ABSTRACT

Title of Document:	EMPIRICAL STUDIES BASED ON HONEYPOTS FOR CHARACTERIZING ATTACKERS BEHAVIOR
	Bertrand Sobesto, Doctor of Philosophy, 2015
Directed By:	Associate Professor Michel Cukier Reliability Engineering Program

The cybersecurity community has made substantial efforts to understand and mitigate security flaws in information systems. Oftentimes when a compromise is discovered, it is difficult to identify the actions performed by an attacker.

In this study, we explore the compromise phase, i.e., when an attacker exploits the host he/she gained access to using a vulnerability exposed by an information system. More specifically, we look at the main actions performed during the compromise and the factors deterring the attackers from exploiting the compromised systems.

Because of the lack of security datasets on compromised systems, we need to deploy systems to more adequately study attackers and the different techniques they employ to compromise computer. Security researchers employ target computers, called honeypots, that are not used by normal or authorized users.

In this study we first describe the distributed honeypot network architecture deployed at the University of Maryland and the different honeypot-based experiments enabling the data collection required to conduct the studies on attackers' behavior.

In a first experiment we explore the attackers' skill levels and the purpose of the malicious software installed on the honeypots. We determined the relative skill levels of the attackers and classified the different software installed.

We then focused on the crimes committed by the attackers, i.e., the attacks launched from the honeypots by the attackers. We defined the different computer crimes observed (e.g., brute-force attacks and denial of service attacks) and their characteristics (whether they were coordinated and/or destructive). We looked at the impact of computer resources restrictions on the crimes and then, at the deterrent effect of warning and surveillance. Lastly, we used different metrics related to the attack sessions to investigate the impact of surveillance on the attackers based on their country of origin.

During attacks, we found that attackers mainly installed IRC-based bot tools and sometimes shared their honeypot access. From the analysis on crimes, it appears that deterrence does not work; we showed attackers seem to favor certain computer resources. Lastly, we observed that the presence of surveillance had no significant impact on the attack sessions, however surveillance altered the behavior originating from a few countries.

EMPIRICAL STUDIES BASED ON HONEYPOTS FOR CHARACTERIZING ATTACKERS BEHAVIOR

By

Bertrand Sobesto

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Acknowledgment

I have met many incredible people over the past six years who have contributed to this significant personal and professional achievement, and I would like to thank all of them.

First, I would like to thank my committee members Monifa Vaughn-Cooke, Tudor Dumitras, Jeffrey Herrmann, and Atif Memon for their insightful and valuable feedback.

I am very thankful to my advisor, Michel Cukier, who gave me the opportunity to study and teach at the University of Maryland. This dissertation would not have been possible without his guidance and tutelage. I am also grateful to Paige Smith for all of the time spent revising and offering extremely valuable suggestions to improve this work.

I had the opportunity to collaborate with several different research teams over the course of my graduate studies. I would like to thank Gregg Vesonder, Dave Kormann and Matti Hiltunen from AT&T Labs Research. Working at Florham Park was a great experience. It was also a great opportunity to work with Ilir Gashi from the City University in London.

I would like to express my gratitude to Gerry Sneeringer, our fearless leader, and the IT Security group: Amy, Erin, Huifang, Lauren, Avery, Jonas, Kevin, Rob, and Steve, thank you for the great and varied conversations. I am really excited to start a new chapter of my life working with all of you.

On a more personal note, moving to a foreign country was not always easy, but I was lucky to build many great friendships along the way: Amy, Dr. Jan, Chaitrali, Cristel, Danielle, Melanie, Nanci, Jamie, Angelo, Asgeir, Colin, Cory, John, Sean, Robin, Sébastien, Teddy and Vivien.

I am also grateful for the support I have received from my friends back in France. For that, I would like to recognize Audrey, Agnès & Xavier, Réjane & Guillaume, Valentine & Matthieu, Fabrice, Guillaume, Renaud and Vincent.

I would like to thank Denise, Mary Anne and Paul for the great conversations over Thanksgiving dinner, their advice and their encouragements.

This journey would not have been possible without the support of my family, especially my mother, Fanny. You have always been there when I needed you. I admire how strong you are; you taught me to never give up, and it is thanks to you that I have made it to where I am now.

Table of Contents

List of Tables

List of Figures

Chapter 1 – Introduction

Whether we like it or not, parts of our everyday lives are stored on computer systems somewhere on the Internet. Today's society heavily relies on information systems and the Internet. Interconnecting computer networks removed country-boundaries and made worldwide communications faster.

Information Technology (IT) is omnipresent, a wide variety of so called "smart" devices and "smart" appliances are now connected to the Internet and offer new services designed to make our lives easier. Owners of connected cars are able, for example, to get real-time traffic information and remotely access some of the car's functions such as starting the engine. Connected cars can also automatically call rescue services in case of an accident. Our homes are also connected, thus allowing us to monitor them when at work, turn on the air conditioning or lights remotely, or alerting us in case of flood or break-in. We are in the era of the technologically connected life.

The increased integration of computer systems into our lives and the respective data they store make information systems extremely valuable. Any disruption to these systems can cause financial loss and have dramatic consequences for society at large. Further, these information systems are becoming more complex and more exposed, thus increasing their vulnerability. Their value and inherent vulnerability make information systems advantageous targets for attackers. They have become the victim of computer-focused crimes [FUR02].

To more adequately secure computer systems from external threats, security researchers aim to understand attackers and the different techniques they employ to compromise computers and to achieve their goals. One fruitful approach is to use a target computer --called a honeypot-- that is not used by normal or authorized users. As general users do not access these computers, all activity towards them can be identified as malicious.

1.1 Research Goals and Questions

The cybersecurity community has made extensive efforts to identify and mitigate security flaws in information systems. Companies such as Microsoft or Google offer a variety of monetary rewards to anyone who identifies security flaws in their software and websites. Once a compromise is discovered, it is often difficult to identify the actions performed by an attacker on a system mostly because of a lack of monitoring.

The main goal of this dissertation is to study the attackers' behavior during the compromise phase, i.e., when an attacker has already exploited a security flaw and gained access to a computer system. The overarching research questions are:

1) *What are the main actions performed during the compromise phase?*

We hypothesize that an attacker will want to exploit the resources offered by the compromised system to engage in criminal activities on the Internet. We believe an attacker will install an attack tool and launch crimes from a compromised system.

2) *Are there factors deterring the attackers from exploiting the compromised systems?*

We believe that the attackers will want to maintain their access to compromised systems by hiding their attack sessions. Therefore actively monitored systems will be less likely to be exploited.

To answer these research questions, we need to be able to identify 1) when the compromise occurs, 2) the different actions performed by the attacker during the different attack sessions, i.e., when an attacker connects to the compromised system to exploit it, and 3) the different criminal activities launched from the compromised system.

We could analyze existing security information on compromised systems from an organization such as the University of Maryland, but often these datasets are not always available because of privacy concerns as well as capacity issues. Systems are not monitored as closely we need them to be in order to conduct analyzes.

Another issue with institutional security datasets are the non-homegeneous nature of the computing environment. Different operating systems (OS) can be deployed, different software packages and services can be installed from one computer to another. As a consequence different populations of attackers can exploit different vulnerabilities and interfere with the factors being investigated.

In order to conduct studies on attacker behavior during the compromise phase, we need to design a homogeneous, and controlled environment. This environment enables the required experiments designed to collect the required datasets.

To address the two global research questions, we designed three experiments, each building on the lessons learned from the previous experiments.

1.1.1 Experiment 1: An empirical study on attacks and attackers

In the first experiment, described in Chapter 4, we needed to understand the nature of attackers' skills, if we could differentiate attackers using a compromised honeypot, identify the purpose of the attack and determine if attackers share their access to compromised honeypot with other attackers. In other words, the research questions associated with the first experiment are:

a) *How can we differentiate attackers on a compromised honeypot?*

b) *How skilled are the attackers based on the observations of attack sessions?*

c) *What is the purpose of the compromise?*

d) *Do attackers share the access to their honeypots?*

Question a: While determining who launches a cyber-crime is challenging, once an attacker logs into a compromised honeypot, the keystrokes and attack types can be used to characterize the crimes. This information can be used to determine identify if attackers who gain entry via brute force attacks are the same as those who log in by matching the IP addresses. In addition, we can determine if a single attacker uses several IP addresses to log-in or if multiple attackers are sharing a single compromised honeypot.

Question b: Attackers may structure their attack such that they can control the compromised system remotely. They may connect to an Internet Relay Chat or they may create a backdoor port to create the remote access. We hypothesize that in either scenario skilled attackers will attempt to remove network restrictions. Therefore if we add network restrictions to our honeypots, we will attract skilled attackers.

To assess attacker skill, we developed a skills-based profile based on ten criteria. We will explore the connection between the attacker’s skills based on the demonstrated execution of these ten criteria. We will then explore the relationship between the demonstrated skills and the acker’s ability to successfully launch an attack to commit crimes.

Question c: In order to understand why attacks are launched, we categorized them into different types of attacks based on the type of rogue software that is installed and the nature of the exploitation that results. We hypothesize that the honeypots will become a bot and join a botnet, a network of compromised hosts. The most commonly used protocol to control each bot is the Internet Relay Chat protocol (IRC) [RAT13]. We expect the attackers to install IRC-based bot tools.

Question d: The authors in [FRA07] showed the existence of an underground market for compromised hosts. We expect the initial attacker who found the correct username and password of the honeypots, to share the credentials. We hypothesize that some of our honeypots will be accessed by different attackers.

1.1.2 Experiment 2: Are Computer Focused Crimes Impacted by System Configuration

In Chapter 4 we realized that an attacker's reaction to honeypot configuration is an important aspect of better understanding an attacker. In the second experiment, described in Chapter 5, we provide several honeypot configurations to assess the attacker's reaction to various configurations.

The research questions for the second experiment focus on the nature of the crime as they related to various honeypot configurations:

a) *Does the presence of a warning banner effectively deter attackers from launching an attack from a compromised computer?*

b) *Does the computer configuration impact whether the crime is destructive or not?*

c) *Does the computer configuration impact whether the target is of choice or opportunity?*

d) *Does the computer configuration impact whether the attack is coordinated or not?*

This study will only include attackers who gain access to the honeypots. We will focus on three specific dimensions of computer crimes: destructiveness, nature of the target, and level of coordination. These aspects will be considered while we test the usefulness of a banner as a deterrence mechanism and if the configuration of the computer is related to the type of crime committed.

Question a: Because human attackers have rational decision-making processes that could prevent them from engaging in criminal online activities [PNG09], we hypothesize that *attackers will be deterred by a banner and as a consequence there will be fewer crimes launched from honeypots that have a warning banner.*

Question b-c: The various computer configurations that may be more or less attractive to an attacker are memory, disk space and bandwidth. We hypothesize that attackers will launch different attacks based on the available resources; with more attacks expected from configurations with high levels of resources (large memory and disk space and high bandwidth). Specifically we expect large memory space, large disk space and high bandwidth:

- to be associated with destructive activity;
- to be associated with targets of choice; and
- to be associated with coordinated attacks.

1.1.3 Experiment 3: Are Computer Focused Crimes Impacted by Surveillance

In Chapter 6, we build on the work of the second experiment to add surveillance as a deterrence system. In the third experiment, we manipulated two aspects of surveillance: the presence of a warning banning announcing the presence of a surveillance system (banner present/not present) and embedded surveillance tools on the computer (present/not present).

The research questions for the third experiment is:

> *Question: Are computer focused crimes (after an attacker gains unauthorized access to a computer and uses the computer to launch an attack towards an external target) impacted by a surveillance warning banner and/or surveillance tools?*

The security community consistently claims that deterrence does not work on the Internet. This claim is rarely accompanied by empirical data to support it. This presents an opportunity to investigate of the claim directly by focusing on the announcement and existence of surveillance mechanisms on a compromised computer system. A more focused study on employee awareness of surveillance on their work computer reduced the intent of system misuse by those employees [ARC09]. Based on this result, *we hypothesize that a banner announcing surveillance and a surveillance tool will discourage attackers from using the compromised system to launch crimes.*

1.1.4 Experiment 4: Effects of Banners on the Commands Typed by Attackers: A Study of Differences across Countries

In Chapter 7, we build on the work of the second and third experiments studying the effectiveness of banners to look at the differences in commands that are typed based on the attacker's country of origin.

In the fourth study we will empirically study the following research questions:

a) *Is there a variation in the probability that an attacker will enter commands depending on the country of origin from which an attack is launched?*

b) *In the presence of a surveillance banner will attackers vary in their use of system activity commands based on their country of origin (as identified by the IP address)?*

Question a: Attackers from different countries may respond differently to a surveillance banner as factors such as cultural differences existing across nations, as well as differential assessments regarding the likelihood of punishment may result in varying responses to a particular sanction threat. While in one country, the presence of a surveillance banner may be a valid signal of a threat, and accordingly, attackers originating from that country would internalize a heightened risk of apprehension and punishment, reducing their adverse behavior in response. However, in other countries the attackers may be aware that no punishment is actually associated with an attack. Furthermore, in other countries, the insinuation of a threat made via a banner may elicit defiance, thus increasing the likelihood of an attack. *Therefore we hypothesize that we will see differences in the subsequent actions of an attacker based on their country of origin in the presence of a surveillance banner.*

Question b: Despites the anonymity offered by the Internet, human attackers exhibit a rational decision-making process [PNG09]. They will attempt to maximize rewards while minimizing the risks of being detected. We hypothesize that *U.S. attackers, which are more easily identifiable and prosecutable, will be deterred by surveillance.* More specifically, *U.S. attackers understanding a surveillance announcement banner would look for effective surveillance cues including monitoring tools*. Therefore the actions after seeing a banner will be related to searching for surveillance cues before moving on to actions related to other aspects of criminal activity.

1.2 Approach and Structure

This dissertation addresses the questions described in Section 1.1 Chapter 2 provides the background literature review which motivated the research questions and hypotheses. Chapter 3 introduces the testbed built to collect the data required for empirical studies based on honeypots. More specifically we describe:

1) the distributed honeypot network architecture deployed at the University of Maryland (UMD), which includes how data is being collected, and how this architecture is being monitored.
2) the framework allowing the deployment of large numbers of honeypots.

The honeypot-based framework, introduced in Chapter 3 was developed to support different honeypot-based experiments, aimed at understanding the attackers' behavior on a system following a compromise. These experiments contributed toward characterizing the attackers' behavior and respective attacks launched against honeypots.

In Chapter 4, we determine the relative skill levels of the attackers according to a set of ten criteria (e.g., the ability to hide their malicious activity, the appropriateness of the command typed, and the familiarity with the rogue software installed). We also classify the different malicious software uploaded and installed on the compromised honeypots according to the software's identified purpose.

Chapters 5 and 6 are more focused on the crimes committed by the honeypots following a "successful" compromise. We define the different computer crimes observed (e.g., brute-force attacks and denial of service attacks) and their characteristics (whether they were coordinated and/or destructive). Different stimuli such as a banner, surveillance processes, and pertinent system configurations were systematically modified within a randomized controlled trial to evaluate each stimulus' impact on the crimes launched by the respective honeypots.

The last empirical study presented in Chapter 7 focuses on the differences in attackers' response to one aspect of deterrence based on their country of origin. Attackers were randomly assigned four different configurations combining a banner announcing the presence of a surveillance system and surveillance processes. By studying the country of origin we draw the focus on the possibility that attackers may respond differently to deterrence cues because of their cultural background but also because of the punishment likelihood that varies from one country to another.

Attackers' differential responses were characterized by comparing several metrics including the total number of sessions, the number of sessions with keystrokes, the number of first sessions with keystrokes, the duration of sessions with keystrokes, and the system activity commands in any sessions and in the first session.

Chapter 8 concludes this dissertation with the limitations, future directions and a review of the main contributions of this work.

Chapter 2 – Background

Very few studies have been conducted in the area of computer security quantification and characterization. This section provides a literature review that motivates the research questions and the associated hypotheses for this research. Furthermore, this section provides an overview on honeypots, the testbed designed for the experiments used to answer our research questions.

2.1 Honeypot-based Experiments

Lance Spitzner defines honeypots as a security tool whose value lies in being probed, attacked, or compromised [SPI02]. In other words, these are highly monitored computer systems meant to attract attackers, analyze their modus operandi and profile attackers [RAM07]. Placed in production environments, honeypots take an active part in the security of a network by providing information on attacker and attack patterns. Niels Provos introduces two types of honeypots [PRO07]: high interaction honeypots (HIH) that involve the deployment of real operating systems (OSs) on real or virtual machines, and low interaction honeypots (LIH) that are computer software emulating OSs and services.

It is important to understand that the traffic observed on honeypots is not legitimate and can thus be considered to be malicious. Levine et al. showed the usefulness of deploying honeypots across large enterprise networks [LEV03]. In their study, Snort [ROE99] was used to detect compromised computers across Georgia Tech's network. DarkNOC [SOB11], a honeypot-monitoring tool, performs a similar detection using network flow records [NFD15] data on the University of Maryland's network.

Over the past several years, honeypot-based experiments have significantly contributed to a better understanding of the threat landscape.

2.1.1 Attackers' Behavior

In [ALA06] and [BER09] the analyses primarily focused on the attackers' behavior after compromise of the honeypots. Both studies leveraged high interaction honeypots using Secure Shell (SSH) as a point of entry. In both studies, simplistic passwords with a high probability of being guessed were implemented to facilitate the attackers' access. In [ALA06], the authors collected the data from 38 SSH intrusions over a period of 131 days. Preliminary attack behaviors were observed, but the testbed in place did not permit outbound connections from the target device, which limited the attackers' pool of available actions. In [RAM07], the authors built a similar testbed, but permitted outgoing traffic from the target device. The authors created a state-machine of the attackers' behavior based upon observations of data pertaining to the 824 attacks collected. The honeypots were re-imaged every 24 hours during this experiment. In [BER09], the authors conducted a more extensive study wherein 1,171 attacks sessions and 250 rogue software elements were collected over a data collection period of eight months.

2.1.2 Binary Analysis

Regarding binary analysis, [ABU06] leveraged the SGNET honeypot project to conduct in-depth analysis of polymorphic malware. The authors introduced clustering techniques based on static and behavioral characteristics analysis.

On the dynamic analysis of rogue software, [BAY06] introduced a tool for dynamically analyzing the behavior of Windows executable files. This tool is available through an online service for analyzing malware [ANU15]. However, only Windows binaries can be submitted for analysis. The same limitation applies to CWSandbox [CWS14] that uses another automated dynamic analysis tool that was introduced by its authors in [WILL07].

2.1.3 Keystroke Analysis

Regarding keystroke analysis, [ABU06] considers the case of guessing typed text in an SSH connection by analyzing only the inter-arrival time of network packets. The authors showed that network latency has little influence on the results. The tools Chaos Reader [CHA15] along with SSH Analysis [SSH15] implement a recognition of commands typed through SSH interactive session using keystroke intervals collected previously from a telnet session. Identification by keystroke analysis is tackled more broadly in [ILO03, MON97, JOY89], where the authors present different algorithms to perform keystroke recognition of users through an analysis of delays between keystroke and keystroke duration. Their approach consists of grouping keystrokes into pairs and observing the time interval between these pairs.

