
1	NO	The artifact is not present.	The DNS lookup is not made (step 5)	URLs are not processed; no data is being collected	High	Medium	Software bug	Functional testing
2			The HTTP request is not made (step 4)	URLs are not processed; no data is being collected	High	Medium	Software bug	Functional testing
3	MORE	More artifacts are present than expected.	More malicious web pages are served than expected (step 6)	Threat to external validity	Medium	Low	Malicious web pages change as new vulnerabilities come available and user's browsing behavior changes. During particular times, e.g. when a zero-day vulnerability is discovered, the number of malicious web pages may temporarily rise.	Record time and duration of when data was collected and analyze for unusual spikes which could be mapped to external events. Normalize data accordingly.
4	FEWER	Fewer artifacts are present than expected.	Specific visitation algorithm leads to fewer responses by malicious web pages. (step 4)	Threat to external validity, False negatives.	Medium	Medium	Some malicious web pages only launch an attack on initial request, but not subsequent requests. If a visitation algorithm need to repeatedly visits a page, it may miss that page.	Select visitation algorithm that does not need repeated visitation of the page. If it does, use caching or distribute requests across a number of clients.
5	*UNRELIABLE*	Same artifacts and same activities but measurement of the same activities vary randomly.	DNS lookup fails intermittently. (step 5)	Some URLs are not processed; some data is being collected	Low	Medium	Connectivity issues on client side	Monitor connectivity and functionality of DNS server.
6	*UNRELIABLE*	Same artifacts and same activities but measurement of the same activities vary randomly.	HTTP request fails intermittently. (step 4)	Some URLs are not processed; some data is being collected	Low	Medium	Connectivity issues on client side	Monitor connectivity.
7	*BIASED*	Same artifacts and same activities but measurement of the same activities vary by a constant amount.	HTTP request leads to biased classification of malicious web pages. (step 4)	Threat to external validity, False negatives.	Medium	Medium	The IP source address of the client reveals the location of the clients, which may be incompatible to the location targeted by the malicious web page.	Make requests from client honeypot at different network locations. Document network locations used and do not generalize beyond the location.
8			HTTP request leads to biased classification of malicious web pages. (step 4)	Threat to external validity, False negatives.	Medium	Medium	Malicious web pages may only trigger during certain hours of their locale.	Make requests from client honeypot at different times.
9			HTTP request leads to biased classification of malicious web pages. (step 4)	Threat to external validity, False negatives.	Medium	Medium	Malicious web pages may only trigger for a particular locale.	Make requests from client honeypot at different locale. Document locale used and do not generalize beyond the location.
10	LATE	One or multiple steps of the activity occur late.	Host lookup occurs late (step 5)	Web page is not loaded and potentially not classified as malicious.		High	Network issues; intentional delay introduced by attacker.	Retry failed lookup x times; upon failure to lookup host after retries, log error.
11	HISTORY	Results change based on history.	Malicious web page exhibits different malicious behavior upon subsequent interaction.	Threat to external validity, False negatives.	Medium	High	Malicious web page implements anti-forensic measures. If visitation algorithm is used that repeatedly interacts with the page, it would not be correctly classified as malicious.	Select appropriate visitation algorithm and caching strategy. Record network traffic for later analysis.

Figure C.4: HAZOP Analysis Worksheet Stimuli 1 of 1

Appendix D

Examples of Malicious Web Pages

An in-depth analysis of a few select malicious web pages is presented in this appendix. During this analysis, malicious web pages were primarily visually inspected to identify clues that indicate a means of exploitation.

Exploits primarily target ActiveX components. These exploits usually instantiate a vulnerable ActiveX component and then use JavaScript to interact with the object to solicit an overflow vulnerability. This has increasingly been the case as components used by the operating system are patched through the automatic update functionality, whereas third-party components are excluded from this automatic update functionality and remain unpatched. The web exploitation kits seem to take advantage of this fact and the exploits they use primarily tackle ActiveX components. An analysis of the web exploitation kits WebAttacker, MPack, and Icepack showed that 10 out of 15 exploits for Internet Explorer target ActiveX components [125].

However, visually inspecting the malicious web pages identified did not reveal attacks on ActiveX components. While some pages, like `http://nzinfo.co.nz/`, did contain ActiveX components, it seems the ActiveX components were not directly involved in the attack, but rather existed to render rich content on the page. The analysis of several malicious web pages explains this apparent discrepancy.

VirtualMagic.co.nz

The first analysis is on the malicious web page at `http://virtualmagic.co.nz`. This web page was first classified as malicious in April 2008 and last classified as malicious in August 2008. The network trace used to analyze this page was from the malicious classification on June 24, 2008. As of February 2009, the web site no longer exists.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
        <title>Virtual Magic Limited homepage</title>
</head>
<body>
<script>var temp="",i,c=0,out="";var
str="60!83!99!114!105!112!116!32!76!97!110!103!117!97!103!101!61!39!74!97!118!97!83!99!114!105!11
2!116!39!62!32!10!102!117!110!99!116!105!111!110!32!107!97!115!112!101!114!115!107!121!40!115!117
!99!107!44!100!105!99!107!41!123!125!59!10!102!117!110!99!116!105!111!110!32!107!97!115!112!101!1
14!115!107!121!50!40!115!117!99!107!95!100!105!99!107!44!97!103!97!105!110!41!123!125!59!10!10!11
8!97!114!32!109!109!32!61!32!110!101!119!32!65
…
!114!116!40!41!32!123!10!32!105!102!32!40!33!32!77!68!65!67!40!41!32!41!32!123!10!32!115!116!97!1
14!116!79!118!101!114!102!108!111!119!40!48!41!59!32!10!125!32!10!125!10!115!116!97!114!116!40!41
!59!32!10!60!47!115!99!114!105!112!116!62!";l=str.length;while(c<=str.length-
1){while(str.charAt(c)!='!')temp=temp+str.charAt(c++);c++;out=out+String.fromCharCode(temp);temp=
"";}document.write(out);</script>
<img src="Black&#32;aluminium&#32;VM&#32;web3f&#32;copy.jpg" width="1024" height="768" alt=""
border="0">
</body>
</html>
```

Figure D.1: Virtualmagic.co.nz – Obfuscated Exploit

The visible content of the page is sparse. Besides a small black square, the page is empty. The abbreviated HTML code of the page is shown in Figure D.1. As shown, the exploit itself is obfuscated in which a custom JavaScript de-obfuscation function is used to convert the long string of ASCII values into the exploit code, which is then appended to the web page via the document.write function. Once this occurs, the exploit code can execute.

An abbreviated version of the exploit code is shown in Figure D.2. Once successfully triggered, it will push and execute the malware ldr.exe onto the end user's machine. The attack code is a multistep attack that first obtains the payload via the XMLHTTP object, writes it to disk via the ADODB (BID: 10514) object, and then executes it with the WScript.Shell or Shell.Application object (BID: 10652). The vulnerabilities targeted are all older vulnerabilities for which patches have been available.