2.1.4 Country of Origin

On the country of origin for the attack, [KAA06] and [VIS11] use the data collected from low interaction honeypots where the malicious activities are limited to port scans and requests sent to fake services. [VIS11] shows the country of the attackers targeting their low interaction honeypots on a map using *GeoPlot*. In [KAA06], the authors studied the correlation of the country of origin of the attackers with the number of attacks per unit of time.

The authors in [STU12] collected and analyzed data from four distributed honeypots presenting a weak SSH configuration. The authors showed the countries of origin from the most observed attacks on the honeypots. They also introduced IP specialization by showing that the origin country of the initial brute force attack is different from the origin country of the subsequent attack sessions.

In [PAU14] the authors presented RASSH, an emulation-based SSH honeypot capable of learning new behaviors during its interaction with attackers. The country of origin of the attackers is used when RASSH chooses an “insult action” to send a message to the attacker’s shell in his or her native language.

The authors of [MAI15] studied the origin of attackers on two different Secure Shell honeypot networks located in two different research sites in China and Israel. More specifically, they looked at the attackers’ region (Europe, Asia, North/South America, and Middle East) for the initial brute-force attack and the first attack session. They compared the temporal trend of both attacks against both Chinese and Israeli target computers. The study revealed the geographical proximity of the attacker during the first attack session, i.e., Chinese honeypots were more likely to be accessed by attackers in Asia.

2.1.5 Cybercrime

Research has also been published in the field of criminology [KIT03, HUNT09] that focuses on defining the different categories of cybercrimes. These studyies are mainly based on anecdotal and other qualitative evidence as opposed to empirical studies directly testing the impact of a theoretically derived honeypot configuration upon subsequent behavior.

2.2 *Honeypot Background*

2.2.1 Problem Statement

The Internet is composed of about 2^{32} Internet Protocol (IP) addresses designating a network device, such as computers, on the public network. Each IP address does not receive the same attacks and is not targeted by the same attackers using the same technics.

Individual honeypots or networks of honeypots, called honeynets, have been used to conduct various studies of attackers [ALM08, CUR04] and cybercrimes such as unsolicited electronic mails, phishing [DHA06], identity theft and denial of service attacks. In their current design, distributed honeypots are often built upon a low interaction tool such as *honeyd* [PRO03]. The distant locations only have to maintain a sensor that will send its logs to a central server.

Another limitation is the number of honeypots that can possibly be deployed. Each honeypot requires one public IP in order to be a target on the Internet. The current version of the Internet Protocol has a limited number of available IP addresses for the whole world. Because of the omnipresence of the Internet in almost every aspect of our lives, more and more devices are connected to the public network. As a consequence, the pool of unallocated IP addresses is getting close to exhaustion. Nowadays, it is difficult and expensive to obtain IP ranges to either expand existing networks or add new ones, and an IP allocated to a honeypot is an IP not allocated to a legitimate Internet service. As a consequence it is difficult to dedicate IPs to honeypots. Furthermore, running high-interaction honeypots is resource consuming.

There is no all-in-one distributed honeypot solution allowing for the deployment of mixed and large environments of low and high interaction honeypots, centralized data collection and provision of safety mechanisms to maintain attack containment.

To design a honeypot architecture that would meet the data collection requirements, we studied the state of the art of honeypots and honeypots networks. More specifically we looked at the existing architecture and software solutions for LIHs, HIHs, and distributed honeypots networks.

2.2.2 Low Interaction Honeypots

Several tools have been developed to emulate different OSs, services and networks. One of the most popular tools is *Honeyd* [PRO03]. *Honeyd* provides a simple way to emulate services, computers with specific OSs and networks of computers. Ad hoc scripts written by hand handle the interactions with the attackers. Authors in [LEI05] propose a method to automatically generate scripts for *Honeyd*.

Other low interaction honeypots focus on specific services, e.g. *Kippo* [KIP15], *Glastopf* [GLA15] and *Conpot* [CON15]. The *Kippo* tool emulates a Secure Shell Server that records usernames and passwords attempted and can even provide a fake shell prompt along with a fake file system when an attacker successfully finds the predefined “successful” credentials. *Glastopf* emulates numerous vulnerabilities to collect data on attacks targeting web applications. *Conpot* collects data on attacks targeting industrial control systems. Honeypots are usually servers collecting attacks against Internet services whereas honey-clients collect malicious code provided by malicious services and executed on the client side. *Thug* [THU15] and *PhoneyC* [PHO15] are example of honey-clients. They browse the Internet collecting malicious content from malicious web pages. Some other LIH tools such as *Dionaea* [DIO15] and *Amun* [AMU15] focus on collecting malware by emulating vulnerable services.

Honeydrive [HON15] is a Linux distribution dedicated to running low interaction honeypots solutions such as *Dionaea*, *Honeyd* or *Glastopf*. About a dozen low interaction tools are pre-installed along with a full suite of security and network tools.

2.2.3 High Interaction Honeypots

Despite their ease of deployment LIHs suffer from their limited fidelity to the real service they emulate, which makes them easy to detect. Numerous studies have used HIHs, but because of the hardware and time resources they require HIHs are often designed to serve a specific purpose: collect the data necessary to answer a research question. In [ALA06, BER09, RAM07, SAL11, STU12] the HIHs expose SSH Servers with either weak passwords [ALA06, BER09, RAM07, STU12] or other vulnerabilities facilitating the attacker access to the honeypot [SAL11]. In [ALA06, STU12] the kernel was patched to collect the keystrokes and specific system calls. In other studies, keystrokes loggers and system call tracers were employed to collect attackers' keystrokes and track their actions [BER09, RAM07].

In [VRA05], the honeypot architecture used Linux lightweight virtualization to deploy HIHs at a large scale. The downside of this method is the constraints on the guest OS: it has to be a modified Linux OS. As a consequence it is not possible to run various types of OSs. On the other end, in [JIA06] presents Collapsar, a honeypot architecture based on full hardware virtualization that allows the deployment of HIHs of various types.

2.2.4 Honeypot Networks

Companies and researchers currently deploy honeypot networks at different scales. Also known as honeynets or also called honeyfarms, these honeypot networks can be limited to a few IP addresses on the local network or distributed systems in several locations such as the Leurre.com project [POU05], the Internet Motion Sensor [BAI05], SGNET [LEI08a, LEI08b], Collapsar [JIA06] or the honeynet initiative from CAIDA [VRA05].

Visoottiviseth et al. present a distributed honeypot framework using low interaction honeypots [VIS11] running the *Honeyd* daemon [PRO03]. More specifically, they describe the working of the *Honeyd* logs centralization and their analysis [VIS11]. The framework only works with *Honeyd* log files. The level of interaction of our framework is also different since we are running LIHs in addition to HIHs.

There are limited software solutions to deploy, maintain and manage networks of honeypots employing different levels of interactions. The main honeynet management solution is *Honeywall* [CHA04] developed by the Honeynet Project. *Honeywall* is a bootable CD-Rom that installs a Linux-based network gateway to manage and control honeypots as well as visualize and analyze honeynet logs. It is an all-in-one solution for small-scale honeypot networks. It provides routing, capture and analysis capabilities.

Bifrozt [BIF15] is also a Linux-based firewall and network gateway configured to act as a transparent Secure Shell proxy allowing the capture of all the attackers interactions with the honeypot Secure Shell Server while blocking potential malicious outgoing traffic. Bifrozt only works with one honeypot running Secure Shell.

Chapter 3 – Experimental Testbed

3.1 UMD Honeynet

3.1.1 Introduction

The honeynpot network hosted at the University of Maryland (called UMD Honeynet) was initially built in 2004 with unused IP addresses from the campus network. More recently, other organizations joined the initiative:

- AT&T Labs, U.S.A.
- The University of Illinois at Urbana Champaign (UIUC), U.S.A.
- The Laboratoire d'Analyse d'Architecture des Systèmes (LAAS) in Toulouse, France
- The Ecole Nationale des Sciences Appliquées (ENSA) in Marrakech, Morocco
- The Technische Universität in Dresden (TUD), Germany
- PJM Interconnection (PJM), U.S.A.

Each organization contributes to the UMD Honeynet by providing ranges of public IP addresses. These IP ranges are routed transparently to the honeypot network hosted at UMD.

3.1.1.1 Objectives

The objective of the UMD Honeynet is to provide an infrastructure to support honeypot-based experiments. The network features centralized data collection and guarantees a realistic, but controlled and flexible environment to safely deploy experiments. The advantages of the present architecture are numerous:

- A single monitoring server collects and stores the Snort events, the flow data and the network traffic, providing visibility across a range of exposed networks.
- The experiments are easy to deploy without the need to create tunnels or to setup specific network configurations.
- The UMD Honeynet is scalable; new organizations can join the project by providing a range of IP addresses.
- The centralization of the honeynet in one location guarantees a uniform configuration of the honeypots.

In addition to the central data collection, ease of honeypot deployment and scalability, the UMD Honeynet presents a range of safety features, such as bandwidth limitation and firewall, which assist the containment of attacks launched by potentially compromised honeypots.

3.1.1.2 History

We are currently running the fourth generation of UMD Honeynet. The first version was deployed in November 2004 and had been used until the end of 2006. It was essentially built upon several chunks of the UMD Institute for Systems Research IP space. Most of the honeypots were high interaction systems running on physical hosts.

The second generation was propelled by the growing interest in honeypots from the University of Maryland's Division of Information Technology. The security and network teams helped to expand the size of our honeypot network: several entire unused subnets were made available to deploy low interaction honeypots.

Deployed in 2008, the third generation of the UMD Honeynet started to support the redirection of Internet traffic from a remote location to the UMD network. This feature has allowed external entities to join the dark network: AT&T has been a part of the UMD Honeynet since 2008, UIUC since 2009, LAAS since 2010, and more recently ENSA, PJM and TUD since 2012.

With the increasing number of public IP addresses, the UMD Honeynet became larger and more complex. The fourth generation, deployed in September 2009, completely revamped the architecture in order to face new challenges such as providing better protection, centralization and stronger reliability of data collection, and easier experiment deployment.

3.1.2 Network Architecture

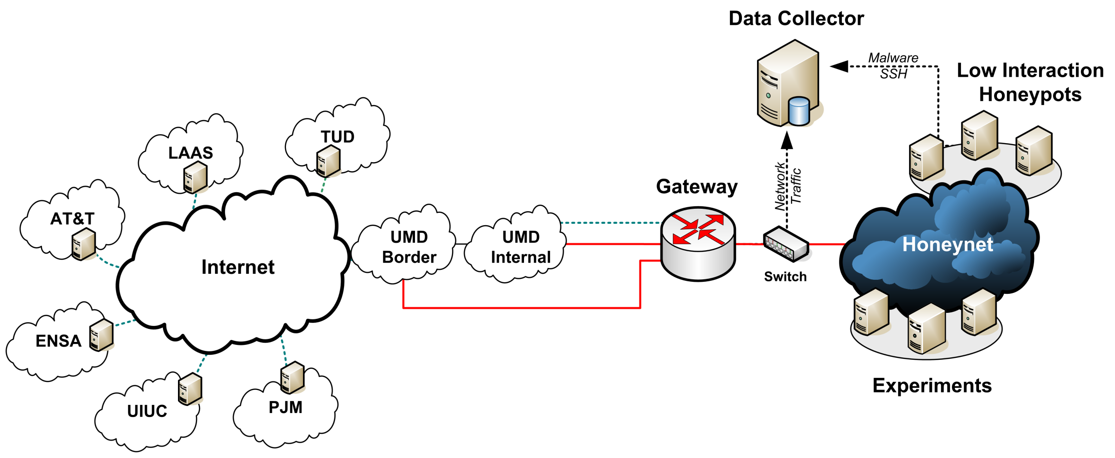


Figure 1. Honeynet Architecture

The current generation of the Honeynet provides a secure network architecture with about 2,000 public IP addresses, a core network at one gigabit per second, a bandwidth of one gigabit per second to the campus network and fifty-two terabytes of Network Area Storage for the data collection. Each organization contributing to the Honeynet provides a range of public IP addresses also called a subnet.

As shown in Figure 1, the Honeynet network architecture is composed mainly of a gateway, a data sensor and data repository.

3.1.2.1 Honeynet Main Router

The Honeygate is the Honeynet main router. This host is in charge of routing or bridging the incoming and outgoing Honeynet traffic. This device is also responsible for enforcing the Honeynet security policy by filtering the outgoing traffic. This policy protects the UMD network and the Internet against potentially compromised honeypots.

3.1.2.2 Data Collector

The host called Spy is responsible for the data collection from the entire Honeynet framework. Spy acts as a passive sensor; it collects data without interacting with the systems it monitors.

3.1.2.3 Low Interaction Honeypots

UMD Honeynet is currently hosting about 2,000 IP addresses. It is practically impossible to use all of these IP addresses in experiments. When an IP address is not in use, it is re-allocated to the farm of low interaction honeypots. Each subnet has its own low interaction honeypot (LIH) host to facilitate the configuration of the LIH tools and the log processing. Each LIH host is a virtual machine that runs *Dionaea* and a fake Secure Shell Server on top of a Linux OS.

<u>*Dionaea*</u>

Dionaea, a LIH tool used to emulate common vulnerable services, has been deployed on the different LIH hosts present on the honeypot network architecture. Dionaea captures a malicious payload submitted by attackers during the exploitation of exposed network services. *Dionaea* presents several advantages compared to a high interaction honeypot (HIH): 1) it emulates many well-known vulnerabilities and protocols, 2) it is easier to maintain than a HIH, and 3) the level of interaction is sufficient to allow successful malicious payload injections.

As shown in Figure 2*, Dionaea's* default configuration exposes several well-known vulnerabilities of common Internet services such as *http*, *ftp*, *smtp*, MS SQL, MySQL, as well as Microsoft Windows and VOIP protocols. Because of the nature of the exposed vulnerabilities, *Dionaea* essentially captures Windows Portable Executable (PE) files [PEF15], the executable file format used on Windows platforms.

```
Starting Nmap 5.21 (http://nmap.org) at 2012-11-12 22:24 EST
Nmap scan report for XXX.XXX.XXX.XXX
Host is up (0.039s latency).
Not shown: 986 closed ports
PORT     STATE    SERVICE
21/tcp   open     ftp
25/tcp   filtered smtp
42/tcp   open     nameserver
80/tcp   open     http
135/tcp  open     msrpc
443/tcp  open     https
445/tcp  open     microsoft-ds
554/tcp  open     rtsp
1433/tcp open     ms-sql-s
2222/tcp filtered unknown
3306/tcp open     mysql
5060/tcp open     sip
5061/tcp open     sip-tls
7070/tcp open     realserver
Nmap done: 1 IP address (1 host up) scanned in 9.18 seconds
```

Figure 2. Open ports in Dionaea

Dionaea waits for attackers to inject malicious payloads known as shellcodes by exploiting one of the service's vulnerabilities. The shellcodes are evaluated using *libemu*, a C library able to detect and execute shellcodes using the GetPC heuristics [POL10]. The shellcode profiling allows *Dionaea* to act upon three possible intentions: 1) providing a remote shell to the attacker by opening a network socket on the targeted system, 2) downloading a file from a remote location using ftp, http or

SMB protocols, or 3) executing an existing binary file on the local file system of the target host. *Dionaea* executes multi-staged shellcodes in a virtual machine using *libemu* to infer their final intent.

Binary files can be captured in different ways: ftp and http downloads, and downloads occurring during the shellcode executions. They can have different formats. The UNIX command *file* [FIL15] allows the file format to be identified. Empty and ASCII files are automatically removed from the repository as well as the data format that describes unknown binary files. *Dionaea* names captured binary files after their MD5 hashes and logs the submitted malware into a SQLite database. Each entry of the submission database contains:

- The MD5 hash of the binary,
- A capture timestamp,
- The source and destination IP addresses,
- The source and destination ports,
- The protocol exploited,
- The transport protocol (TCP or UDP), and
- The URL used to download the binary file.

We observed that the same binary file can be submitted several times by different originating hosts.

Fake Secure Shell Server

Secure Shell (SSH) is a network protocol used to access the shell (or command line interface) of a remote computer through a secure channel. This remote access service is often the target of attackers trying to guess usernames and passwords.

The fake Secure Shell server is a C program emulating the authentication phase of a SSH server. The attackers can connect to the fake SSH server and try different combinations of usernames and passwords. This LIH tool is designed to reject all authentication attempts. As shown in Figure 3, each attempted login name and password are logged along with the offender IP address and a timestamp.

```
umd-1;2014-04-09 23:56:51;117.27.158.102;X.Y.Z.Z;root;jiahuo1111111
umd-1;2014-04-09 23:56:52;117.27.158.102;X.Y.Z.Z;root;gks
umd-1;2014-04-09 23:56:49;117.27.158.102;X.Y.Z.Z;root;!@#$%^&*(
umd-1;2014-04-09 23:56:49;117.27.158.102;X.Y.Z.Z;root;rootrootroot
umd-1;2014-04-09 23:56:51;117.27.158.102;X.Y.Z.Z;root;110120
```

Figure 3. SSH brute force attempts logs

3.1.2.4 Honeymole Servers Farm

Honeymole [HON15] is a tunneling program that creates a secure communication bridge between a Honeymole client and server. This tool allows us to forward the honeypot traffic from the external organizations to the UMD Honeynet.

The client, hosted on the remote location network and the server, hosted on the UMD network, captures the required traffic to port the external entity honeypot IP addresses to the UMD Honeynet.

3.1.3 Datasets

The main switch used on the UMD Honeynet replicates the network traffic of the whole framework on a special port called a mirroring port. One of Spy's network interfaces is connected to this mirroring port allowing it to collect data from the whole architecture without interacting with the monitored systems.

Spy automatically collects and organizes the following data repositories for each subnet:

- Raw network traffic

- Network flow records
- Intrusion Detection System (IDS) alerts

In addition to these datasets, Spy maintains a repository of malware and centralizes the logs of the Secure Shell login attempts. Both of these repositories are collected by the LIHs introduced in Section 3.1.2.3 of Chapter 3.

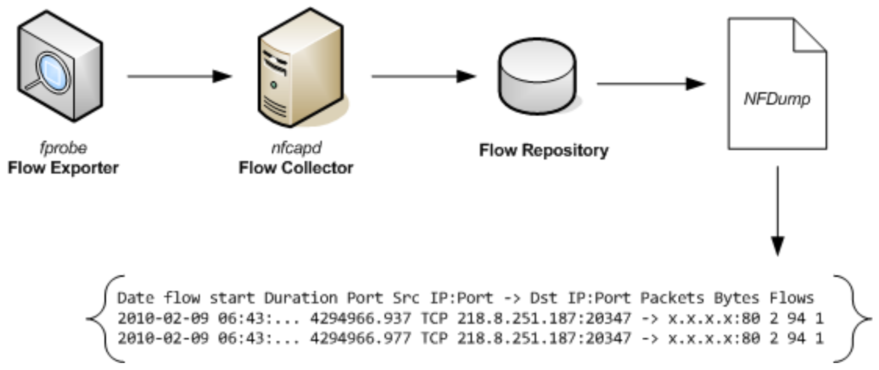
3.1.3.1 Raw Network Traffic

Raw network traffic is collected with *tcpdump* [TCP15], a command line packet analyzer that uses the *libpcap* [TCP15] to capture network traffic. A script launches the *tcpdump* tool in the background to collect the network traffic in a temporary file. Every hour, the script stops the network traffic collection, rotates the temporary file and restarts *tcpdump*. The newly rotated file is then used to create a new file containing the network traffic of each experiment deployed on the UMD Honeynet and each subnet. The rotated file is then given a new name and moved to the network traffic repository.

3.1.3.2 Network Flow Records

A network flow record summarizes the communication between two network end points (defined by the IP addresses and port numbers of the end points). Included are the time, duration, and numbers of bytes and packets, but not the payload information (i.e., content of the messages transmitted). Figure 4 shows the different components involved in the collection and storage of network flow records.

The flow exporter *fprobe* [FPR15] has been setup on Spy to export Netflow version 9 records. The exported flow records are then collected by a flow collector called *nfcapd*. *nfcapd* is part of the *nfdump* [NFD15] tools, a set of tools that collects and processes flow data.



```
Date flow start Duration Port Src IP:Port -> Dst IP:Port Packets Bytes Flows
2010-02-09 06:43:... 4294966.937 TCP 218.8.251.187:20347 -> x.x.x.x:80 2 94 1
2010-02-09 06:43:... 4294966.977 TCP 218.8.251.187:20347 -> x.x.x.x:80 2 94 1
```

Figure 4. Network Flow Records Collection Architecture

The network flow repository is entirely managed by *NFSen* [NFS15] a web interface for visualizing the network flow records. *NFSen* relies on the *nfdump* tools to configure and automatically launch the flow collector to organize the repository. It offers the ability to create profiles. These profiles are used to create different sets of network flow record files for each subnet and deployed experiment.

3.1.3.3 Network Intrusion Detection System

Snort is a Network Intrusion Detection System (IDS) [ROE99] designed to detect attacks by monitoring and analyzing the network traffic. IDSs use two methods to detect malicious traffic: signature-based detection where known malicious behaviors are described by a set of signatures and anomaly-based detection where heuristic algorithms are used to determine normal versus anomalous network traffic.

Spy runs two instances of *Snort*. The live instance updates in real time a database of Snort events. The second instance is executed every night against the network traffic collected during the previous 24 hours. A text file of the alerts for each subnet is generated and archived in a repository.

3.1.3.4 Malware and Secure Shell Repositories

Every night, Spy fetches the malware and the Secure Shell login attempts from the different LIH hosts of the Honeynet. The malware repository is organized per subnet.

3.1.4 Security and Deployment Policy

Because of the nature of the honeypots, operating them can be a risky activity. The compromise phase can be part of the honeypot-based experiment design or an unknown vulnerability can be exploited. An attacker can use one compromised honeypot to attack some other host on the Internet from a UMD IP address. To mitigate that risk, two policies ensure that good practices for honeypot deployment and proper security mechanisms are in place to maintain the network isolation rule.

3.1.4.1 UMD Honeynet Global Security Policy

Table 1 shows the global UMD Honeynet security policy enforced by Honeygate and the actions taken by the administrators to mitigate the risk of having one honeypot launching attacks and compromising other systems.

Table 1. UMD Honeynet Global Security Policy

Interaction	Likelihood	Risk Mitigation	Incident Response
Low	**Low**	Physical host must be hosted on a dedicated *honeypot management network*. Physical host must use a private IP within a dedicated *honeypot management network.* Firewall must block outgoing traffic from the physical host IP address	Host Isolation (Block IP on Honeygate) Host restoration
High	**Medium**	Outgoing traffic must be rate limited to avoid compromises of other Campus computers and Internet hosts	Host Isolation Host restoration
All	**All**	Bandwidth limitation 100Mbit/s Honeypots cannot reach UMD Network	Host Isolation Host Restoration

By default, the various firewalls set up on the main gateway allow all traffic to enter the UMD Honeynet, but per the security policy, the outgoing traffic is filtered to avoid network congestion and to block any communication initiated by the honeypots targeting any network device on campus.

3.1.4.2 Deployment Policy

To maintain attack containment and UMD Honeynet isolation, the deployment policy tasks the experiment designer with a list of recommendations to implement and actions to execute before the launch of an experiment.

1) Determine the level of interaction and likelihood of compromise

The level of interaction will greatly influence the risk associated with the experiment. LIHs are less likely to be compromised, and thus represent a lower risk to operate. On the other hand, HIHs life cycle often includes a compromise phase where the attacker gets control of the system. Also, the attackers may exploit a vulnerability in the architecture and escape their containment area within high interaction honeypots.

2) Determine the risks associated with compromise (experiment-specific risks)

This step is consists of enumerating the possible problems associated with the experiment and their likelihood of occurrence. For example, Linux computers are often used to launch SSH brute-force attacks on the Internet after compromise. This list may evolve over the course of the experiments.

3) Mitigation plan or experiment security policy

The mitigation plan aims to create an experiment-specific security policy to limit all of the risks or problems previously identified. In the Secure Shell brute-force example, we can add a firewall rule to limit the outgoing Secure Shell traffic to only five brute force attempts per minute per target. This would considerably limit the risk of compromising another Internet host. The mitigation plan or experiment specific security policy should be applied on both the Honeygate and at the experiment level. Applying the security policy on Honeygate provides an additional layer of security in case the protection mechanisms on the experiment are compromised or inactive.

4) Remote management interface

Each honeypot experiment should have a remote management interface connected to the Honeynet Management network. This is to maintain network isolation and separate malicious traffic and legitimate administration traffic.

5) Experiment data backup

The data collected by an experiment is considered to be critical and should be stored in a more secure server separated from the UMD Honeynet.

6) Experiment kill-switch

The experiment design should include a procedure to shutdown the experiment in case of emergency (testbed out of control).

7) Experiment monitoring and compromise detection mechanism

Details regarding the compromise detection and monitoring of the different components of the experiment should be included in the design.

8) **Obtain approval before launching an experiment**

Because of the sensitivity and risks associated with hosting honeypots, all of the details of the experiment design must be validated by the Division of Information Technology, Security and Policy Office at the University of Maryland.

3.2 Cybercrime Framework

3.2.1 Introduction

The framework described in this section has been developed to support different honeypot-based experiments designed to understanding the attackers' behavior on a system following a "successful" compromise, i.e., when an attacker gains access to a honeypot. All the testbeds for this research have been grouped in one framework providing the core functionalities common to all experiments. Each experiment has been designed to satisfy the following constraints: 1) the honeypots must offer a frequently probed and vulnerable point of entry; 2) a large number of honeypots must be made available to the attackers at the same time; 3) different honeypot configuration types must be randomly assigned to attackers in order to identify the impact of the specified treatment condition, i.e. the configuration factor we want to test upon crime outcomes; 4) relevant data must be collected to characterize the attacks developed and the crimes committed by the honeypots following the initial compromise of the honeypot.

Each honeypot-based experiment presents several characteristics such as the level of interaction, the point of entry, the vulnerability allowing the attacker to enter the honeypot and the "honey," the artifact that will make the honeypot attractive to attackers.

3.2.1.1 Level of Interaction

For the purpose of this experiment we want to study the behavior of the attacker after a successful compromise. Our honeypot must be "compromised" and provide an environment as close as possible to a real systems. This can only be achieved with a high-interaction honeypots, as low-interaction honeypots are not sophisticated enough to support all of the actions made by an attacker.

3.2.1.2 Point of entry

We need a point of entry that provides remote access to systems. Among the most commonly probed services on the UMD network, only three provide remote access functionalities: VNC (TCP/5900), Remote Desktop Protocol (RDP) (TCP/3389) and Secure Shell Protocol (TCP/22).

VNC or Virtual Network Computing is a tool allowing users to take control of a remote computer. The user interface, whether it is graphic or text, is displayed on the user's remote computer. This protocol would require a video recording of the attackers' session to capture all of his or her interactions with the honeypot. Videos are difficult and burdensome to process and analyze.

Remote Desktop Protocol or RDP provides the same remote access experience as VNC for Microsoft Operating Systems such as Windows XP or Windows Server 2008. It presents the same drawbacks as VNC.

Secure Shell creates a secured communication channel to establish a connection to a remote host shell. This protocol is entirely text-based and the interactions are keystroke-based. The keystroke capture will provide the list of commands executed on the honeypot. *For this experiment, SSH is the chosen point of entry.*

3.2.1.3 Vulnerability

A common vulnerability introduced on SSH servers to facilitate the attackers' honeypot compromise is the weak password. Several studies involving SSH-based honeypots introduced accounts with commonly attempted usernames and passwords. This method presents a few limitations:

- Several attackers can "compromise" the honeypots and apart from the IP address nothing can discriminate them.
- An attacker may come back several weeks later and use the same credentials to gain access to the honeypot again.
- Depending on their brute-forcing method and the dictionaries used, they may not find the right credentials.

A better solution is to allow the attacker to access the honeypot after a random number of attempts and create the "successful" credentials on the compromised honeypot. This method was selected for our research.

3.2.1.4 "Honey"

Computer resources such as processing power or storage can be also considered as "honey." Attackers can use the hard disk space for file sharing purposes. A Linux box can be turned into a web server and host a fishing website. The processing power that represents these hosts can be used for attackers to achieve other bigger mischiefs on the Internet.

3.2.1.5 Datasets

In order to study the attackers' behavior we need to know all the commands they type during their SSH sessions. The data collector already captures the raw network traffic and the network flow records. These two datasets are useful to identify the attacks launched by the honeypots after their compromise.

3.2.2 Framework Design

To satisfy the different requirements in data collection and design, we developed the experimental platform shown in Figure 5. This testbed allows us to implement the SSH vulnerability, the keystroke capture and the security policy.

The study of the attackers' behavior requires deploying several experiments with a similar design. Only the "treatment" received by the attackers changes from one experiment to another. The different treatments are motivated by the respective research questions inherent to each experiment.

Each Cybercrime experiment uses three different types of hosts: a network gateway, a collector host and a set of machines, called OVZ hosts. All of these hosts are virtual machines running on the lab VMware cluster. Because of the specific network configuration that has to be shared across three different hypervisors, the virtual machines for the cybercrime projects are hosted on three specific VMware servers.

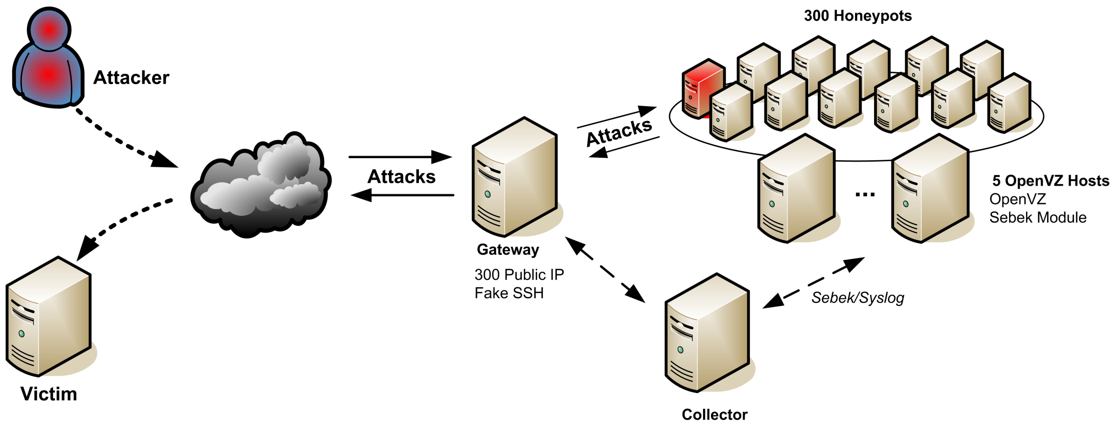


Figure 5. Experiment Design

The gateway for the system shown in Figure 5 is placed between the Internet and the other components of the framework, and accepts SSH connections on port TCP/22. The OVZ hosts run OpenVZ [OVZ15], a lightweight virtualization solution for Linux systems. OpenVZ allows us to run several honeypots per OVZ host in parallel. The collector is common to all the cybercrime experiment testbeds. This host centralizes the collected data and the processing.

3.2.2.1 Collector and Management Host

The cybercrime framework provides an environment that facilitates deployment and management of the cybercrime experiments. It also provides a consistent way to collect and process the data generated by the honeypots.

Data collection, processing and storage

The collector host is responsible for centralizing and organizing the data generated by the different monitoring tools used on the framework.

This host receives:

- The authentication events from all the honeypots
- The key logger traces from all the OVZ hosts
- The honeypot deployments from all the gateways

A Perl script processes the raw data and uploads them in a database on a daily basis. The data processing script rebuilds the attackers' sessions on the honeypots from the Syslog [SYS15], a computer log storage management program, authentication logs sent by the Honeypots. It also cleans the keystrokes data and matches them with the attackers' sessions.

Central Repository

All of the scripts running on the cybercrime framework are stored in a repository on the collector host. The scripts are automatically distributed to the different components of all of the experiments. The framework is flexible enough to allow experiment specific scripts. In addition to the scripts, the configuration files and the honeypots base images are all stored on the collector host and are all automatically distributed when necessary.

Monitoring

The collector host also monitors the health of the framework: it makes sure that each component is online and that all the monitoring tools are operating correctly. A daily email is generated and sent out to the cybercrime team. It contains information on the data collected as well as the health of each experiment.

The collector host also runs the website that provides access to the live data and status of the framework.

Maintenance

The framework provides a set of scripts that performs the daily maintenance operations of the different components. These operations include the removal of honeypot containers reaching the end of their lifecycle and the automatic creation of new containers to allow further honeypot deployments.

3.2.2.2 Network Gateway

The gateway is attached to three different networks. One network interface is connected directly to the Internet and is configured with the 300 public IP addresses made available for the honeypots. The second interface is attached to a private network where all the honeypots containers are connected. The third network is used for management and data collection purposes. The gateway runs a Linux Ubuntu 12.04 operating system. The Secure Shell server is a custom designed *C* program using the *libssh* [LIB15].

The fake SSH service returns a SSH successful authentication message to the attackers after a number of brute force attempts. This number is randomly selected between 100 and 200 at the very first login attempt by the attacker from a specific honeypot IP address. At this point each attacker is identified by his or her IP address.

When the expected number of attempts is reached, the C program calls an external script that 1) records the deployment in a database on the collector server, 2) configures the next available honeypot container with the login credentials that "successfully" broke into the system, 3) creates a Network Address Translation rule (NAT) to associate the public IP address targeted with the private IP of the newly configured honeypot container and, 4) attributes and applies one of the configuration types of the corresponding experiment. Once the execution of the script is complete, the Secure Shell server establishes a Secure Shell connection with the newly configured honeypot and redirects it to the attacker.

This overall sequence of brute-force entry and the resulting procedures discussed above can collectively be termed as a deployment wherein the intruder is successfully assigned to a honeypot with a randomly assigned type or treatment condition. The execution of the script and configuration of the honeypot container only takes a few seconds. In addition to running the fake SSH server and routing the Internet traffic to the honeypots, the gateway also limits the attacks targeting other Internet hosts to prevent their subsequent compromise. This is achieved by rate-limiting the outgoing traffic on specific protocols and ports. The firewall on the gateway limits:

- SSH scans and brute force attempts;
- UDP datagrams to prevent DoS attacks;
- Web and RDP scans.

3.2.2.3 OpenVZ Hosts

While the gateway manages the traffic and deployment of the honeypots, these honeypots need to be constructed and directly maintained by additional hosts. As we wanted to provide a UNIX environment to the attackers, OpenVZ was deployed on five CentOS 5.4 systems to perform the construction and maintenance tasks necessary for the honeypots to exist and function as intended. We used the stable release of the OpenVZ kernel 2.6.18-164.15.1.el5.028stab068.9 for the deployment.

As mentioned previously, OpenVZ is a lightweight virtualization tool for Linux-based OSs. An OS is fundamentally composed of two main elements: the kernel and the user space. The kernel controls the computer hardware and provides the applications executed by the users functions to interact with the hardware. The user space or application space is where all the users processes and software are executed. Each virtual machine on VMware or other full virtualization solutions will have virtual hardware (i.e., CPU, memory, network, hard drive) and will run a complete operating system with kernel and user space. On the other hand OpenVZ, the host OS will share its kernel and user space. Each container will be a sub-tree of the host operating system process list. Each container will have its own file system, but this file system will be a sub-directory on the host OS file system.

The advantages of OpenVZ are threefold: first, OpenVZ allows running several lightweight Linux OSs in parallel on a single OVZ host. After stress-testing the OVZ hosts, we determined that we could easily run up to 60 containers per OVZ host at the same time. With 5 OVZ hosts, this solution gave the ability to run up to 300 honeypots in parallel. Second, OpenVZ provides the tools to easily adjust the containers' configuration including the IP address and the credentials. Moreover, the OpenVZ virtual network interface does not permit the change of IP address within the honeypot container. As a consequence, attackers are not able to change the honeypot

IP address following compromised entry. Lastly, attackers cannot interact with other honeypot containers or with the host OS. The attacker's actions are restricted to his or her assigned container. Even with root privileges, each container is isolated from the other devices within the design and from the host OS running on each OVZ host, but we can access the containers (honeypots) file systems and process list.

Each container or honeypot is built upon Fedora 12 operating system. Each container comes with two servers: Apache and Secure Shell Server.

3.2.2.4 Datasets Definition

In addition to the network flow records and the raw network traffic collected by the data collector, we also gather information relative to the attackers' Secure Shell sessions that include the commands typed by the attackers.