```
<script>

…

var urlRealExe = 'http://www.rxpromotion.info/workspase/local/ldr.exe';

…

if (a) {
 if (! v[0]) {
 v[0] = CreateObject(a, "msxml2.XMLHTTP");
if (! v[0])
v[0] = CreateObject(a, "Microso"+"ft.XM"+"LHT"+"TP");
if (! v[0])
v[0] = CreateObject(a, "MSX"+"ML2.Se"+"rverXM"+"LHT"+"TP");}
if(! v[1]) {
v[1] = CreateObject(a, "ADODB.Str"+"eam"); }
if (! v[2]){
v[2] = CreateObject(a, "WSc"+"ript.Sh"+"ell");
if (! v[2]) {
v[2] = CreateObject(a, "Shel"+"l.Ap"+"pl"+"icati"+"on");
if (v[2])
n=1;

…

</script>
```

Figure D.2: Virtualmagic.co.nz – De-obfuscated Exploit

Virtualmagic.co.nz illustrates that exploit code can be hidden from plain view. The example shown here is obvious in that the long string of ASCII characters is an indication of some suspicious activity. Obfuscated exploit code, however, can be much more subtle or hidden on other pages that are loaded as part of the main web page (e.g., through external JavaScript code).

B-guided.co.nz

The second analysis is on the malicious web page at `http://b-guided.co.nz`. This web page was first classified as malicious in April 2008 and last classified as malicious in June 2008. The network trace used to analyze this page was from the malicious classification on April 14, 2008. As of February 2009, the web site exists, but appears to be no longer malicious. A screenshot of the web page is shown in Figure D.3.

B-guided.co.nz, a guide about New Zealand, is an example in which

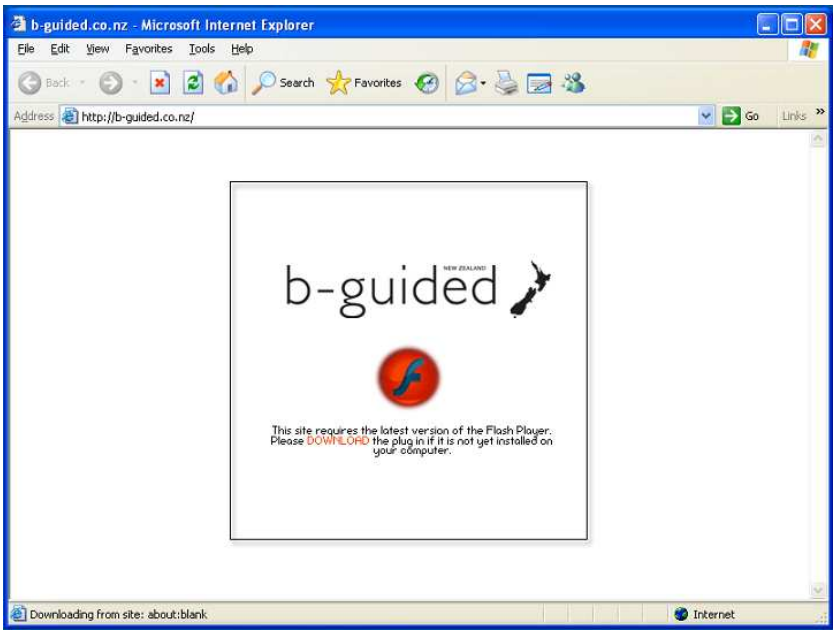


Figure D.3: B-guided.co.nz – Screenshot

the main page contains an exploit import statement. Instead of containing the exploit directly, either in clear or obfuscated form as shown above, an import statement is used to pull the exploit from a different server onto the page. These servers can be centralized exploit servers used by several thousands of web pages, which allow attackers to track exploitation across multiple pages and to update their exploit code easily to increase the effectiveness of their attacks.