Deployments

This dataset contains for each honeypots:

- The deployment timestamp
- The attacker's IP address
- The country of the attacker based on the IP address
- The origin network number based on the attacker's IP address
- The targeted public address on the Honeynet network
- The successfully "guessed" login and password
- The honeypot type or treatment number

Session

In the session dataset, it is possible to find all of the Secure Shell sessions for each honeypot along with the following information:

- The username used by the attacker to access the honeypot
- The login and logout times
- The IP of the attacker
- The country and network of origin of the attacker based on his or her IP address

Keystrokes

To capture the attackers' keystrokes, we use a key logger from the Honeynet Project called Sebek [SBK15]. Sebek is a kernel module that extracts the keystrokes from the *read* system call. A modified version of the Sebek module has been deployed on each of the OVZ hosts. The module has been modified to support OpenVZ and add the Honeypot ID in the log. The keystroke collection generates two different outputs per sessions:

- The raw keystrokes (Figure 6)
- The keystrokes processed into lists of commands (Figure 7)

For both datasets, Sebek provides a timestamp, a Virtual Environment ID (VEID) also known as honeypot (HP) ID and container (CT) ID, the OVZ host IP address, the user ID, the process ID, the file descriptor and i-node of the standard output, and the command name. The keystroke output provides an additional timestamp in microseconds. This timestamp allows us to see the time difference between each keystroke.

```
[2014-10-05  10:28:38  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash MS:1412519318193697 ]#w

[2014-10-05  10:28:38  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash MS:1412519318389729 ]#[RETURN]

[2014-10-05  10:28:38  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash MS:1412519318655731 ]#c

[2014-10-05  10:28:38  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash MS:1412519318817674 ]#d

[2014-10-05  10:28:38  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash MS:1412519318859656 ]#

[2014-10-05  10:28:38  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash MS:1412519318925694 ]#/

[2014-10-05  10:28:39  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash MS:1412519319075653 ]#t

[2014-10-05  10:28:39  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash MS:1412519319187660 ]#m

[2014-10-05  10:28:39  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash MS:1412519319375652 ]#p

[2014-10-05  10:28:39  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash MS:1412519319452646 ]#[RETURN]
```

Figure 6. Sebek Filter Keystrokes Output

```
[2014-10-05  10:28:38  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash ]#w

[2014-10-05  10:28:39  Veid:1021  Host:10.0.10.21  UID:500  PID:12998  FD:0  INO:2
COM:bash ]#cd /tmp
```

Figure 7. Sebek Filter Commands Output

3.3 Initial Cybercrime Experiment

The experimental setup shown in Figure 5 is the initial architecture of the cybercrime framework presented in Section 3.2 of Chapter 3. This experiment was used to collect the required data for the study in Chapter 4. This architecture contained a standalone experiment and consisted of one gateway and one OpenVZ host. At the time, the OVZ host could only handle forty honeypots in parallel. The architecture of the OpenVZ host is similar to the one presented in Section 3.2 of Chapter 3, however the design of the SSH gateway was different. The OS was a Linux Ubuntu 9.10 server installed with modified versions of OpenSSH [OPS15] Secure Shell server, PAM-MySQL, a module handling user authentication against a MySQL database, NSS-MySQL, a library providing users information from MySQL database, and a specific shell provided to attackers when their brute force attack succeeds.

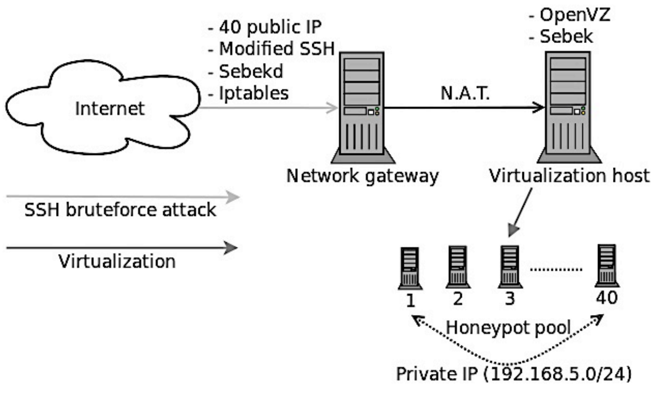


Figure 8. Initial Testbed Design

OpenSSH was modified to reject Secure Shell login attempts on its public IP addresses until a predefined number of attempts. When this predefined threshold was reached, OpenSSH skipped password verification, created a new user with the latest credentials attempted by the attacker and called PAM, which was configured to use the modified PAM-MySQL module for user verification. The predefined threshold was randomly selected between 150 and 200. Moreover, to limit the number of deployed honeypots to three per attacker IP address, the modified version of Secure Shell server rejected any attempt from an IP address that has already deployed three honeypots.

PAM-MySQL initial purpose is to verify user credentials using MySQL as backend. In the modified version, PAM-MySQL created user accounts for attackers and recorded on a MySQL database user information including: login, password, source

IP address, targeted IP address on the gateway, a user ID and the path of the specific shell on the gateway which handles the honeypot deployment.

NSS-MySQL is required to read user information from the MySQL database. This component is required by OpenSSH to access user account information before granting access to a session. The module was modified to handle conflicting cases of user entries with a same user login. The distinction between these entries was based on the targeted IP address of the SSH connection.

Attackers, who had been identified according to their IP addresses, could deploy up to three honeypots that were configured as follows: The first honeypot (HP1) had no network limitation, the second one (HP2) had the main IRC port blocked (port 6667, incoming and outgoing traffic), the third one (HP3) had every port blocked except HTTP, HTTPS, FTP, DNS, and SSH.

The set of honeypots consisted of three configurations that enforced increasing network limitations including a first fully functional environment, a second configuration where only the IRC port is blocked, and a third configuration where only a few services were allowed (HTTP, HTTPS, FTP, DNS, SSH). Attackers could compromise these three honeypot configurations incrementally during one month (the honeypots were backed up and redeployed at the beginning of each month), starting with the first configuration being deployed on 40 public IP addresses.

3.4 Design Limitations of the Testbed

3.4.1 Honeynet

The UMD Honeynet presents several limitations in term of design. First, even though the Honeymole allows transparent traffic forwarding, the attacker could detect the network latency introduced when redirecting traffic to the UMD Honeynet. Second, not all organizations participating to the UMD Honeynet apply the same security policy. Some networks will be protected by firewalls or intrusion prevention systems whereas some others will not have any security. In addition, some organizations do not allow the deployment of high interaction honeypots.

3.4.2 Cybercrime framework

The design of the cybercrime framework contains several limitations on the studies presented in the following chapters. First, the OS is Linux. It is not guaranteed that attackers targeting Windows vulnerabilities will exhibit the same behavior. The point of entry is limited to Secure Shell. As a consequence, the population of the different studies will be limited to attackers targeting Unix systems via SSH. In addition, the use of a modified SSH server granting access to the honeypots after a random number of attempts between 100 and 200 limits also the population of attackers on our experiments. To summarize, the honeypots are accessible to attackers targeting Secure Shell servers running on Unix OS and attempting at least 100 to 200 combinations of username and password during their brute-force attack.

All the honeypots have UMD IP addresses. All the analyses performed using the datasets introduced before will be limited to one location. Moreover, a dedicated set of 1500 IP addresses were provided for the experiments. Finally, the configuration applied to the honeypots should not reveal their nature and should be in line with the UMD policies and the law.

3.5 Human Subjects Research

The subjects in all the experiments deployed on the UMD Honeynet (that consists of low and high interaction honeypots) are attackers who cannot be identified. In addition, it is not possible to determine the exact number of subjects due to the automated nature of some attacks.

After consulting with the IRB office, it was decided that these experiments did not need to be covered by an IRB.

Chapter 4 – An Empirical Study to Analyze Attacks and Attackers

The study presented in this chapter was published in [SAL11]. I co-advised the two student interns developing the project. My contribution is about 35% for this project. I performed part of the data analysis and the writing of the result sections requiring network flow records, IRC traffic and IRC logs analyses.

4.1 Introduction

In this chapter, we present an empirical study to characterize attackers and attacks against targets of opportunity, i.e. when the victim host happens to be on the Internet and has been randomly selected. The experimental design used in this study was deployed on the initial cybercrime testbed described in Section 3.3 of Chapter 3.

4.2 Research Questions and Hypotheses

In this chapter we describe the experiment to study the attackers' characteristics and behavior after successful compromise to empirically address the following research questions:

a) How skilled are the attackers based on the observations of attack sessions?

b) How can we differentiate attackers on a compromised honeypot?

c) What is the purpose of the compromise?

d) Do attackers share the access to their honeypots?

Discerning who is launching computer-focused crimes, i.e., crimes committed on the Internet, is difficult. We divided the attack process between the initial brute force-attack against Secure Shell accounts and the intrusion step when the attacker logs in using the "successfully" guessed username and password. For brute force attacks, we can only rely on the attacker's IP address. However, once the attacker logs into the compromised honeypot, we can also use the keystrokes and the attack types to try to characterize computer-focused crimes further. We analyzed whether the attacker who launched the brute force attack is the same attacker as the one who logs in based on the IP address and whether a single attacker using several IP addresses logs in or multiple attackers share the same compromised honeypot.

A number of compromised Internet hosts can become part of a bot network launching crimes. According to [GOE07], these bots may connect to IRC servers to allow the attackers to control them remotely. Attackers may also set up a backdoor port to access the compromised machine remotely [LEV03]. We hypothesize that when blocking IRC and/or backdoors, skilled attackers will attempt to remove the network restrictions. Therefore, by adding network restrictions, we hypothesize that we will attract skilled attackers.

To expand our knowledge of attacker profiles, we characterized the attackers' skills by introducing a list of ten criteria. We discuss the link between the attacker's skill and particular actions including hiding his/her malicious activities, changing the password or checking for the presence of other users. We also discuss the relationship between the attacker's skill and the attacker's capacity to successfully launch an attack to commit crimes from the compromised honeypots under different network restrictions.

To understand why attacks were launched we categorize them into different classes according to the rogue software installed and exploited. As explain previously, a number of the honeypots will become a bot and will take part of a network of compromised hosts called botnet. IRC is the most commonly used protocol to control the different bots [RAT13]. Therefore we expect the attackers to mainly install IRC-based bot tools.

Finally, we expect a number of the honeypots to be shared with different attackers. It is possible that some of the honeypots will be part of an underground market for compromised hosts [FRA07]. Therefore, we expect the honeypots to be accessed by different attackers.

4.3 Method

To answer these questions we used the data collected from May 17, 2010 to October 31, 2010 with the experimental design described in Section 3.3 of Chapter 3.

For this study we analyzed network flow records and attackers' keystrokes processed into lists of commands, and we use the Honeynet Project's *Honeysnap* tool [HON15] and *Wireshark* [WIR15] to collect the rogue software installed by the attackers on the honeypot.

Two analyses were performed on the keystrokes. The first one was to detect copied and pasted text in attackers' sessions to show how much attacks are automated. The second one is an approach to evaluate how many different users exploit the same honeypot.

To detect copy and pasted, we calculated the average delay between keystrokes for each command typed. A copy and pasted command is generally performed in two steps: 1) the user copies and pastes the command in his/her shell, 2) the user hits the return key to launch the action. We removed the last keystroke interval between the last keystroke of the command and the return because of a possible excessive delay that could give the wrong results. To detect these commands containing copy and pasted text, we use a threshold of 100 milliseconds of average delay interval between keystrokes. This threshold corresponds to the maximum speed of a skilled typist who can type up to 120 words per minute (i.e., 10 characters per second). We assumed that a human being could not go beyond such typing speed.

We noted that attackers often mixed interactive commands and copy and pasted text, especially when they use long paths or URLs. The attack session shown in Figure 9 illustrates this behavior. Text in italic indicates commands containing copied and pasted text and recorded average time interval are shown on the left-hand side.

```
[28 μs]  cat /usr/share/man/man1/.error
[>100 ms]     cat /proc/cpuinfo
[>100 ms]     history -c
[>100 ms]     w
[>100 ms]     ps -x
[>100 ms]     cat /proc/cpuinfo
[42 μs]  wget http://download.microsoft.com/…
[>100 ms]     ls
[>100 ms]     rm -rf W2Ksp3.exe
[>100 ms]     cd /usr/sbin
[28 μs]  wget http://sbebe.110mb.com/img
```

Figure 9. Attack Session Commands

4.3.1 Attackers and Crimes Identification

4.3.1.1 Attacker Identification

A critical step to understand the threat landscape is to correctly identify attackers. Most of the time, an assumption is made by mapping each IP address to a single attacker. However, it is more realistic to assume that attackers can use multiple IP addresses to more adequately hide their traces. In order to go beyond this assumption, we defined multiple indicators to build attacker profiles that we then use to try to uniquely identify human individuals behind each attack. In addition to the IP address, these indicators are:

- Attacker AS routing number and attacker's geographic location: to potentially detect if an attacker comes from a single ISP that has changed the client IP address over the time window of our experiment.
- Attacker specific actions: three subcategories are considered.
    - Rogue software origin: a same attacker often downloads his/her tool from a same location while compromising a target.
    - Techniques to perform specific actions: for example, attackers who try to erase their traces can employ multiple commands. However, they usually use the same method from one attack session to another.
    - Files accessed: many attackers install rogue software in hidden or complex locations. If two sessions access this kind of file, it is usually a hint about the attacker's identity.
    - Comparison of keystroke profiles that we describe further in the remainder of this section.

The goal of keystroke analysis is to help differentiate attackers. According to [SON01], delays due to the network for SSH connections in comparison to delays between human keystrokes can be neglected. Our analysis of keystrokes is based on this assumption. Extensive research exists on keystroke profiling [ABU06, MON97, JOY89]. We considered, in particular, the study presented in [MON97] because it introduces an approach adapted to our goals, which consists of user authentication and user recognition by analyzing data collected from various users typing on their workstations. The approach consists of grouping the recorded text following the most used syllables in English. Then, the graph of keystroke latencies (i.e., time between successive keystrokes) and durations (i.e., length of time keys are pressed) as a function of the syllables reveals the user's keystroke profile. In our case, we recorded session keystroke latencies from attackers' commands that are not copied and pasted text and we compared the delays of each matching pair of keystrokes between all the attack sessions recorded on a honeypot.

Our experiment differs from [MON97] because contrary to their experiment where the authors evaluated the efficiency of recognizing a user through his/her keystrokes, we don't know in advance from which user the session keystrokes originates. Therefore, keystrokes profiles are considered in our experiment as an indicator among other parameters (IP address, AS number, attacker's techniques to perform specific actions, files accessed and rogue software downloaded) to differentiate attackers. Moreover, contrary to their experiment, the quantity of keystrokes used to perform the comparison directly depends on the number of commands the attacker typed. In certain cases the recognition through keystrokes is not possible due to the lack of recorded keystrokes for a session.

4.3.1.2 Attacker Skill

To assess the attacker's skill over the entire recorded attack sequence, we discussed two approaches. The first one consists of asking an analyst to review each attacker session and to attribute a score. The second one introduces several criteria that correspond to various actions performed by attackers. We postulate that a variety of criteria fulfilled by an attacker indicates a relatively higher skill level. As a result, we compute the skill level as a function of the criteria fulfilled. While the first approach is subjective, the second may be limited due to the definition of the criteria. We opted for the second approach because it enables others to easily replicate our experiment.

We developed ten criteria based on four generic questions to evaluate the competences of the attacker. These questions are:

1) *Is the attacker careful about not being seen?* The fact that an attacker does not want to be noticed indicates that the attacker knows that such behavior increases the chances to maintain access to the target. It also indicates that the attacker knows how to reduce his/her traces. We identified four approaches in the criteria: erasing the files that attackers imported on the target from the Internet (Criterion 4, Table 2), deleting the logs that contain traces of the attacker's activity (Criterion 1), restoring the logs not to catch the attention of a user who could notice that the logs are missing (Criterion 2) and checking the presence of other users during the attack (Criterion 3).

2) *Does the attacker pay attention to the environment of the honeypot before committing a crime?* Learning about the compromised host and its environment (especially the network environment) is often important to carrying out the crime successfully. Therefore, we created Criterion 5, whether the attacker checks the environment. In addition, checking the presence of other users (Criterion 3), which is one of the criteria used to evaluate the attacker's discretion, also contributes to answering this question.

3) *How familiar is the attacker with the rogue software he/she is using?* Some attackers attempted to use specific rogue software unsuccessfully. For example, a rogue software having network functionalities that is installed on a target where the

corresponding network port is blocked (Criterion 10) indicates a poor knowledge of the software or a lack of expertise to detect blocked ports. On the contrary, editing the configuration file(s) (Criterion 6) before the installation shows that the attacker is aware of what he/she is doing and is not just reproducing an attack without understanding it. We can also link to the question the fact that the attacker changed the system to make the rogue software work (Criterion 7).

4) *Is the attacker protecting the compromised honeypot?* Even once an attacker compromises a honeypot, brute force attacks continue to be launched against it. Since the attacker gained access through a brute force attack, it means that the attacker account credentials (login and password) are somehow weak. Therefore, we introduced two criteria to take in account if the attacker was protecting his or her account, changing the password and creating a new user account (Criteria 8 and 9).

These ten criteria are evaluated with a value between 0 and 1 and summed over the entire period of the experiment, leading to an overall attacker' skill level between 0 and 10 for the global attacker's activity. We are able to identify all the sessions from a specific attacker through the profile analysis described previously (Section 4.3.1.1).

Table 2 lists the different criteria with their definitions and the method used to evaluate them.

Criteria 1, 2 and 3 are evaluated using ratios between numbers of sessions involving a given attacker. More precisely, we take into account whether these actions have been performed during each session or not, because the corresponding action(s) are more efficient if they are performed every time the attacker connects to the honeypot. For example, to be the most efficiently hidden, an attacker needs to hide at each session, or if the attacker wants to verify who else is using the target, the attacker has to check the presence of other users at every session. The score for these criteria is the ratio of the number sessions where the attacker fulfilled the criteria over the total number of sessions, leading to any number between 0 and 1.

Table 2. Attacker Skills Criteria

Criterion ID	Criteria Name	Definition	Assessment
1	Hide	Deletion of log files or deactivation of logging	Ratio of the number of sessions where the attacker hid
2	Restore Deleted Files	Restoration of deleted files	Ratio of the number of sessions where deleted files were restored
3	Check Presence	Observation of users	Ratio of the number of sessions where presence has been checked
4	Delete Downloaded File	Deletion of downloaded rogue software after usage	0 if downloaded file is not deleted in any session, 1 otherwise
5	Check System	Observation of system configuration or state	0 if the system has never been checked in any session, 1 otherwise
6	Edit Configuration File	Edition of rogue software configuration file	0 if configuration file is not edited in any session, 1 otherwise
7	Change System	Modification of system configuration or state to have a working attack	0 if the system has never been modified in any session, 1 otherwise
8	Change Password	Change user password	0 if the password was never changed in any session, 1 otherwise
9	Create New User	Creation of new user	0 if no new user is created in any session, 1 otherwise
10	Rogue Software Adequacy	Adequacy of rogue software	0 if less than half of the installed rogue software is adequate, 1 otherwise

Most other criteria (4, 5, 6, 7, 8, and 9) are considered for the global activity of the attacker. The criterion is evaluated as either 0 or 1, depending if the attacker did the corresponding action(s) at least once among all sessions. Indeed, these actions do not need to be performed in each session to give an indication of the attacker's skill level. The remaining criterion is the rogue software adequacy (Criterion 10). The idea is to assess whether the rogue software requirements match the target configuration. For each rogue software, we know its type from the rogue software analysis (see Section 0) and the honeypot on which it has been run. Depending on the honeypot configuration, the rogue software is noted as adequate or not. We attribute 1 if more than half of the attacker's rogue software is adequate and 0 otherwise. This choice does not penalize too much an attacker who installed inadequate software, realizes it, and installs another more adapted one. And gives 0 to an attacker that insisted in installing several inadequate rogue software.

4.3.1.3 Attack Tools Identification

To identify the tools installed by attackers to commit crimes from the compromised honeypots, we developed an application in Java that automates a part of the analysis and stores the results in a database. This application performs two tasks, the static analysis and the dynamic analysis of a given attack tool.

Information obtained by static analysis includes: the name of the file that was downloaded on the honeypot (*filename*), the URL address from where it was downloaded (*url_origin*), the IP address corresponding to the URL (*ip_origin*), the time of the download (*download_time*), the session in which the software was downloaded (*session_id*), the file type (*file_info*), the file size (*filesize*), the number of files after unpacking if it is an archive (*nb_files*), the *roguesoft_id* of other rogue software already in the database that are similar (*matching_files*) and finally the files affected by the rogue software during execution and that were identified to be likely used as configuration files (*configuration_files*).

Information obtained by dynamic analysis includes: the ports that are being opened during an execution of the rogue software (*open_ports*), the files that are being accessed (*files_open*), the log of the network traffic generated (*log_iptables*), and the processes created (*new_processes*).

This dynamic analysis consists of replaying attacker sessions involving the rogue software in a sandbox identical to the honeypot environment. At this point, human intervention is required to reproduce the sequence of commands entered by the attacker after he/she downloaded the rogue software. A full automation of the execution would have been possible if attackers were not making any mistakes while typing commands and if keystrokes were perfectly recorded. However, having to execute rogue software manually helps identify them. Once we have reproduced the attack, the modifications to the sandbox resulting from the rogue software execution are saved and analyzed.

The information gathered statically and dynamically is reviewed to identify the type of rogue software. Ports opened during the execution give information on the protocol linked to the rogue software acting as a server. If no port has been opened, then this type of information can be retrieved from the network traffic traces in case the rogue software was used as a client. The files accessed during execution are usually a relevant clue to identify a specific type of rogue software. Similarly, a particular process called during execution can reveal the type of malware.

It can happen that rogue software belongs to several categories (e.g., an archive containing several pieces of malware). In those cases, by replaying attackers actions, we only consider the observed usage(s) of the rogue software to determine the category, or categories, it belongs to.

Different versions of rogue software are identified as distinct rogue software through the static analysis. However, they will belong to the same category if the dynamic analysis reveals that the attacker's purpose while using it was the same.

The same rogue software used with different configurations are also distinguished. The *matching_files* field in the static analysis helps identify identical or similar files used by attackers.

The information gathered during this analysis process about attackers' tools and their behavior are stored in a database. This database helps to understand the scope of attacks. We note that unlike traditional malware analysis services like VirusTotal [VIR15], our approach is not exclusively based on a comparison with a repository of already known dangerous files. Running both static and dynamic analyses often reveals the purpose of the attack tools instead of only giving a general idea about the dangerousness of the file.

4.4 Results and Analysis

This section first presents overall results, then focuses on the attacks characteristics. As previously mentioned, three types of honeypots were deployed: HP1 is fully functional, HP2 has the IRC port blocked for incoming and outgoing traffic, and HP3 has all ports blocked besides HTTP, HTTPS, FTP, DNS, and SSH. Moreover, the network is configured so that only attackers who gained access to HP1 can reach HP2, and once access is granted to HP2 attackers can reach HP3.

4.4.1 Results

The experiment was conducted from May 17, 2010 to October 31, 2010. At the beginning of each month, the honeypots were recycled. During the experiment, 106 honeypots were successfully deployed, 56 HP1, 30 HP2 and 20 HP3. Table 3 shows the distribution of sessions per honeypot type and the number of computer compromise sessions (non-empty sessions).

Table 3. Distribution of Session per Honeypot Type

	Number of sessions	**Number of non-empty sessions**
All honeypots	312	211 (68%)
HP1	160	110 (69%)
HP2	105	74 (70%)
HP3	47	27 (57%)

4.4.2 Attackers' Origin

We first considered the origin of the attackers by analysing IP addresses related information: the IP address itself, the AS number related to this IP and the country of origin.

It has been shown that attackers can be divided in different groups: those who conduct reconnaissance by brute-force scanning hosts, and those who attempt to actually gain control of hosts by compromising them [ALA06]. We verified this statement by comparing the origins of both groups in the case of the SSH experiment. Figure 10 shows the percentage of attackers who performed brute-force scanning but who did not compromise the computer (i.e., even if they found the login/password combination, they didn't use a shell), the percentage who did both, and the percentage who only compromised the target (i.e., they knew the login/password combination and got a shell at the first login attempt). Figure 10.a shows the results when we assume that a single IP address is associated with a single attacker. Figure 10.b shows the results when we assume that the attacker could have used another IP address that belongs to the same organization, identified by its AS number. Both figures confirm previous results because the majority of attackers are clearly divided into two groups and only a small fraction of attackers are associated with both brute-force scanning and compromising hosts.

We also note that these results should be taken with caution because a given attacker could easily appear to connect from different IP addresses, different AS numbers and different countries.

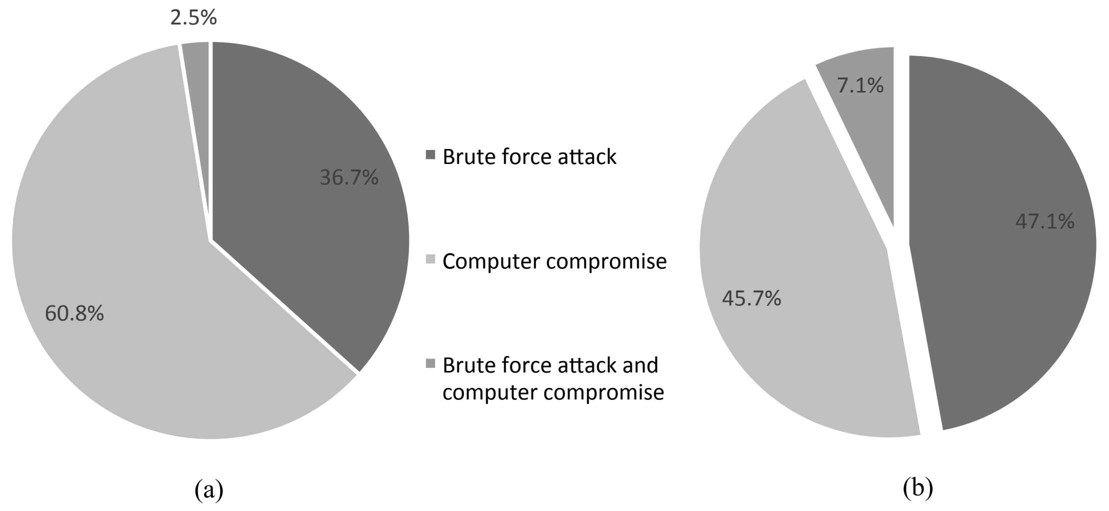


Figure 10. IP Addresses (a) and ASN (b) Distributions of Attackers

4.4.3 Attackers' Characteristics

In this section we introduce the results about attackers' differentiations and attackers' skills. Once the attacker has compromised the honeypot, we used keystrokes profiles and the attackers keystroke logs to characterize the attack.

4.4.3.1 Identifying Attackers Through Keystroke Profile

Once the attacker compromised the target, the interactive shell sessions provide keystrokes to build profiles and keylogs. We applied a new approach of keystrokes analysis, discussed in Section 4.3.1.1, to differentiate attackers.

Figure 11 and Table 4 introduce an example of keystrokes analysis for three sessions of a given honeypot. According to the IP addresses, the attacks come from the USA (session 1), Israel (session 2) and Romania (session 3). Figure 11 shows the delays between pairs of keystrokes as a function of the characters composing the pair. Values in Table 4 indicate the average delays interval between matching pairs of keystrokes for the three sessions. The smaller the value is, the closer the profiles are. In light grey we observe that profiles 1 and 2 are closer than profiles 1 and 3 or 2 and 3 are. Thus, the attacker from session 3 is likely to be different from the attacker in session 1 and 2.

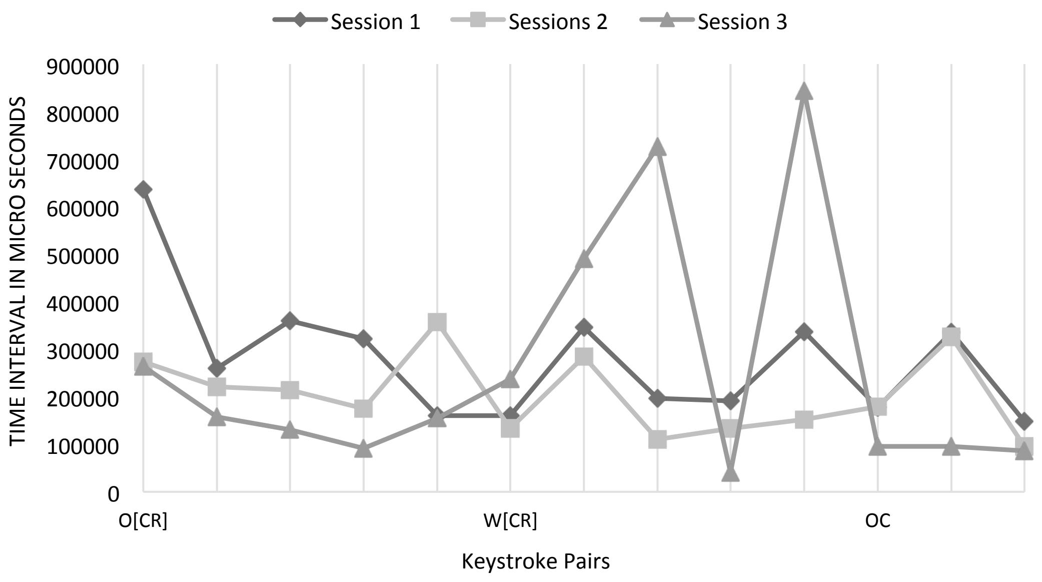


Figure 11. Time Intervals between Keystrokes for a Sample of Matching Pairs

In this case, further manual analysis of the attackers' key logs sessions allowed distinguishing two different attackers: in session 1 and 2 the attacker used identical rogue software and executed the command "`unset HISTFILE`" at the beginning of both sessions whereas no similar actions were performed in session 3.This method allowed to significantly improve the estimation of the number of attackers for each honeypot. Starting from 73 distinct IP addresses exploiting the honeypots over the five and a half months, we concluded that these were actually 39 different attackers. We note that we favored expert judgment over a traditional clustering technique to analyze those results, because the lack of ground truth prevented us from identifying correct thresholds to build clusters from the different characteristics reviewed.

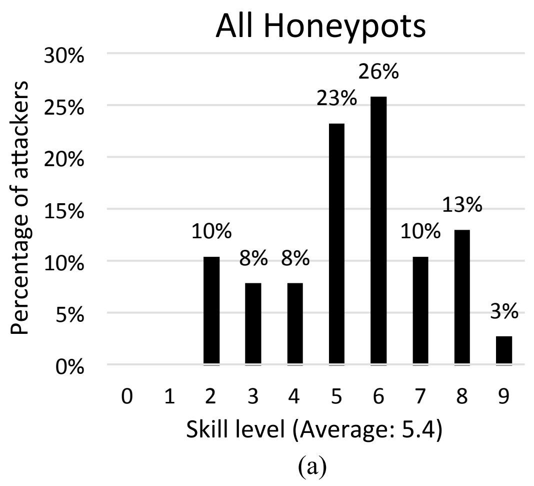
Table 4. Table of average keystroke delays

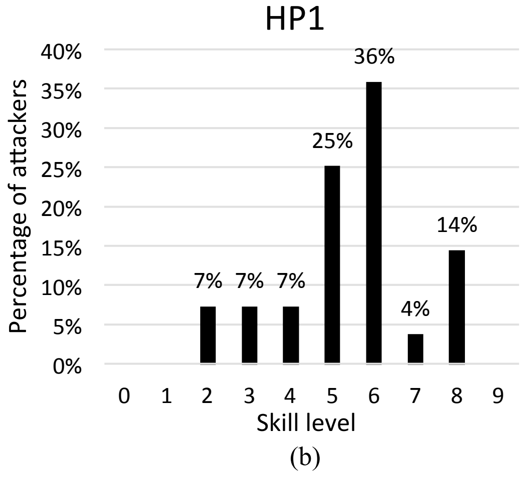
Session ID	**1 (USA)**	**2 (Israel)**	**3 (Romania)**
1 (USA)	0	**57.7 (121 matching pairs of keystrokes)**	**197.5 (15 matching pairs of keystrokes)**
2 (Israel)	X	0	**163.8 (20 matching pairs of keystrokes)**
3 (Romania)	X	X	0

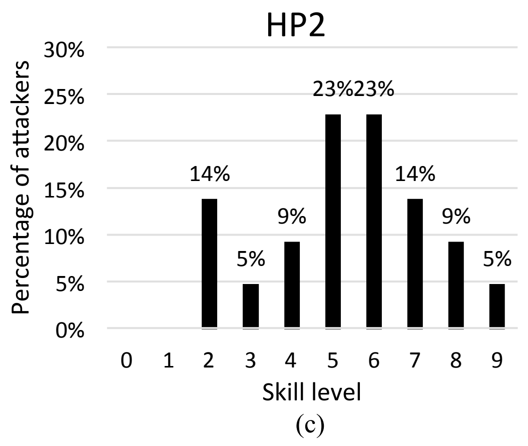
4.4.3.2 Characteristics of Attacker Sessions based on Attacker Skills

Based on the skill ranking technique introduced in Section 4.2, Figure 12 shows the distribution of attackers' skill levels from 0 (no skill criterion observed) to 10 (all skill criteria observed) for the different honeypots. Figure 12 seems to indicate that the configuration of the honeypot is not linked to the attackers compromising it. This leads to the conclusion that deploying honeypots with very different configurations does not ensure the observation of different types of attackers.

Having defined attackers' skills using ten criteria, we show the percentage of attackers who conducted each action in Figure 13. We see that more than 90% of the attackers checked the system and the presence of other users on the target. Almost 80% of the attackers changed the account password. Less than 60% hid their actions.







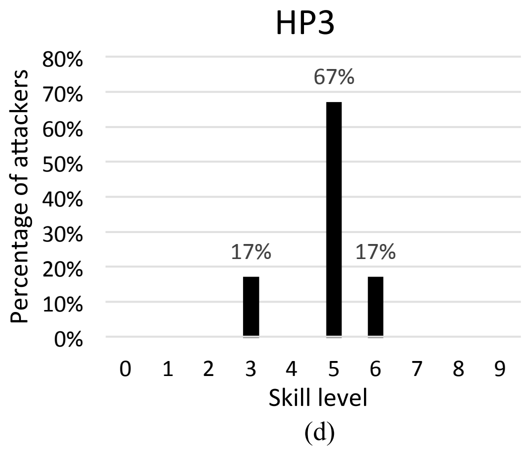


Figure 12. Distribution of attackers by skill levels

This number is surprising as we expected that one of the attackers' main goals is to remain undetected. Less than 60% of the attackers installed correctly rogue software. Note also that new user accounts are created for only about 15% of the attackers.

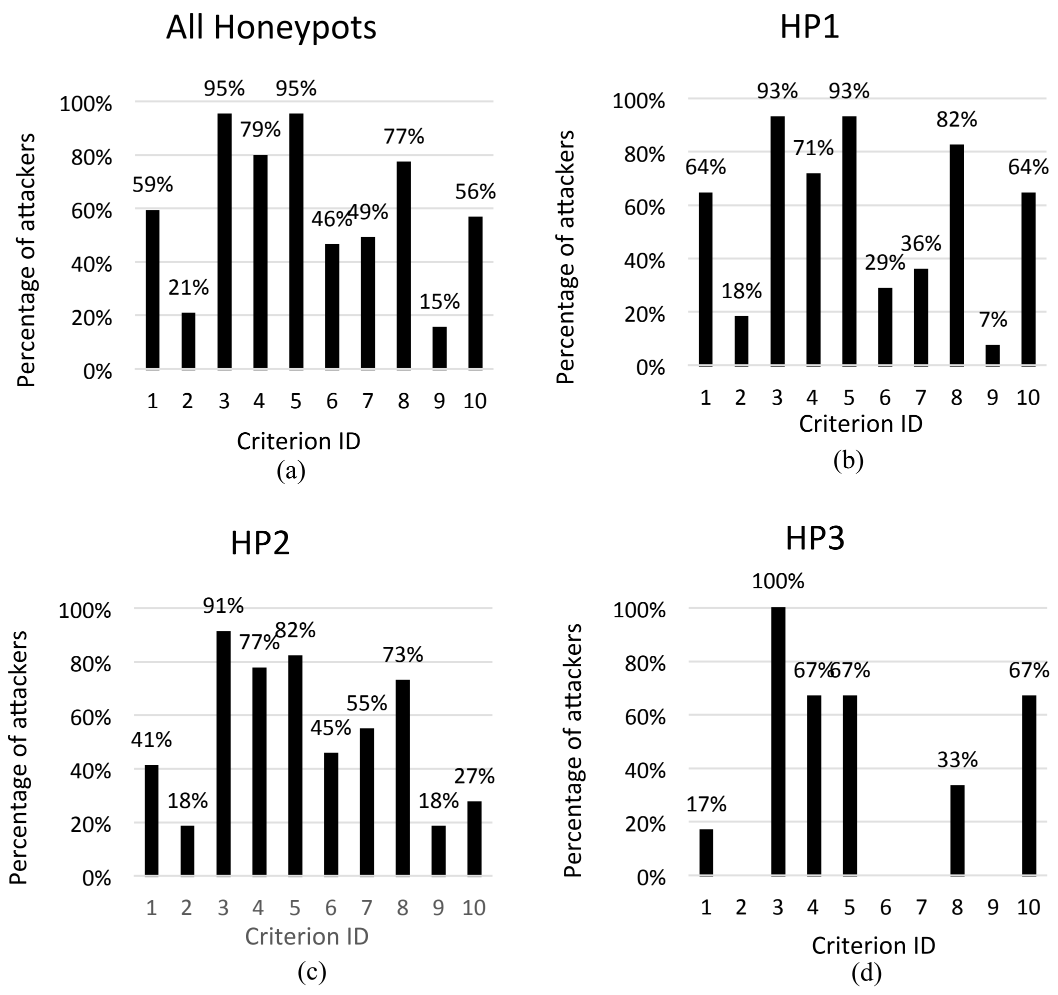


Figure 13. Percentages of Attackers Fulfilling Skill Criteria

Most of the attackers used copy and pasted commands in their session. To further expand the analysis, Figure 14 represents the percentage of copied/pasted commands by skill level. One can observe that the most advanced attackers are more likely to paste directly command lines. One might expect that copied and pasted commands comes from less skilled attackers (a.k.a. script kiddies). However, to limit their visibility, it makes sense that skilled attackers prepare their attacks and limit the amount of time on the compromised host launching the crime via the copy and paste command. However, another interpretation of this result could be that some attackers are simply using copied and pasted commands to execute a sophisticated attack without understanding it.