```
<iframe src=http://google-analysis.com/in.cgi?9 width=1 height=1></iframe>
```

Figure D.4: B-guided.co.nz – Exploit Import

The exploit is pulled onto the page via a simple iFrame statement as shown in Figure D.4. When opening the page, the page `http://google-analysis.com/in.cgi?9` is opened, which contains the actual exploit. (This link does not point to the legitimate Google Analytics service [49].) The iFrame code itself might have been placed by the web site administrator in good

faith that this code is required to make use of the Google Analytics service. Alternatively, the code might have been placed there by a third party. Several posts on the Internet suggest the latter [152].

B-guided.co.nz illustrates that exploits do not necessarily need to be contained on the web page that is denoted by the URL. The exploit code can be imported onto the page from a different server.

Stargames.co.nz

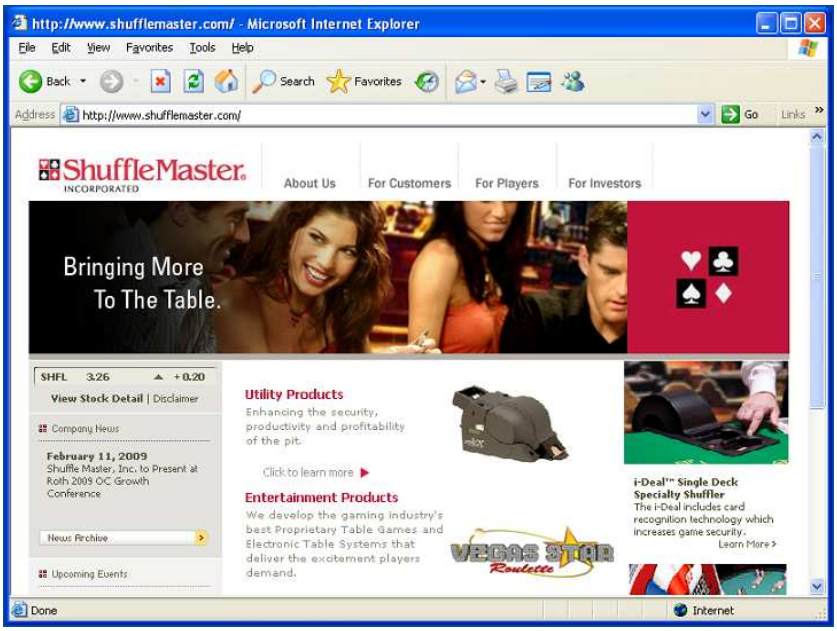


Figure D.5: Stargames.co.nz – Screenshot

The last analysis is on the malicious web page at `http://stargames.co.nz`. This web page was classified as malicious only in the month of July 2008. The network trace used to analyze this page was from the malicious classification on July 14, 2008. As of February 2009, the web site exists, but appears to be no longer malicious.

A screenshot of the web page is shown in Figure D.5. This screenshot shows that the stargames.co.nz web page is not hosted on the web

site stargames.co.nz. Instead, when navigating to stargames.co.nz, the browser is redirected via a server side 302 HTTP redirect to `http://www.shuffelmaster.com`, which is the page that contains the actual exploit.

This example illustrates again that web pages denoted by the URL might not contain the exploit directly. Rather, a redirect is used to have the browser navigate to a page that does. This redirect mechanism could be a 302 HTTP redirect as shown in our example or it could be, among other things, a more subtle client-side redirect, such as meta-refresh or setting the JavaScript location property. Redirects are used as a legitimate means to manage web page content. This particular web page seems to make use of redirects in a legitimate way and it appears that the redirect has not been a specific technique to import an exploit. However, redirects could certainly be used as a way to trick users into navigation of a malicious web page as illustrated by the abuse of the Google ”I feel lucky” feature [75].

Bibliography

[1] ALADDIN KNOWLEDGE SYSTEMS, LTD. Home page, 1985. Available from `http://www.aladdin.com`; accessed on 4 September 2008.

[2] ALADDIN KNOWLEDGE SYSTEMS, LTD. Aladdin eSafe CSRT 2005 malicious code report: The big threats shift, 2006.

[3] ALLEN, J., CHRISTIE, A., FITHEN, W., MCHUGH, J., PICKEL, J., AND STONER, E. State of the practice of intrusion detection technologies, 2000.

[4] AMAZON, INC. Amazon elastic compute cloud (Amazon EC2), 2006. Available from `http://aws.amazon.com/ec2/`; accessed on 10 November 2008.

[5] ANAGNOSTAKIS, K., SIDIROGLOU, S., AKRITIDIS, P., XINIDIS, K., MARKATOS, E., AND KEROMYTIS, A. Detecting targeted attacks using shadow honeypots. In *14th USENIX Security Symposium* (Baltimore, 2005), Usenix.

[6] ANTI-VIRUS COMPARATIVES. Anti-virus comparative August 2008, 2008. Available from `http://www.av-comparatives.org/seiten/ergebnisse_2008_08.php`; accessed 20 September 2008.

[7] ANUPAM, V., KRISTOL, D. M., AND MAYER, A. A user's and programmer's view of the new javascript security model. In *2nd*

USENIX Symposium on Internet Technologies and Systems (Boulder, 1999), Usenix.

[8] ANUPAM, V., AND MAYER, A. Security of web browser scripting languages: Vulnerabilities, attacks, and remedies. In *7th USENIX Security Symposium* (San Antonio, 1998), USENIX.

[9] ASADOORIAN, P. Web browser insecurity, 2005. Available from `http://www.sans.org/reading_room/whitepapers/application/1637.php`; accessed on 25 September 2006.

[10] AXELSSON, S. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *6th ACM Conference on Computer and Communications Security* (Singapore, 1999), ACM Press.

[11] BACE, R. *Intrusion Detection*. Macmillian Technical Publishing, Indianapolis, 2000.

[12] BAECHER, P., KOETTER, M., WICHERSKI, G., AND TILLMANN, W. Nepenthes - stats:scantest, 2008. Available from `http://nepenthes.mwcollect.org/stats:scannertest`; accessed on 20 September 2008.

[13] BAOFENG. Baofeng storm codec, 2008. Available from `http://www.baofeng.com`; accessed on 10 November 2008.

[14] BBC NEWS. Clipboards hijacked in web attack, 2008. Available from `http://news.bbc.co.uk/2/hi/technology/7567889.stm`; accessed on 20 August 2008.

[15] BERNERS-LEE, T. Naming and addressing: URIs, URLs, ..., 1993. Available from `http://www.w3.org/Addressing/`; accessed on 1 October 2006.

[16] BOSCOVICH, R., DCANAVOR, D., FAULHABER, J., GULLOTTO, V., JONES, J., LAMBERT, J., LAUDANSKI, P., MADOR, Z., MORDANI, R., O'DEA, H., PARTHASARATHY, S., PENTA, A., SEIFERT, C., SHOSTACK, A., STATHAKOPOULOS, G., STONE, A., THOMLINSON, M., WU, S., AND ZINK, T. Microsoft security intelligence report, 2009. Available from `http://go.microsoft.com/fwlink/?LinkId=147935`; accessed on 12 April 2009.