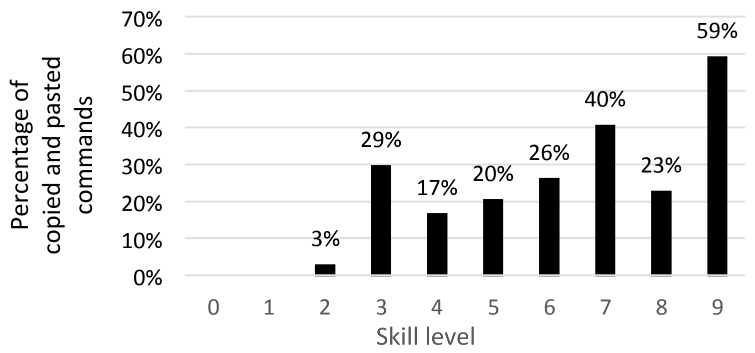
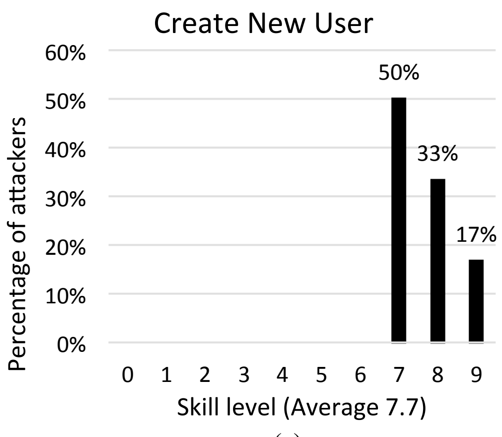


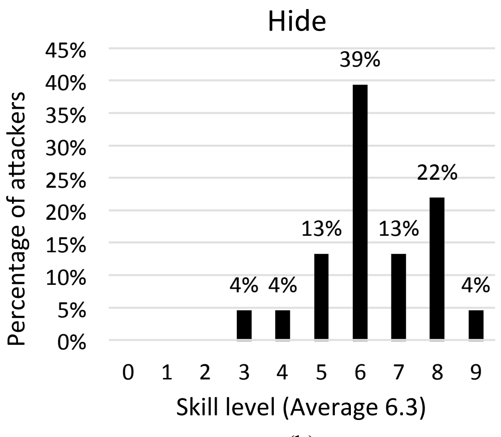
Figure 14. Percentages of Copied/Pasted Commands per Skill Level

To identify what actions are the most representative of the attacker's skill level, we calculate the skill distributions for groups of attackers who were observed performing specific actions. Figure 15 show the distributions for creating a new user (Figure 15.a), hiding his/her actions (Figure 15.b), changing his or her password (Figure 15.c), correctly launching rogue software (Figure 15.d) and finally checking the presence of other users (Figure 15.e).

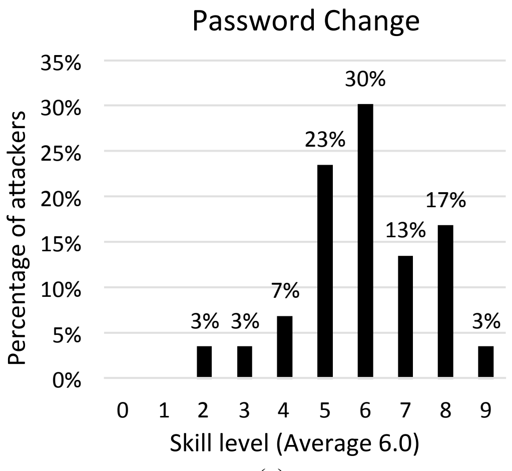
On Figure 15.a, we clearly notice that the population of attackers that created a new user has a higher average skill level (7.7) than the entire attacker population average (5.4) as shown on Figure 12.a. This observation increases our confidence in selecting "Create new user" as a criterion for the skill level. As shown on Figure 15.b, Figure 15.c and Figure 15.d, similar cases can be observed to a lesser extent for a majority of the other criteria. However, Figure 15.e shows that the average skill level of attackers that checked presence (5.5) is very close to the global average. This, as well as the fact that 95% of the attackers performed this action, shows that this action is performed by any type of attacker.



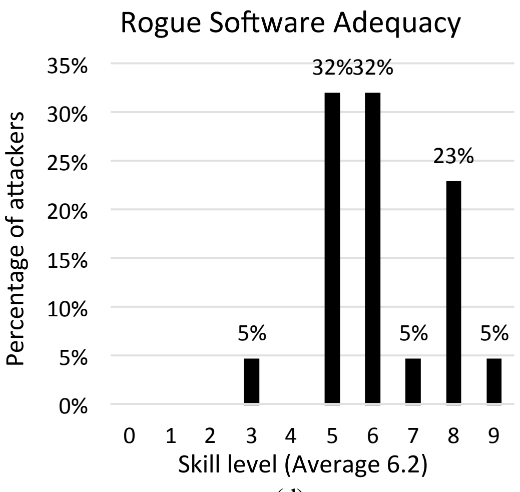
(a)



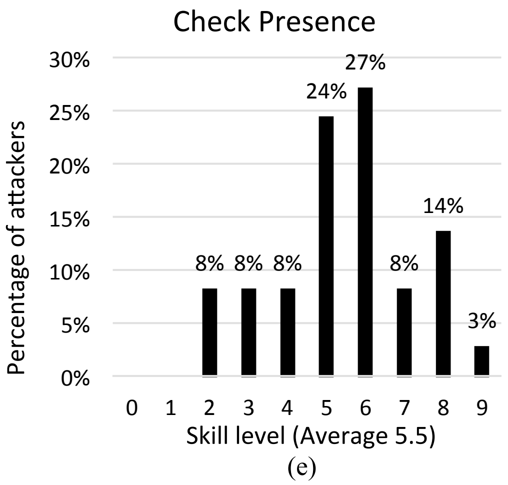
(b)



(c)



(d)



(e)

Figure 15. Distributions of Attackers per Skill Level Specific Actions

4.4.4 Compromise Purpose

This section introduces the results about the third research question that motivated this study, the compromise purpose. Table 5 presents the list of categories of rogue software resulting from the analysis described in Section 0. This analysis allowed concluding on the type of software for 90% of the rogue software collected. The remaining 10% could not be identified because they could not be correctly extracted from the network traces, either because of a prematurely stopped download or because of a broken network trace.

Table 5. Type of file or rogue software used by attackers

IRCbot	Software used to enroll a compromised host in a botnet that uses the IRC protocol to communicate
Bouncer IRC	IP address spoofing software for the IRC protocol
Backdoor	Software allowing the attacker to come back on the host by another means
Scanner	IP address or port scanning tool to look for potential vulnerability(ies)
Attack tools	File(s) download by the attacker to assist him or her during the attack (File editor, hiding script…)
Flooder	Software used to flood other IP address(es) with a large volume of packets (denial of service)
Privilege escalation	Tool to try to gain root privileges on the host
Download testing files	Big files used by attacker to test the speed of the Internet connection of the host
Library	Libraries added by attackers to make other rogue software work.
VoIP exploit	Tool to exploit vulnerabilities in VoIP software
Audio server	Server to stream Internet radios

We observe that this list contains many categories that were expected based on the results found in [BER09], including IRC bots and IRC bouncers. In addition, we found some unexpected software, including a server for audio streaming.

By replaying every attack in a sandbox, we were able to gain an understanding of the compromise purpose. In Figure 16, we see that installing IRC bot and IRC bouncers are still the most frequently occurring software categories. It is also interesting that the third most common software was for attackers to test the network bandwidth or the connectivity by downloading test files.

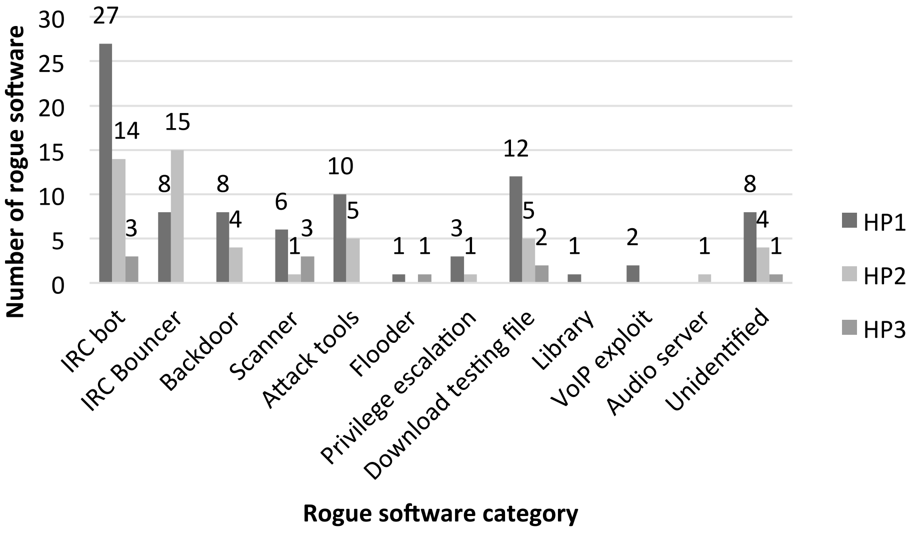


Figure 16. Number of Downloaded Files per Category and Honeypot Type

In order to compare the rogue software installed per honeypot type, we regrouped them into two groups as shown in Table 6:

- Coordination tools: This group includes IRCbot, bouncer IRC and backdoor
- The attack tools: This group includes the other rogue software categories shown in Table 6 (besides "Audio server" and "Unidentified").

Using these two groups we applied a χ-square test to compare the differences in the number of rogue software downloads between the honeypot types HP1 and HP2. The low number of rogue software installed on honeypot type HP3 did not permit us to include it in this analysis.

Table 6. Rogue software groups per honeypot types

Software group	HP1	HP2	HP3
Coordination Tools	43	33	3
Attack Tools	35	12	6

With a p-value of 0.045, the χ-square test shows that the number of downloaded rogue software, for the two considered groups, is not independent of the type of honeypot (when considering HP1 and HP2).

One of the goals of our experiment was to study the attacker's behavior over a long period of time. The attackers retain access to their honeypot for up to one month. It allowed us to observe the time intervals between sessions occuring on a same honeypot after the initial deployment triggered by a successful brute-force attack. We found that on average, attackers exploit the brute-forced host after 12 hours. However, for 30% of the honeypots, the fourth attack session occurred, on average, 13 days after the deployment. This result is important to justify the extended period of time that we allocated to honeypot exploitation. Limiting honeypot life time to 24 hours [BER09, RAM07] would have significantly reduced the scope of our data collection. We also noted that the first session corresponds to the installation and the execution of a malicious program for 39% of the honeypots studied.

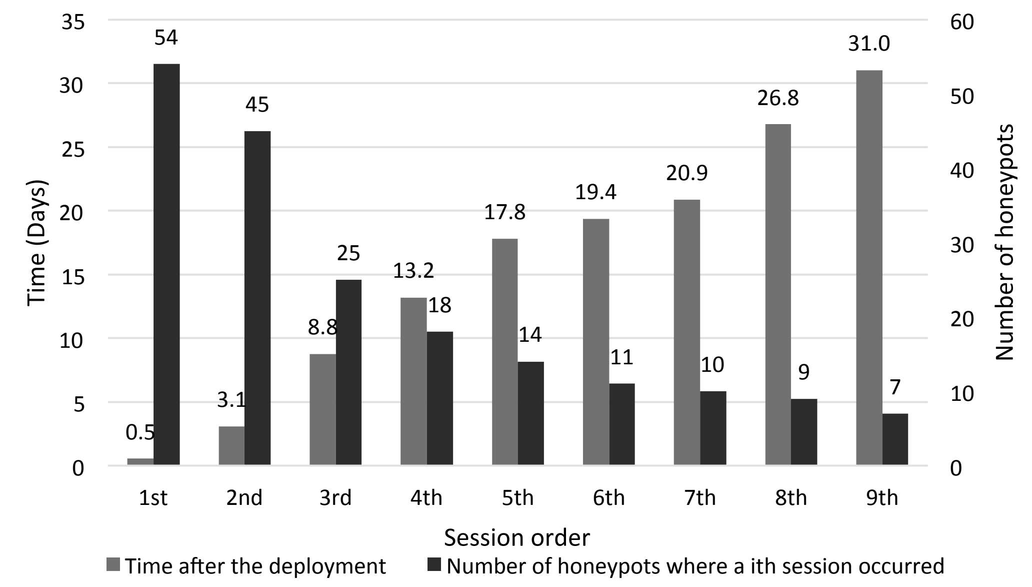


Figure 17. Average Time between Sessions and Number of Honeypots

Another objective was to be able to determine whether or not and how often compromised hosts are shared among attackers. For the 60 honeypots that were targeted by the 39 attackers identified in Section 4.4.3, 20% of them shared honeypots. Nine honeypots (15%) were exploited by more than one attacker. We found that seven honeypots had been exploited by two different attackers, one honeypot by three different attackers and one honeypot by five different attackers. This raises the important issue about how the access to honeypots was shared and why. Even though 77% of the attackers changed the password, 15% shared the access with at least one other attacker.

Figure 18 shows the timeline of activity for eight of the nine honeypots that were shared. Each diamond on Figure 18 represents a session and each shade of grey a unique attacker. These eight honeypots present the particularity of all having an IRC bot installed during their lifetime. IRC logs were analyzed from these bots in order to evaluate the influence of IRC as a vector of transmission of honeypots credentials.

We discovered that five of the eight honeypots used two identical IRC servers. We also found credential information being shared on IRC channels.

These results are important to show that attackers seem to be organized to compromise and exploit honeypots. However, we note that we observed this type of shared access for only a minority of honeypots.

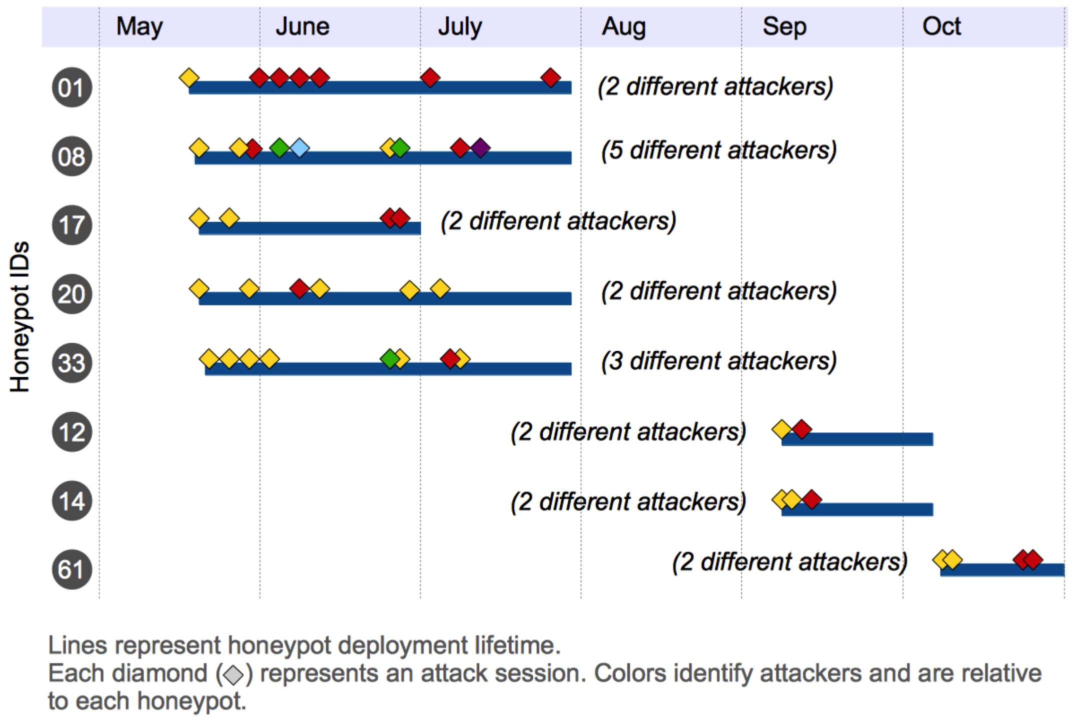


Figure 18. Timeline of Attacker Access to the Shared Honeypots

4.5 Limitations

The limitations of the experiment are the following. First, since the attackers and computer-focused crimes evolve, we expect these results will change over time.

Second, we have only looked at an SSH entry point with different network configurations. We will need to consider other entry points as well as other configurations to assess how attackers react to them (e.g., large disk space).

Third, our empirical results cover a single organization. While we do not believe that the location will have had a significant impact on our results since these are targets of opportunity that were found using automated tools to gain access, it would be interesting to replicate our experiment in other organizations to compare the results.

Fourth, the keystroke analysis technique assumes that the path followed by the packets carrying attacker's commands is static and does not change over the course of the attacker sessions. We believe this assumption holds for the majority of attackers, but it would be interesting to validate it.

Fifth, the experimental setup constrained the honeypots to be reset at the beginning of each month. This may have cut off attacks occurring towards the end of the month. Furthermore, the honeypots types (HP1, HP2 and HP3) were not assigned randomly. A better design would randomly allocate a honeypot type every time an attacker obtains a honeypot.

Sixth, we are not sure attackers perceived the network restrictions and tried to correct them in order to set up their backdoor or IRC bot.

Lastly, the number of sessions and honeypots deployed are too low to draw many strong conclusions from this study.

4.6 Conclusions

An empirical research study was conducted in order to gain insights on attackers and the type of crimes launched from the honeypots. The experiment focused on targets of opportunity where attackers face targets with different network capabilities. We introduced novel approaches to characterize the attacker and the attack. In particular, we discussed how to use keystrokes profiles analysis and attacker behavior to have strong evidence that attackers are different. We also introduced criteria to assess attackers' skills. We used a variety of different data to conduct our analysis: IP address, AS routing number, network traffic, key logs, keystrokes profile, attack sequence, and rogue software downloaded.

We found that the main motivation behind the compromise was to install IRC-based botnets. We collected evidence to show that 15% of attackers shared honeypot access with at least one other attacker. This represents nine honeypots, seven of which had their password changed. This reveals that attackers appear to be organized and share credentials. Changing password was a frequent action for attackers (77% of attackers updated the target computer password). Only skilled attackers created a new user account, but most attackers checked the presence of other users. We also noticed that the configuration of the compromised honeypot did not seem to impact the type of attacker attacking it.

Chapter 5 – Are Computer Focused Crimes Impacted by System Configurations?

The study presented in this chapter was published in [SOB12]. It marked the beginning of a collaboration with faculty and students of the Criminology and Criminal Justice Department at the University of Maryland.

5.1 Introduction

In Chapter 4, we listed the assessment of the attackers' reaction to honeypot configuration changes as future work. The experimental setup of the study described in this chapter provides different honeypot configurations. Several limitations were raised concerning the experimental design of the work described in Section 3.2 of Chapter 3:

- Attackers were assigned a honeypot configuration sequentially (HP1 for the first honeypot deployed, HP2 for the second and HP3 for the third). A better design would employ randomization of the honeypot configuration.
- The number of honeypots was too low in order to perform a strong statistical analysis.
- Not all the attackers had access to their honeypot(s) for 30 days since the honeypots were recycled at the beginning of every month.

To address these limitations, the experiment used to collect the data was built using the cybercrime framework described in Chapter 3.

More specifically, we concentrated on crimes committed by attackers who gain access by finding the correct combination username/password on SSH to a computer running UNIX. Once an attacker has access to the computer, he/she can build the attack over a period of 30 days. We focused specifically on the crime(s) the attacker commits, i.e., the attacks launched from the computer the attacker gained access to towards any external computer.

5.2 Research Questions and Hypotheses

In this chapter we discuss computer-focused crimes and address the following research questions:

a) Does the computer configuration impact whether the crime is destructive or not?

b) Does the computer configuration impact whether the target is of choice or opportunity?

c) Does the computer configuration impact whether the attack is coordinated or not?

Network flow records and attackers' keystrokes were used for this study. Network flow records helped identify the difference crimes committed by the attackers. Keystrokes, processed into lists of commands, were used to identify the attackers' actions. Details on these data were provided in Section 3.2.2.4 of Chapter 3.

Many dimensions of a computer-focused crime can be studied. The focus is on three specific dimensions: 1) whether the crime was destructive or not, 2) whether the victim of the crime was a target of choice or a target of opportunity, and 3) whether the attack was coordinated or not. Flooding attacks are characterized as destructive, while scanning activity or brute force attacks are non-destructive. Flooding attacks and phishing campaigns are examples of targets of choice. Scanning activity or brute force attacks can be considered as targets of opportunity. Coordinated attacks can be identified by the exchange of some IRC communication before or during the attack.

This study is divided into two parts. First, we assess the effectiveness of a warning banner in deterring the attackers to launch crimes. More specifically the warning banner from the NIST 800-53's system use notification control (AC-8) recommendation.

Deterrence is based on the fear of threat punishment refraining a criminal to engage in a criminal activity [CUS93, GIB75, PAT87]. Criminal activities involving computer misuse are punished by the Computer Fraud and Abuse Act of 1986 up to 10 years of imprisonment [KER09]. According to [GEE75], deterrence effectiveness depends on the communication mechanism used to inform the offenders of potential detection and punishment. Studies on such mechanisms showed warning signs are an efficient way to carry the deterrence message [CLA97, CUS93]. Mixed results have been observed in the physical world on the deterrent effect of warning signs. A study on unsafe driving indicated that they were effective [RAM00] whereas in [GRE85], they had no effect on cable television signal theft. Several theoretical studies argued on the effectiveness of warning on the Internet. Because of anonymity of the Internet prevents the authorities to identify, locate and prosecute authors of crimes, the deterrence has less effect [BLA01, HAR96]. However, authors of crimes do not have to be identifiable for the deterrence to work [GOO10] as they exhibit a rational decision-making process during attack sessions [PNG09].

Because of their rational decision-making process that could prevent them from engaging in criminal online activities [PNG09], the possible effectiveness of deterrence on the Internet and the effect of warning signs in the physical work, we hypothesize that *attackers will be deterred by the standard NIST banner and as a consequence discouraged from launching crimes from the honeypots.*

Another goal of this study is to empirically assess whether the configuration of the computer compromised by the attacker impacts the type of crime committed. Authors in [CHA11] demonstrated that malicious activities on a computer could be detected by monitoring abnormal resources usage when the user is away from his/her computer. With this study we determine that CPU, network and disk access are computer resources used by malicious software. Since all the instructions interpreted by the CPU reside in the main memory of the computer, all programs executed by a system will reside in memory at some point [COL03], it is safe to assume that malicious activities use memory as well. Not all crimes require the same computer resources; we hypothesize *that attackers will launch different attacks depending on the available system resources (disk, memory and bandwidth).*

To explore the two different hypotheses, we will consider the following computer configurations: 1) whether a warning message was provided to the attacker when he/she gained access to the target computer, 2) the size of the hard drive, 3) the size of the memory, and 4) the size of the bandwidth. For each of these configurations we will study the number of crimes that are (non)-destructive, attacks that are (non)-coordinated, and victims that are targets of choice/opportunity.

5.3 Experimental Design

To answer these questions we used the data collected from October 10, 2011 to April 30, 2012.

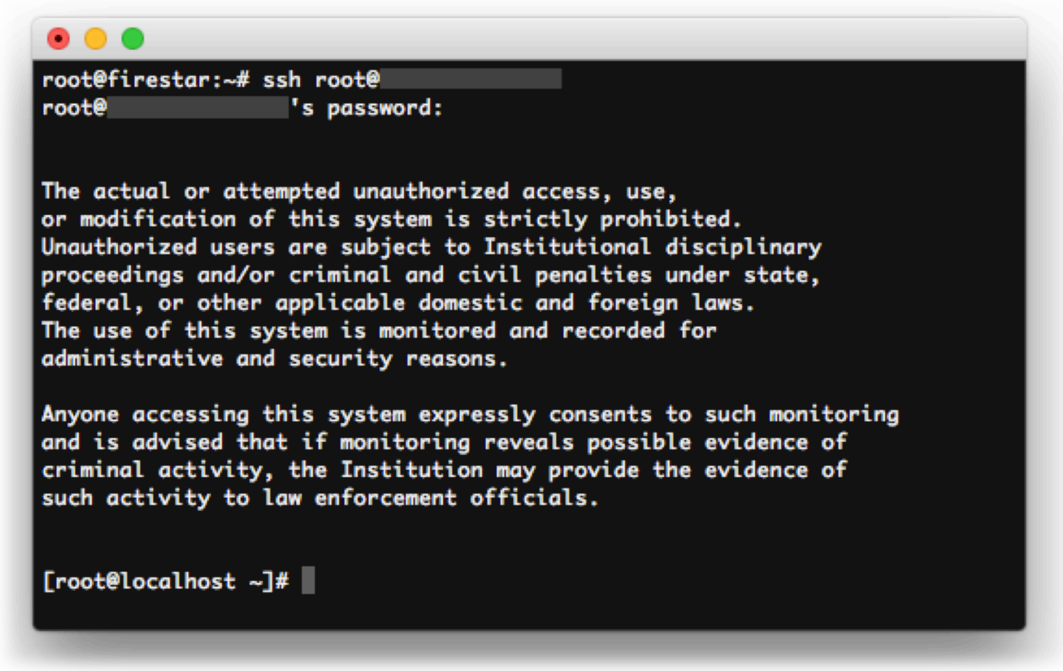
Attackers, who have been identified according to their IP addresses, are randomly attributed one of the sixteen honeypots configurations introduced in Table 7. These configurations combine:

- Low (512 Mbytes) and high (2.25Gbytes) memory space,
- Low (5 Gbytes) and high (30 Gbytes) disk space,
- Low (128 Kbits/s) and high (512 Kbits/s) bandwidth,
- Banner or no banner displayed after a successful SSH login (see Figure 19).

Table 7. Honeypots Configuration

Configuration	Memory	Disk Space	Bandwidth	Banner
1	High	High	High	No Banner
2	Low	High	High	No Banner
3	High	Low	High	No Banner
4	Low	Low	High	No Banner
5	High	High	Low	No Banner
6	Low	High	Low	No Banner
7	High	Low	Low	No Banner
8	Low	Low	Low	No Banner
9	High	High	High	Banner
10	Low	High	High	Banner
11	High	Low	High	Banner
12	Low	Low	High	Banner
13	High	High	Low	Banner
14	Low	High	Low	Banner
15	High	Low	Low	Banner
16	Low	Low	Low	Banner

At the time of the design of this experiment, the low and high values were chosen to reflect a computer with limited hardware resources that could impact the execution or download of programs vs. a powerful machine that will permit fast Internet access and execution of memory and disk consuming programs.



```
root@firestar:~# ssh root@[redacted]
root@[redacted]'s password:


The actual or attempted unauthorized access, use,
or modification of this system is strictly prohibited.
Unauthorized users are subject to Institutional disciplinary
proceedings and/or criminal and civil penalties under state,
federal, or other applicable domestic and foreign laws.
The use of this system is monitored and recorded for
administrative and security reasons.

Anyone accessing this system expressly consents to such monitoring
and is advised that if monitoring reveals possible evidence of
criminal activity, the Institution may provide the evidence of
such activity to law enforcement officials.


[root@localhost ~]#
```

Figure 19. Banner Displayed

5.4 Results

The results come from data collected during the period of October 10, 2011 to April 30, 2012. A total of 939 honeypots were deployed. Figure 20 shows the distribution of the number of each honeypot type deployed.

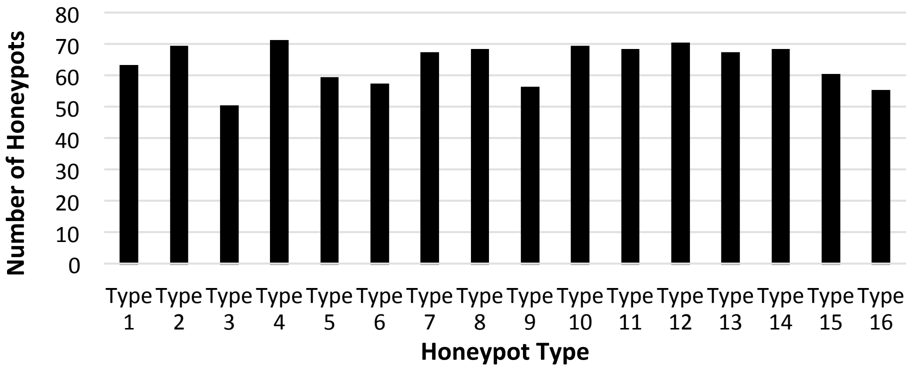


Figure 20. Number of Deployed Honeypots

5.4.1 Analysis of Number of Crimes per Honeypot Type

This chapter focuses on the committed crimes, i.e., attacks launched from the honeypots to some external computer. Specifically, an attacker commits a crime using the compromised honeypot when communicating in a particular way with some targets outside the organization network. These communication patterns are shown in the network flow records, different patterns characterizing different crimes. The period of time during which we see these patterns defines the crime. A total of 245 crimes (i.e., attacks launched from honeypots towards external computers) were committed, representing an average of 0.261 crimes per honeypot deployed, reflecting in fact a large disparity of the number of committed crimes per honeypot. We found that most honeypots contained one crime (15 honeypots) or two crimes (8 honeypots). We also found three honeypots launched respectively 26, 43 and 83 crimes. Figure 21 shows the distribution of the number of honeypots on which were observed a given number of crimes. As expected, we also found some disparity of the number of crimes per type of honeypot deployed as shown in Figure 22.

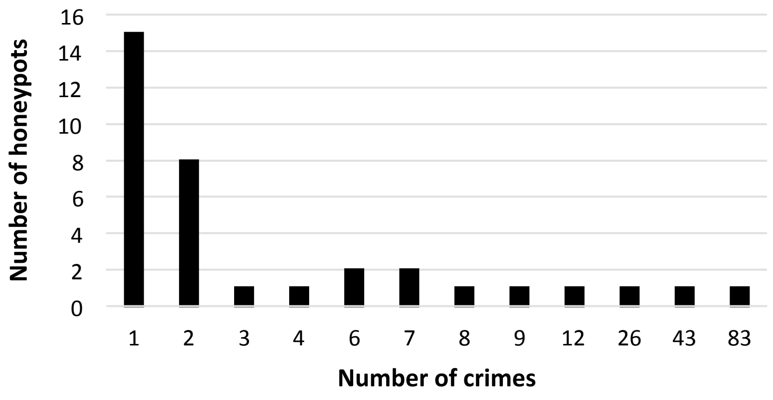


Figure 21. Number of Honeypots and Number of Crimes

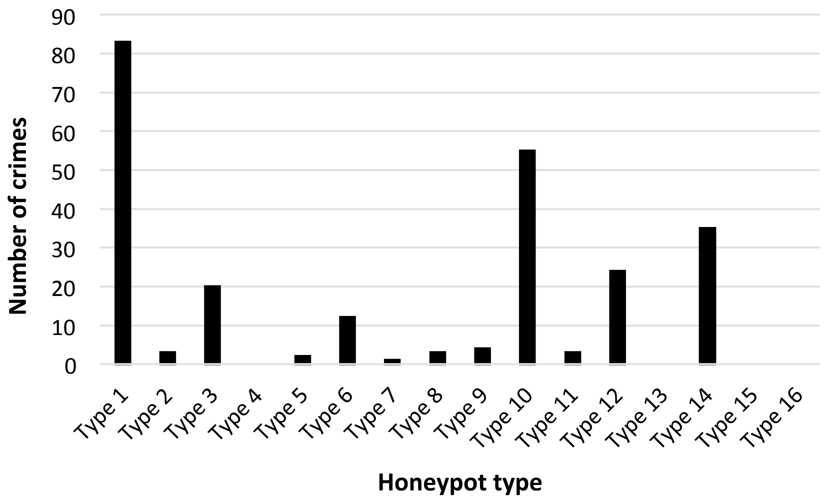


Figure 22. Number of Crimes per Honeypot Type

If we calculate the crime rate (i.e., number of crimes observed divided by the number of honeypots deployed) per 100 honeypots deployed for each honeypot configuration with and without a warning banner, we obtain the results shown in Table 8.

Among the six highest crime rates, three include a warning banner and three do not include one. The highest rate does not include a warning banner but the second one does. Four out of the top six highest crime rates and three out of the four top ones are linked to a high bandwidth. The same applies to high disk space. Among the six highest rates, four are linked to low memory size. But among the four highest ones, only two are linked to low memory size.

Table 8. Crime Rate per Honeypot Type

Memory	Disk Space	Bandwidth	Warning	No Warning
High	High	High	7.14	131.75
Low	High	High	79.71	4.35
High	Low	High	4.41	40.00
Low	Low	High	34.29	0.00
High	High	Low	0.00	3.39
Low	High	Low	51.47	21.05
High	Low	Low	0.00	1.49
Low	Low	Low	0.00	4.41

5.4.2 Classification of Observed Crimes

The 245 observed crimes can be classified into the following five main groups.

C1: Reconnaissance activities (68 instances on 23 honeypots): The attacker downloaded and launched a tool scanning wide ranges of IP addresses to discover specific services such as Secure Shell Servers (SSH). During our investigations we discovered that Web and SSH services were particularly targeted. Each packet has a different source port. However, the destination port is always the same. The network flow pattern is also characterized by the TCP SYN flag [GAD08].

C2: Flooding attacks (161 instances on 15 honeypots): The attacker downloaded and launched a tool generating a large amount of User Datagram Protocol (UDP) packets towards one specific IP address [XU09]. We observed two types of UDP flooding attacks. Different source and destination ports for each packet, and a large number of packets and bytes sent characterize the first type. The second type is characterized by a random high number as source port and the port number of well-known services including SSH (22), DNS (53) and web (80). During the whole attack, packets are sent using the same source port to the same destination port. A simple aggregation of the network flows summarizes this activity in one line. The attack is less visible than the first type where millions of records will be shown for an equivalent attack.

C3: Brute force attacks (5 instances on 2 honeypots): A brute force attack consists in guessing the credentials of an already known service. Several short connections to the targeted service characterize this attack [ALS07]. We detect a brute force attack when we identify several flows with different source ports, the same destination IP address and port, and the TCP flags SYN, FIN and RESET.

C4: Phishing attacks (4 instances on 4 honeypots): Few honeypots have been setup to host fishing websites to steal credentials or other personal information by providing a fake website which looks almost exactly like the legitimate one [IRA08]. A phishing website is coordinated with emails to attract the users to the trap. In addition to the network flows, for phishing attacks, keystrokes logs were analyzed to detect the modification and installation of a phishing website.

C5: SPAM Sender (7 instances on 2 honeypots): The attacker downloaded and launched a tool sending unwanted electronic messages [DHI07]. When unsolicited mails are sent, the network flows show high volume of bytes exchanged between mail servers and the honeypots.

Applying a criminological approach, we will examine in more details three aspects of the crime:

- Whether the crime was destructive or not,
- Whether the crime focused on a target of choice or opportunity,
- Whether the attack was coordinated or not.

We now need to classify the five attack types we have observed based on the crime aspects discussed in this chapter: level of destructiveness, target of choice/opportunity, and coordinated/non-coordinated attack.

5.4.3 Destructiveness

Destructiveness indicates the potential damage that can occur for the victim. The different attacks have been classified depending on their level of destructiveness.

Reconnaissance activities (C1) are not destructive since only one probe packet is sent to a server to determine whether the port is open or not. It is the interest of the attacker to keep the service running since it might be used to gain access to the targeted system later.

Flooding attacks (C2) are typically destructive. The large volume of UDP packets can affect the targeted system as well as the network equipment. The damages can cause a single computer or an entire network to go down.

The login attempts associated to the brute force attack (C3) are not causing any damage to the system. This malicious activity is considered to be non-destructive. The objective is to access a system using a specific service. This service has to be running if the attacker wants to take advantage of it.

A phishing attack (C4) resulting in the deployment of a website is also non-destructive. It aims at collecting users' information.

A SPAM campaign (C5) can quickly overload a mail server. The attackers want to target as many users as possible. It is not in their interest to interrupt the mail delivery of their phishing or commercial messages. Thus this attack is not considered as destructive in this study.

5.4.4 Target of Choice or Opportunity

A target of opportunity is a system that happens to be within a wide range of targets. A target of choice is a particular system or network selected by the attacker.

Reconnaissance activities (C1) are generally observed on wide ranges of IP addresses. The program probing for open ports is usually given entire subnets to analyze targeting several systems and organizations. A brute force attack (C3) often comes after the reconnaissance phase. It uses the results of the scan to determine which hosts to target. Thus, victims of brute force attacks can also be considered to be targets of opportunity.

The victims of UDP flood attacks (C2) can be classified as targets of choice. These attacks only target one specific IP address. The victims of phishing websites (C4) and SPAM campaigns (C5) are also targets of choice. Phishing attacks target one specific organization including banks, mail providers and their customers. Such an attack requires specific knowledge about the organization to deceive its users and obtain credentials. SPAM campaigns, which are either advertising a phishing website or a product, are targeting specific categories of people.

5.4.5 Coordinated/Non-Coordinated Attack

Finding out whether an attack is coordinated is more complicated. We define a coordinated attack when several other hosts are contributing to the same attack. Even though this cannot be determined with certainty, it is possible to identify the honeypots generating Internet Relay Chat (IRC) traffic. IRC is known to be used by Botnet to command and control several compromised hosts [ZHU07]. We postulate that any crime for which we observe IRC traffic right before the start of the crime or during the crime provides evidence of a coordinated attack. On the contrary to levels of destructiveness and targets of choice/opportunity, we cannot systematically link a type of crime to a coordinated/non-coordinated attack. A detailed analysis of each crime is needed.