[17] BRANT, A., AND DAHL, E. New ad attacks, 2008. Available from `http://pcworld.about.com/magazine/2302p020id118781.htm`; accessed on 20 August 2008.

[18] BRUEMMER, P. J. Putting SEO in your dashboard, 2006. Available from `http://www.imediaconnection.com/content/9395.asp`; accessed on 10 October 2008.

[19] BUESCHER, A., MEIER, M., AND BENZMUELLER, R. Monkey-Wrench - boesartige webseiten in die zange genommen. In *11. Deutscher IT-Sicherheitskongress* (Bonn, 2009).

[20] CACHIN, C., DACIER, M., DAEAK, O., JUBLISCH, K., RANDELL, B., RIORDAN, J., TSCHARNER, A., WESPI, A., AND WUEST, C. Towards a taxonomy of intrusion detection systems and attacks, 2001.

[21] CARDENAS, A., BARAS, J. S., AND SEAMON, K. A framework for the evaluation of intrusion detection systems. In *IEEE Symposium on Security and Privacy* (Oakland, 2006), IEEE.

[22] CAVNAR, W., AND TRANKLE, J. N-gram-based text categorization. In *Third Annual Symposium on Document Analysis and Information Retrieval* (Las Vegas, 1994), UNLB Publications/Reprographics, pp. pp. 161–175.

[23] COLLIN, J., BORTZ, A., BONEH, D., AND MITCHELL, J. C. Same origin policy: Protecting browser state from web privacy attacks. In *WWW* (Edinburgh, 2006), ACM.

[24] DANCHEV, D. Fake celebrity video sites serving malware, 2008. Available from `http://ddanchev.blogspot.com/2008/06/fake-celebrity-video-sites-serving.html`; accessed on 10 August 2008.

[25] DANFORD, R. 2nd generation honeyclients. In *SANSFIRE* (Washington, D.C., 2006).

[26] DEBAR, H., DACIER, M., AND WESPI, A. Towards a taxonomy of intrusion-detection systems. *Computer Networks 31* (1999), 805–822.

[27] EGELE, M., WURZINGER, P., KRUEGEL, C., AND KIRDA, E. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks, 2009. Available from `http://www.iseclab.org/papers/driveby.pdf`; accessed on 15 May 2009.

[28] ENDICOTT-POPOVSKY, B., FRINCKE, D. A., AND TAYLOR, C. A. A theoretical framework for organizational network forensic readiness. *Journal of Computers 2*, 3 (2007), 11.

[29] ENDICOTT-POPOVSKY, B., NARVAEZ, J., SEIFERT, C., FRINCKE, D., O'NEIL, L. R., AND AVAL, C. Use of deception to improve client honeypot detection of drive-by-download attacks. In *5th International Conference on Augmented Cognition* (2009).

[30] ERNST, M. Self-defending software: Collaborative learning for security, 2008.

[31] EUROPEAN NETWORK OF AFFINED HONEYPOTS. D0.1: Survey of the state-of-the-art, 2005. Available from `http://www.fp6-noah.`

org/publications/deliverables/D0.1.pdf; accessed on 20 July 2006.

[32] EUROPEAN NETWORK OF AFFINED HONEYPOTS. D3.1b: Client-side honeypots, 2008.

[33] FEINSTEIN, B., AND PECK, D. Caffeine Monkey: Automated collection, detection and analysis of malicious JavaScript. In *Black Hat USA 2007* (Las Vegas, 2007).

[34] FELTEN, E. W., AND SCHNEIDER, M. A. Timing attacks on web privacy. In *CCS* (Athens, 2000), ACM.

[35] FILO, D., AND WANG, J. Yahoo! search engine, 1994. Available from http://www.yahoo.com/; accessed on 20 November 2007.

[36] FINJAN. Home page, 1996. Available from http://www.finjan.com; accessed on 4 September 2008.

[37] FINJAN. Web security trends report - Q1/2008, 2008. Available from http://www.finjan.com/GetObject.aspx?ObjId=563&Openform=50; accessed on 4 September 2008.

[38] FINJAN. Web security trends report - Q2/2008, 2008. Available from http://www.finjan.com/GetObject.aspx?ObjId=620&Openform=50; accessed on 4 September 2008.

[39] FINJAN. Web security trends report - Q2/2008, 2008. Available from http://www.finjan.com/GetObject.aspx?ObjId=620&Openform=50; accessed on 4 September 2008.

[40] FORD, S., COVA, M., KRUEGEL, C., AND VIGNA, G. Wepawet, 2008. Available from http://wepawet.cs.ucsb.edu/index.php; accessed on 20 July 2009.

[41] FOUNDATION, A. S. HttpClient, 2001. Available from `http://hc.apache.org/httpclient-3.x/`; accessed on 10 February 2009.

[42] FOXNEWS. CIA: Hackers shut down foreign power grid, 2008. Available from `http://www.foxnews.com/story/0,2933,324547,00.html`; accessed on 10 June 2008.

[43] FREI, S., DUEBENDORFER, T., OLLMAN, G., AND MAY, M. Understanding the web browser threat: Examination of vulnerable online web browser populations and the "insecurity iceberg", 2008. Available from `http://www.techzoom.net/papers/browser_insecurity_iceberg_2008.pdf`; accessed on 10 August 2008.

[44] GAFFNEY, J., AND ULVILA, J. Evaluation of intrusion detectors: A decision theory approach. In *IEEE Symposium on Seucirty and Privacy* (Oakland, 2001), IEEE, pp. 50–61.

[45] GARFINKEL, T., AND ROSENBLUM, M. A virtual machine introspection based architecture for intrusion detection. In *10th Annual Network and Distributed Systems Security Symposium* (San Diego, 2003), The Internet Society, pp. 191–206.

[46] GLEASON, C. Firetrust limited, 2007. Personal Communication.

[47] GOOGLE INC. Home page, 1996. Available from `http://www.google.com`; accessed on 5 September 2008.

[48] GOOGLE INC. Putting a stop to spyware, 2006. Available from `http://googleblog.blogspot.com/2006/01/putting-stop-to-spyware.html`; accessed on 5 September 2008.

[49] GOOGLE INC. Google Analytics, 2007. Available from `http://www.google.com/analytics`; accessed on 12 February 2009.

[50] GOOGLE INC. Google safe browsing API, 2007. Available from `http://code.google.com/apis/safebrowsing/`; accessed on 12 February 2008.

[51] GOOGLE INC. Chrome, 2008. Available from `http://www.google.com`; accessed on 5 September 2008.

[52] GORDON, L. A., LEOB, M. P., LUCYSHYN, W., AND RICHARDSON, R. CSI/FBI computer crime and security survey, 2006.

[53] GRAHAM, R. Sidejacking with hamster, 2007. Available from `http://erratasec.blogspot.com/2007/08/sidejacking-with-hamster_05.html`; accessed on 10 August 2008.

[54] GRANGER, S. Social engineering fundamentals, part I: Hacker tactics, 2001. Available from `http://www.securityfocus.com/infocus/1527`; accessed on 10 August 2008.

[55] GROSSMAN, J. I know where you've been, 2006. Available from `http://jeremiahgrossman.blogspot.com/2006/08/i-know-where-youve-been.html`; accessed on 10 August 2008.

[56] GROSSMAN, J., AND NIEDZIALKOWSKI, T. Hacking intranet websites from the outside. In *BlackHat* (2006, 2006).

[57] GU, G., FOGLA, P., DAGON, D., LEE, W., AND SKORIC, B. Measuring intrusion detection capability: An information-theoretic approach. In *ASIACCS* (Taipei, 2006).

[58] GULLI, A., AND SIGNORINI, A. The indexable web is more than 11.5 billion pages, 2005. Available from `http://www.cs.uiowa.edu/~asignori/web-size/`; accessed on 16 October 2006.

[59] GYONGYI, Z., AND GARCIA-MOLINA, H. Web spam taxonomy, 2004.

[60] HAUTESECURE. Home page, 2006. Available from `http://hautesecure.com`; accessed on 4 September 2008.

[61] HOFFMAN, B. Circumventing automated JavaScript analysis. In *Black Hat USA* (Las Vegas, 2008).

[62] HOLZ, T., GORECKI, C., RIECK, K., AND FREILING, F. Measuring and detecting fast-flux service networks. In *15th Annual Network & Distributed System Security Symposium* (San Diego, 2008).

[63] IKINCI, A., HOLZ, T., AND FREILING, F. Monkey-Spider: Detecting malicious websites with low-interaction honeyclients. In *Sicherheit* (Saarbruecken, 2008).

[64] JAKOBSSON, M., AND RAMZAN, Z. *Crimeware - Understanding New Attacks and Defenses*. Symantec Press, 2008.

[65] JAKOBSSON, M., AND STAMM, S. Invasive browser sniffing and countermeasures. In *WWW* (Edinburgh, 2006), ACM.

[66] JOSHI, A., KING, S., DUNLAP, G., AND CHEN, P. Detecting past and present intrusions through vulnerability-specific predicates. In *Symposium on Operating Systems Principles* (Brighton, 2005), ACM.

[67] KANICH, C., KREIBICH, C., LEVCHENKO, K., ENRIGHT, B., VOELKER, G., PAXSON, V., AND SAVAGE, S. Spamalytics: An empirical analysis of spam marketing conversion. In *CCS* (Alexandria, 2008), ACM.

[68] KAREN, P. M., AND ROMANOSKY, S. A complete guide to the common vulnerability scoring system, version 2.0, 2007. Available from `http://www.first.org/cvss/cvss-guide.pdf`; accessed on 11 August 2008.

[69] KIJEWSKI, P. Personal communication, 2008.

[70] KLETZ, T. A. *Hazop and Hazan: Identifying and Assessing Process Industry Hazards*, fourth edition ed. 1999.

[71] KOMISARCZUK, P., SEIFERT, C., PEMBERTON, D., AND WELCH, I. Grid enabled internet instruments. In *IEEE Global Communications Conference* (Washington DC, USA, 2007).

[72] KOUNS, J., SULLO, C., AND MARTIN, B. Open source vulnerability database, 2004. Available from `http://www.osvdb.org`; accessed on 1 October 2006.

[73] KRISTOL, D., AND MONTULLI, L. RFC2965 - HTTP state management mechanism, 2000. Available from `http://tools.ietf.org/html/rfc2965`; accessed on 20 November 2007.

[74] LAM, V., ANTONATOS, S., AKRITIDIS, P., AND ANAGNOSTAKIS, K. Puppetnets: Misusing web browsers as a distributed attack infrastructure. In *CCS* (Alexandria, 2006), ACM.

[75] LARKIN, E. Hackers rig Google to deliver malware, 2008. Available from `http://www.networkworld.com/news/2008/012808-google-hack.html`; accessed on 10 January 2009.

[76] LEMON, S. Mass SQL injection attack hits Chinese web sites, 2008. Available from `http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9086658`; accessed on 5 September 2008.

[77] LEVY, E. The making of a spam zombie army. dissecting the Sobig worms. *IEEE Security and Privacy 1*, 4 (2003), 58–59.

[78] LIPPMANN, R. P., FRIED, D. J., GRAF, I., HAINES, J. W., KENDALL, K. R., MCCLUNG, D., WEBER, D., WEBSTER, S. E., WYSCHOGROD,

D., CUNNINGHAM, R., AND ZISSMAN, M. A. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Proceedings of the DARPA Information Survivability Conference and Exposition* (Los Alam, 2000), IEEE Computer Society, pp. 12–26.

[79] MAVITUNA, F. BSQL hacker, 2008. Available from `http://labs.portcullis.co.uk/application/bsql-hacker/`; accessed on 5 September 2008.

[80] MAXION, R. A., AND ROBERTS, R. R. Proper use of ROC curves in intrusion/ anomaly detection, 2004. Available from `http://www.cs.newcastle.ac.uk/research/pubs/trs/papers/871.pdf`; accessed on 20 July 2006.

[81] MAXMIND. MaxMind GeoLite country, 2002. Available from `http://www.maxmind.com/app/geolitecountry`; accessed on 29 October 2007.

[82] MCAFEE, INC. Home page, 2005. Available from `http://www.siteadvisor.com/`; accessed on 4 September 2008.

[83] MCAFEE, INC. Mapping the mal web, revisited, 2008. Available from `http://us.mcafee.com/root/campaign.asp?cid=45044`; accessed on 10 August 2008.

[84] MCHUGH, J. Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln laboratory. In *ACM Transactions on Information and System Security* (2001), vol. 3, ACM, pp. 262–294.

[85] METASPLOIT. The Metasploit framework, version 2.6. Available from `http://www.metasploit.org`; accessed on 13 July 2006.

[86] MICROSOFT CORPORATION. Home page, 1975. Available from `http://www.microsoft.com`; accessed on 4 September 2008.

[87] MICROSOFT CORPORATION. ActiveX controls, 1996. Available from `http://msdn.microsoft.com/library/default.asp?url=/workshop/components/activex/activex_node_entry.asp`; accessed on 10 August 2006.

[88] MICROSOFT CORPORATION. Internet Explorer, 2006. Available from `http://www.microsoft.com/windows/ie/default.mspx`; accessed on 10 September 2006.

[89] MICROSOFT CORPORATION. Microsoft security bulletin MS06-014: Vulnerability in the microsoft data access components (MDAC) function could allow code execution (911562), 2006. Available from `http://www.microsoft.com/technet/security/Bulletin/MS06-014.mspx`; accessed on 14 February 2007.

[90] MICROSOFT CORPORATION. Microsoft security intelligence report, 2008. Available from `http://www.microsoft.com/security/portal/sir.aspx`; accessed on 4 September 2008.

[91] MIMOSO, M. S. SQL injection attack infects hundreds of thousands of websites, 2008. Available from `http://searchsecurity.techtarget.com/news/article/0,289142,sid14_gci1311815,00.html`; accessed on 5 September 2008.

[92] MIRKOVIC, J., DIETRICH, S., DITTRICH, D., AND REIHER, P. *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Series in Computer Networking and Security)*. Prentice Hall, 2005.

[93] MOESTL, T., AND ROMBOUTS, P. Pdnsd - proxy DNS server, 2000. Available from `http://www.phys.uu.nl/~rombouts/pdnsd/index.html`; accessed on 24 September 2007.

[94] MOORE, H. Internet drive-by shootings, 2006. Available from `http://metasploit.blogspot.com/2006/07/`

`internet-drive-by-shootings.html`; accessed on 1 October 2006.

[95] MOSHCHUK, A., BRAGIN, T., DEVILLE, D., GRIBBLE, S. D., AND LEVY, H. M. SpyProxy: execution-based detection of malicious web content. In *16th USENIX Security Symposium on USENIX Security Symposium* (Boston, 2007), ACM.

[96] MOSHCHUK, A., BRAGIN, T., GRIBBLE, S. D., AND LEVY, H. M. A crawler-based study of spyware on the web. In *13th Annual Network and Distributed System Security Symposium* (San Diego, 2006), The Internet Society.

[97] MOZILLA CORPORATION. Firefox 3.0, 2008.

[98] MVPS.ORG. Blocking unwanted parasites with a hosts file, 1997. Available from `http://www.mvps.org/winhelp2002/hosts.htm`; accessed on 13 Janurary 2008.

[99] NAZARIO, J. PhoneyC: A virtual client honeypot. In *2nd Usenix Workshop on Large-Scale Exploits and Emergent Threats* (Boston, 2009), vol. 2007, Usenix.

[100] NETCRAFT. July 2008 web server survey, 2008. Available from `http://news.netcraft.com/archives/2008/07/07/july_2008_web_server_survey.html`; accessed on 10 August 2008.

[101] NETSCAPE COMMUNICATIONS CORPORATION. The same origin policy, 1998. Available from `http://www.mozilla.org/projects/security/components/same-origin.html`; accessed on 10 August 2008.

[102] OPERA SOFTWARE. Opera web browser v9.5, 2008. Available from `http://www.opera.com/index.dml`; accessed on 1 October 2008.

[103] OSWALD, D. HTML parser, 2002. Available from `http://sourceforge.net/projects/htmlparser/`; accessed on 10 February 2009.

[104] PERRY, M. 365-day: Active https cookie hijacking. In *Defcon* (Las Vegas, 2008). Available from `https://www.defcon.org/html/defcon-16/dc-16-speakers.html#Perry`; accessed on 20 August 2008.

[105] PETKOV, P. D. JavaScript port scanner, 2006. Available from `http://www.gnucitizen.org/projects/javascript-port-scanner/`; accessed on 10 August 2008.

[106] POUGET, F., AND HOLZ, T. A pointillist approach for comparing honeypots, 2005. This paper compares low interaction and high interaction honeypots.

[107] PROVOS, N. Honeyd virtual honeypot. Available from `http://www.honeyd.org/`; accessed on 6 July 2006.

[108] PROVOS, N. SpyBye, 2007. Available from `http://spybye.org/index.php?/archives/10-SpyBye-0.3-released.html`; accessed on 10 October 2008.

[109] PROVOS, N., AND HOLZ, T. Client honeypots. In *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison Wesley Professional, Upper Saddle River, NJ, 2007, pp. 231–272.

[110] PROVOS, N., MAVROMMATIS, P., RAJAB, M. A., AND MONROSE, F. All your iFRAMEs point to us, 2008. Available from `http://googleonlinesecurity.blogspot.com/2008/02/all-your-iframe-are-point-to-us.html`; accessed on 15 Feburary 2008.

[111] PROVOS, N., MCNAMEE, D., MAVROMMATIS, P., WANG, K., AND MODADUGU, N. The ghost in the browser: Analysis of web-based malware. In *HotBots'07* (Cambridge, 2007), Usenix.

[112] RAFAIL, J. Cross-site scripting vulnerabilities, 2001. Available from `http://www.cert.org/archive/pdf/cross_site_scripting.pdf`; accessed on 10 August 2008.

[113] RAFF, A. Cross-X scripting and redirecting to local resources, 2007. Available from `http://aviv.raffon.net/2007/03/10/CrossXScriptingAndRedirectingToLocalResources.aspx`; accessed on 10 August 2008.

[114] REIS, C., DUNAGAN, J., WANG, H. J., DUBROVSKY, O., AND ESMEIR, S. BrowserShield: Vulnerability-driven filtering of dynamic HTML. In *7th USENIX Symposium on Operating Systems Design and Implementation* (Seattle, 2006), Usenix.

[115] RICHARDSON, R. CSI/FBI computer crime and security survey, 2008. Available from `http://i.cmpnet.com/v2.gocsi.com/pdf/CSISurvey2007.pdf`; accessed on 10 August 2008.

[116] ROCASPANA, J. R. SHELIA: A client honeypot for client-side attack detection, 2007. Available from `http://www.cs.vu.nl/~herbertb/misc/shelia/shelia07.pdf`; accessed on 10 October 2008.

[117] RODRIGUEZ, S. Clipboard exploit, 2003. Available from `http://www.arstdesign.com/articles/clipboardexploit.html`; accessed on 10 August 2008.

[118] ROESCH, M. Snort-lightweight intrusion detection for networks. In *13th Large Systems Administration Conference* (Seattle, 1999), Usenix, pp. 229–238.

[119] RUEF, M. browserrecon project, 2008. Available from `http://www.computec.ch/projekte/browserrecon/`; accessed on 20 August 2008.

[120] RVDH. Internet Explorer 7 header forwards, 2007. Available from `http://www.0x000000.com/?i=547`; accessed on 10 August 2008.

[121] SANS INSTITUTE. SQL injection attack infects thousands of websites (January 7 and 8, 2008), 2008. Available from `http://www.sans.org/newsletters/newsbites/newsbites.php?vol=10&issue=2&portal=18568e8354c1477939922bd793b68360#sID200`; accessed on 5 September 2008.

[122] SCANSAFE. Home page, 1999. Available from `http://www.scansafe.com`; accessed on 4 September 2008.

[123] SCANSAFE. Global threat report, 2008. Available from `http://www.scansafe.com/__data/assets/pdf_file/8277/gtr_June2008.pdf`; accessed on 4 September 2008.

[124] SECURITYFOCUS. Vulnerabilties, 2008. Available from `http://www.securityfocus.com/vulnerabilities`; accessed on 10 August 2008.

[125] SEIFERT, C. Know your enemy: Behind the scenes of malicious web servers, 2007. Available from `http:/www.honeynet.org/papers/wek`; accessed on 7 November 2007.

[126] SEIFERT, C. Live Search: Battling the plague of the web, 2008. Available from `http://blogs.msdn.com/livesearch/archive/2008/12/02/battling-the-plague-of-the-web.aspx`; accessed on 5 January 2009.

[127] SEIFERT, C., WELCH, I., KOMISARCZUK, P., AND NARVAEZ, J. Drive-by-downloads, February 2008 2008. Available from `http://www.mcs.vuw.ac.nz/comp/Publications/index-byyear-08.html`; accessed on 01 February 2008.

[128] SIDIROGLOU, S., IOANNIDIS, J., KEROMYTIS, A., AND STOLFO, S. J. An email worm vaccine architecture. In *1st Information Security Practice and Experience Conference* (Singapore, 2005).

[129] SKOUDIS, E., AND ZELTSER, L. *Malware: Fighting Malicious Code*. Prentice Hall, 2003.

[130] SOPHOS. Home page, 1997. Available from `http://www.sophos.com`; accessed on 4 September 2008.

[131] SOPHOS. Sophos threat report july 2008, 2008. Available from `http://www.sophos.com/securityreportjul2008`; accessed on4 September 2008.

[132] SPI LABS. Detecting, analyzing, and exploiting intranet applications using JavaScript, 2006. Available from `http://www.spidynamics.com/assets/documents/JSportscan.pdf`; accessed on 10 June 2007.

[133] SPITZNER, L. *Honeypots: Tracking Hackers*. Addison-Wesley, Boston, 2002.

[134] SPITZNER, L. Honeypots mailing list post: 'SF new column announcement: Time to dump IE', 2004. Available from `http://www.securityfocus.com/archive/119/366293/30/1410/threaded`; accessed 20 September 2008.

[135] SPOOR, R. J., KIJEWSKI, P., AND OVERES, C. The HoneySpider network: Fighting client-side threats. In *First* (Vancouver, 2008).

[136] STANIFORD, S., PAXSON, V., AND WEAVER, N. How to 0wn the Internet in your spare time. In *11th USENIX Security Symposium* (San Francisco, 2002), Usenix.

[137] STIRLING, D., WELCH, I., AND KOMISARCZUK, P. Designing workflows for grid enabled internet instruments. In *Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)* (Lyon, 2008).

[138] STOLFO, S. J., FAN, W., LEE, W., PRODROMIDIS, A., AND CHAN, P. K. Cost-based modeling for fraud and intrusion detection: Results from the jam project. In *DISCEX* (Hilton Head, 2000), pp. 130–144.

[139] STOLL, C. Stalking the wily hacker. *Communications of the ACM 31*, 5 (1988), 484–497.

[140] STONE-GROSS, B., COVA, M., CAVALLARO, L., GILBERT, B., SZYDLOWSKI, M., KEMMERER, R., KRUEGEL, C., AND VIGNA, G. Your botnet is my botnet: Analysis of a botnet takeover. Tech. rep., UCSB Technical Report, April 2009.

[141] STOPBADWARE.ORG, . Badware websites report 2008, 2008. Available from `http://www.stopbadware.org/pdf/StopBadware_Infected_Sites_Report_062408.pdf`; accessed on 5 September 2008.

[142] STUURMAN, T., AND VERDUIN, A. Honeyclients - low interaction detection methods. Tech. rep., University of Amsterdam, 2008.

[143] SUNBELT SOFTWARE USA, . Home page, 1994. Available from `http://www.sunbeltsoftware.com`; accessed on 4 September 2004.

[144] SZOR, P. *The Art of Computer Virus Research and Defense*. Addison-Wesley Professional, 2005.

[145] THE HONEYNET PROJECT. Honeywall CDROM Eyeore, 2003. Available from `http://project.honeynet.org/tools/cdrom/eeyore/download.html`; accessed 6 July 2006.

[146] THE HONEYNET PROJECT. Know your enemy: Fast-flux service networks, 2007. Available from `http://www.honeynet.org/papers/ff/`; accessed on 25 September 2007.

[147] THE MITRE CORPORATION. CVE - common vulnerabilities and exposures (CVE), 1999. Available from `http://cve.mitre.org/index.html`; accessed on 20 August 2008.

[148] THE MITRE CORPORATION. Insecure method vulnerability in Sina Inc. DLoader Class ActiveX control (CVE-2008-6442), 2008. Available from `http://cve.mitre.org/index.html`; accessed on 20 March 2009.

[149] THE TECHWEB NETWORK. drive-by download definition: TechEncyclopedia from TechWeb, 2008. Available from `http://www.techweb.com/encyclopedia/defineterm.jhtml?term=drive-by+download`; accessed on 10 August 2008.

[150] THOMAS, R., AND MARTIN, J. The underground economy: Priceless. *;Login December 2006* (2006).

[151] TOMBINI, E., DEBAR, H., ME, L., AND DUCASSE, M. A serial combination of anomaly and misuse idses applied tohttp traffic. In *20th Annual Computer Security Applications Conference* (Tucson, 2004), IEEE, pp. 428–437.

[152] TUNG, L. Google: Garn, ill swap ya privacy for security, 2008. Available from `http://www.zdnet.com.au/blogs/securifythis/soa/Google-G-arn-I-ll-swap-ya-privacy-for-security/0,139033343,339286907,00.htm`; accessed on 1 Feburary 2009.

[153] VAN NOORD, G. TextCat language guesser, 1994. Available from `http://www.let.rug.nl/~vannoord/TextCat/`; accessed on 13 October 2007.

[154] W3C WORLD WIDE WEB CONSORTIUM. Hypertext markup language (HTML) home page, 1995. Available from `http://www.w3.org/MarkUp/`; accessed on 1 October 2006.

[155] W3SCHOOLS. Browser statistics, 1999. Available from `http://www.w3schools.com/browsers/browsers_stats.asp`; accessed on 14 September 2006.

[156] WANG, H. J., GUO, C., SIMON, D. R., AND ZUGENMAIER, A. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. In *SIGCOMM* (Portland, 2004), ACM, pp. 193–204.

[157] WANG, K. MITRE HoneyClient, 2005. Available from `http://www.honeyclient.org/trac`; accessed on 2 Janurary 2007.

[158] WANG, Y.-M. Personal communication, 2006.

[159] WANG, Y.-M., BECK, D., JIANG, X., ROUSSEV, R., VERBOWSKI, C., CHEN, S., AND KING, S. Automated web patrol with Strider HoneyMonkeys: Finding web sites that exploit browser vulnerabilities. In *13th Annual Network and Distributed System Security Symposium* (San Diego, 2006), Internet Society.

[160] WATSON, D. Global distributed honeynet, phase II, 2008. Personal Communication.

[161] WATSON, D., HOLZ, T., AND MUELLER, S. Know your enemy: Phishing, 2005. Available from `http://www.honeynet.org/papers/phishing/`; accessed on 10 August 2008.

[162] WEBROOT SOFTWARE INC. Home page, 1997. Available from `http://www.webroot.com/`; accessed on 4 September 2008.

[163] WEBSENSE INC. Home page, 1994. Available from `http://www.websense.com`; accessed on 4 September 2008.

[164] WEBSENSE INC. State of Internet security, Q1 - Q2, 2008, 2008. Available from `http://www.websense.com/securitylabs/docs/WSL_Report_1H08_FINAL.pdf`; accessed on 4 September 2008.

[165] WEBSENSE INC. Wget denied, 2008. Available from `http://securitylabs.websense.com/content/Blogs/3183.aspx`; accessed on 25 September 2008.

[166] WELCH, I., AND MAXION, R. The application of hazop to experimental design. Tech. rep., Victoria University of Wellington, 2009.

[167] WELLS, J. Brief history of computer viruses, 1996. Available from `http://www.research.ibm.com/antivirus/timeline.htm`; accessed on 17 October 2006.

[168] WESSELS, D., NORDSTROEM, H., ROUSSKOV, A., CHADD, A., COLLINS, R., SERASSIO, G., WILTON, S., AND FRANCESCO, C. Squid web proxy cache, 1996. Available from `http://www.squid-cache.org`; accessed on 25 September 2007.

[169] WIKIPEDIA - THE FREE ENCYCLOPEDIA. Client honeypot, 2006. Available from `http://en.wikipedia.org/wiki/Honeyclient`; accessed on 10 October 2008.

[170] WITTEN, I. H., AND FRANK, E. *Data Mining: Practical machine learning tools and techniques*, 2nd ed. Morgan Kaufmann, San Francisco, 2005.

[171] XIE, M., WU, Z., AND WANG, H. HoneyIM: Fast detection and suppression of instant messaging malware in enterprise-like networks. In *Computer Security Applications Conference* (Miami Beach, 2007), IEEE, pp. 64–73.

[172] YAHOO! INC. A safer way to search, 2008. Available from `http://www.ysearchblog.com/archives/000578.html`; accessed on 14 September 2008.

[173] YUAN, B., AND HOLZ, T. Client-side honeypots, 2007. Thesis. Rheinisch-Westflischen Technischen Hochschule Aachen.

[174] ZHUGE, J., HOLZ, T., SONG, C., GUO, J., HAN, X., AND ZOU, W. Studying malicious websites and the underground economy on the chinese web. Tech. rep., University of Mannheim, 2007.