Table 9 summarizes the crimes observed to the levels of destructiveness and whether the victim is a target of choice or opportunity.

Table 9. Characterization of Observed Crimes

Crime	Destructiveness	Target Choice/Opportunity
Reconnaissance	No	Opportunity
Flooding Attacks	Yes	Choice
Brute force	No	Opportunity
Phishing Website	No	Choice
SPAM	No	Choice

5.5 Warning Banner Impact Analysis

We investigated the impact of the warning banner on the destructiveness of the crime, the type of victim (choice or opportunity) and the type of attack (coordinated or not).

5.5.1 Does the Warning Banner Impact Whether the Crime is Destructive or Not?

In the previous section, we categorized the 245 observed crimes into destructive and non-destructive crimes. In this section, we analyze whether the level of destructiveness is linked to the warning banner.

Table 10 shows the number of destructive and non-destructive crimes observed for the honeypot configurations with a warning banner and without a warning banner.

Table 10. Computer Configuration vs. Level of Destructiveness

	Destructive	Non-Destructive
Warning	65	56
No Warning	96	28

For each computer configuration, we apply a χ-square test to see whether the warning banner impacts whether the crime is destructive or not. We find a P-value of 9.3E-5. So we reject the hypothesis of independence. We conclude that the existence of the warning banner and whether the crime is destructive are not independent.

5.5.2 Does the Warning Banner Impact Whether the Target is of Choice or Opportunity?

In this section, we analyze whether the victim is a target of opportunity or choice is linked to the warning banner.

Table 11 shows the number of observed crimes where the victim is a target of opportunity or choice for honeypots with a warning banner and without a warning banner.

Table 11. Computer Configuration vs. Target of Choice/Opportunity

	Opportunity	Choice
Warning	47	74
No Warning	26	98

We apply a χ-square test to see whether the warning banner impacts whether the victim is a target of opportunity or choice. We find a P-value of 2.2E-3. So we reject the hypothesis of independence. We conclude that the existence of the warning banner and whether the victim is a target of opportunity or choice are not independent.

5.5.3 Does the Warning Banner Impact Whether the Attack is Coordinated or Not?

In this section, we analyze whether the attack is coordinated or not is linked to the existence of a warning banner.

Table 12 shows the number of crimes observed where the attack is coordinated or not for the honeypot configurations with a warning banner and without a warning banner.

Table 12. Computer Configuration vs. Coordinated/Non-Coordinated Attack

	Coordinated	Non-Coordinated
Warning	70	51
No Warning	39	85

We apply a χ-square test to see whether the warning banner impacts whether the attack is coordinated or not. We find a P-value of 3.22E-5. So we can reject the hypothesis of independence. We conclude that the existence of the warning banner and whether the attack is coordinated or not are not independent.

5.6 Computer Resources Impact Analysis

In this section we investigate the impact of the computer configuration on the destructiveness of the crime, the type of victim (choice or opportunity) and the type of attack (coordinated or not).

5.6.1 Do The Computer Resources Impact Whether the Crime is Destructive or Not?

In the previous section we categorized the 245 observed crimes into destructive and non-destructive crimes. In this section, we analyze whether the level of destructiveness is linked to the computer resources.

Table 13 shows the number of observed destructive and non-destructive crimes for each of the computer resources configuration: low/high memory size, low/high disk space, and low/high bandwidth.

Table 13. Computer Configuration vs. Level of Destructiveness

	Destructive	Non-Destructive
Low Memory	89	24
High Memory	72	60
Low Disk Space	69	60
High Disk Space	92	24
Low Bandwidth	28	46
High Bandwidth	133	38

For each computer configuration option, we apply a χ-square test to see whether the computer configuration impacts whether the crime is destructive or not. For each case, we find a P-value between 6.9E-05 (low/high memory size) and 1.5E-09 (low/high bandwidth). So in all cases we reject the hypothesis of independence. We conclude that the computer configuration (for the considered values of memory sizes, disk space and bandwidth) and whether the crime is destructive are not independent.

5.6.2 Does The Computer Configuration Impact Whether the Target is of Choice or Opportunity?

In this section, we analyze whether the victim is a target of opportunity or choice is linked to the computer resources.

Table 14 shows the number of observed crimes where the victim is a target of opportunity or choice for each of the computer resources configurations: low/high memory size, low/high disk space, and low/high bandwidth.

Table 14. Computer Configuration vs. Target of Choice/Opportunity

	Opportunity	Choice
Low Memory	24	89
High Memory	49	83
Low Disk Space	51	78
High Disk Space	22	94
Low Bandwidth	42	32
High Bandwidth	31	140

For each computer configuration option, we apply a χ-square test to see whether the computer configuration impacts whether the victim is a target of opportunity or choice. For each case, we find a P-value between 0.0067 (low/high memory size) and 1.3E-09 (low/high bandwidth). So in all cases we reject the hypothesis of independence. We conclude that the computer configuration (for the considered values of memory size, disk space and bandwidth) and whether the victim is a target of opportunity or choice are not independent.

5.6.3 Does The Computer Configuration Impact Whether The Attack is Coordinated or Not?

In this section, we analyze whether the attacks is coordinated or not is linked to the computer resources.

Table 15 shows the number of observed crimes where the attack is coordinated or not for each of the computer configurations: low/high memory size, low/high disk space, and low/high bandwidth.

Table 15. Computer Configuration vs. Coordinated/Non-Coordinated Attack

	Coordinated	Non-Coordinated
Low Memory	31	82
High Memory	78	54
Low Disk Space	76	53
High Disk Space	33	83
Low Bandwidth	27	47
High Bandwidth	82	89

For each computer configuration option, we apply a χ-square test to see whether the computer configuration impacts whether the attack is coordinated or not. We found a P-value of 0.09 corresponding to low/high bandwidth. The other P-value were 1.7E-07 and 6.7E-07, respectively for low/high disk space and low/high memory size. So we cannot reject the hypothesis of independence for bandwidth size. However, in the other cases, we can reject the hypothesis of independence. We conclude that the computer configuration (for the considered values of memory size and disk space) and whether the attack is coordinated or not are not independent.

5.7 Discussion

The different empirical studies led to the following conclusions:

1. The existence of the warning banner and 1) whether the crime is destructive, 2) whether the victim is a target of opportunity or choice, and 3) whether the attack is coordinated or not, are not independent.
2. The computer configuration (for the considered values of memory size, disk space and bandwidth) and 1) whether the crime is destructive and 2) whether the victim is a target of opportunity or choice, are not independent.
3. The computer configuration (for the considered values of memory size and disk space) and whether the attack is coordinated or not are not independent.

Even though it might seem counter-intuitive that warning banners have no dissuasive effect on the crimes, i.e., attacks launched from the honeypots towards external computers, one explanation might be that attackers who are committing a crime still decided to engage in crimes despites the warning banner. More precisely, these attackers appear to have decided to ignore the banner and downloaded their attack tools, and deployed the attack and finally launched it. So, if banners do not have an effect on the crimes, we might expect them to have an effect on the attacks. Some attackers who would have been tempted to launch an attack might rethink this and stop the attack. Such behavior has been confirmed in another study on the deterrent effect of the warning banner, published in [MAI14], where we showed that when looking at the attack sessions, the warning banner does reduce their duration.

The result regarding high bandwidth was expected. These results confirm that attackers are indeed searching for bandwidth. The fact that low disk space favors non-destructive crimes should be understood as non-destructives crimes being committed independently on the disk space.

The results regarding the memory size are more intriguing. We should not conclude that attackers are interested in low memory size. Instead these results show that attackers are much more interested by bandwidth and disk space than they are by memory size.

This chapter presents some empirically driven conclusions that help to identify the attack threat. Additional studies are needed to revisit/confirm the conclusions we found.

5.8 Limitations

One limitation of this study is the location of the data collected. The dataset was obtained by honeypots deployed at the University of Maryland. It is important to replicate these experiments at different locations and at different times.

A time period of 30 days was provided to attackers for developing and launching their attacks. A study should focus on this duration and its impact on the observed crimes.

The honeypots did not contain specific "honey" to attract attackers. They were basic computers with various configurations of disk space, bandwidth and memory size. A study including different types of "honey" would be interesting to see whether similar conclusions can be derived.

Attackers gained access to the honeypots using SSH as an entry point. Providing such entry point might favor particular attackers and attacks. It would be interesting to compare the results we obtained with outcomes of studies that provided other access points to attackers.

The warning banner also presents several limitations. We have no guarantee that attackers read the warning banner or understood it since the message displayed is in English. Attackers from non-English speaking countries may not understand the message announcing the punishment threat. In order to prevent the attackers from detecting the nature of the compromised hosts, the honeypots behavior should be close to a regular UMD operated system. The means to convey the warning, i.e. the banner following the NIST recommendation 800-53, was chosen based on the UMD policy and not reveal the nature of the honeypots. For example, a banner announcing counter attack threats would be against the UMD policy, unlawful and suspicious.

The present study does not provide any insight on the attacker. We concede that we cannot guarantee that each attacker using the framework is human. Conversely, we cannot guarantee that each attacker using the framework is an automated bot.

5.9 Conclusions

This chapter focused on three specific dimensions of computer focused crimes: 1) whether the crime was destructive or not, 2) whether the victim of the crime was a target of choice or a target of opportunity, and 3) whether the attack was coordinated or not. We empirically assessed whether the configuration of the computer compromised by the attacker impacts the type of crime committed. We considered the following computer configurations: 1) whether a warning message was provided to the attacker when he/she gained access to the target computer, 2) the size of the hard drive, 3) the size of the memory, and 4) the size of the bandwidth. For each of these configurations, we studied the number of crimes that are (non)-destructive, attacks that are (non)-coordinated, and victims that are targets of choice/opportunity.

The three empirical studies led to the following conclusions.

1. The existence of the warning banner and 1) whether the crime is destructive, 2) whether the victim is a target of opportunity or choice, and 3) whether the attack is coordinated or not, are not independent.

2. The computer configuration (for the considered values of memory size, disk space and bandwidth) and 1) whether the crime is destructive and 2) whether the victim is a target of opportunity or choice, are not independent.

3. The computer configuration (for the considered values of memory size and disk space) and whether the attack is coordinated or not are not independent.

Chapter 6 – Are Computer Focused Crimes Impacted By Surveillance Warning Banners or Surveillance Tools?

6.1 Introduction

In Chapter 5, we studied the impact on the crimes committed by attackers on different honeypot configurations as well as the effect of one aspect of deterrence: a warning banner. In this chapter we focus on another aspect of deterrence: surveillance.

6.2 Research Question and Hypothesis

We investigate one aspect of cybercrime deterrence: the announcement and existence of surveillance mechanisms. More specifically, we ask the following research question:

> *Are computer focused crimes (i.e., after an attacker gains unauthorized access to a computer, the use of this computer to launch an attack towards an external target) impacted by a surveillance warning banner and/or surveillance tools?*

We analyzed the following data: 1) network flow records to identify the different crimes committed by the attackers, and 2) the keystrokes processed into lists of commands to study the attackers' actions. Details on these data were provided in Section 3.2.2.4 of Chapter 3.

The authors in [CLA97] introduce three different types of surveillance mechanisms aimed at preventing crime by increasing offenders' perception of threat detection. Formal surveillance is when a person or a system is dedicated to surveillance. Dedicated individuals such as cops or surveillance technologies such as closed-circuit television (CCTV) or monitoring tools deployed by IT security professional, are examples of formal surveillance. Individuals perform natural surveillance during their daily activities whereas a designated employee such as a parking attendant carries out surveillance by place managers.

In our study, we will focus on formal surveillance. A review of the research on the effectiveness of formal surveillance in the physical work showed mixed results. Formal surveillance reduced the number of car thefts in guarded parking lots [LAY92, HES95, BAR96] whereas CCTV was ineffective in preventing violent and property crimes in city centers, public housing communities and public transportation facilities [WEL08].

The security community has been claiming that deterrence does not work on the Internet. Unfortunately, such a claim is rarely accompanied by empirical data to support it. This work allows for an investigation of this claim directly by focusing on the announcement and existence of surveillance mechanisms on a compromised computer system.

A study has shown that awareness of surveillance tools on employees' computers reduces the intent of system misuse by employees [ARC09]. Based on this result we hypothesize that a banner announcing surveillance banner and a surveillance tool will discourage attackers from using the compromised system to launch crimes.

We present findings from a field experiment we conducted over a period of 19 months, which is concentrated on computer-focused crimes [FUR02]. The goal of this chapter is to assess whether these crimes are impacted by a surveillance warning banner and surveillance tools.

The impact of a surveillance banner or surveillance tools is analyzed in terms of the following metrics: the number of crimes committed, the timing of the first crime related to the timing of initial compromise, the temporal distribution for all crimes following an initial compromise, and the specific crime rates pertaining to the most frequently observed crimes.

For the purpose of this experiment, a farm of four honeypots were randomly assigned to attackers as follows: Type 0) no surveillance banner nor tools, Type 1) a surveillance banner but no surveillance tools, Type 2) no surveillance banner but surveillance tools, and Type 3) a surveillance banner and surveillance tools.

6.3 Experimental Design

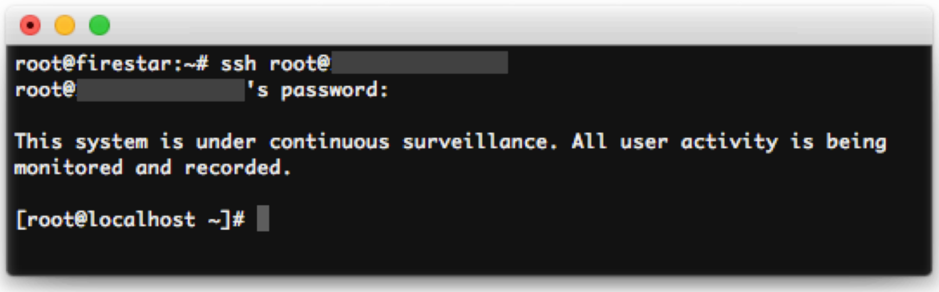
Attackers, who have been identified according to their IP addresses, are randomly attributed one of the configuration types listed in Table 16. The configuration randomly assigned to the honeypot container involves a two (banner vs no banner) x two (processes vs no processes) design:

- The display of a banner after a SSH login informing the user that the system is under surveillance (Figure 23),
- The presence of surveillance tool processes (Figure 24)

Figure 24 shows two different surveillance processes. One is called *zabbix_agentd*, which is the agent of Zabbix an open source monitoring solution. The other is a script named "*monitor.*" "*monitor*" updates and saves a file every minute containing the disk usage, the system uptime, the available memory, the users' logins and the running processes. The attacker has access to these files.

Table 16. Honeypot Configuration Types

Honeypot Type	Surveillance Banner	Surveillance Processes
0	No	No
1	Yes	No
2	No	Yes
3	Yes	Yes

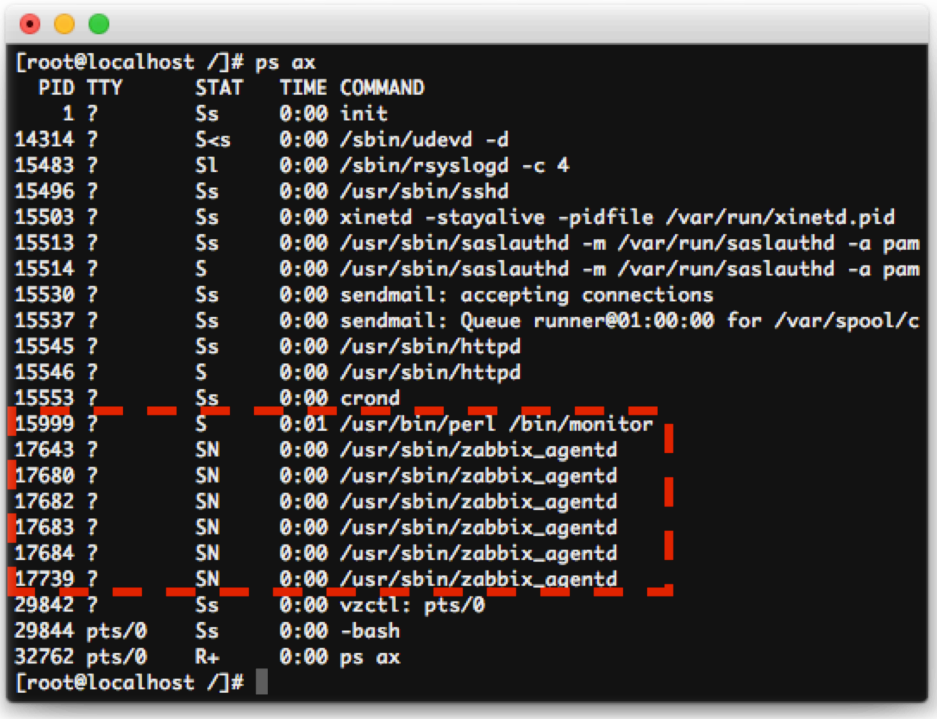


```
root@firestar:~# ssh root@[redacted]
root@[redacted]'s password:

This system is under continuous surveillance. All user activity is being
monitored and recorded.

[root@localhost ~]#
```

Figure 23. Surveillance Banner Displayed



```
[root@localhost /]# ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 init
14314 ?        S<s    0:00 /sbin/udevd -d
15483 ?        Sl     0:00 /sbin/rsyslogd -c 4
15496 ?        Ss     0:00 /usr/sbin/sshd
15503 ?        Ss     0:00 xinetd -stayalive -pidfile /var/run/xinetd.pid
15513 ?        Ss     0:00 /usr/sbin/saslauthd -m /var/run/saslauthd -a pam
15514 ?        S      0:00 /usr/sbin/saslauthd -m /var/run/saslauthd -a pam
15530 ?        Ss     0:00 sendmail: accepting connections
15537 ?        Ss     0:00 sendmail: Queue runner@01:00:00 for /var/spool/c
15545 ?        Ss     0:00 /usr/sbin/httpd
15546 ?        S      0:00 /usr/sbin/httpd
15553 ?        Ss     0:00 crond
15999 ?        S      0:01 /usr/bin/perl /bin/monitor
17643 ?        SN     0:00 /usr/sbin/zabbix_agentd
17680 ?        SN     0:00 /usr/sbin/zabbix_agentd
17682 ?        SN     0:00 /usr/sbin/zabbix_agentd
17683 ?        SN     0:00 /usr/sbin/zabbix_agentd
17684 ?        SN     0:00 /usr/sbin/zabbix_agentd
17739 ?        SN     0:00 /usr/sbin/zabbix_agentd
29842 ?        Ss     0:00 vzctl: pts/0
29844 pts/0    Ss     0:00 -bash
32762 pts/0    R+     0:00 ps ax
[root@localhost /]#
```

Figure 24. Result of the `ps ax` command

6.4 *Results and Analysis*

The results reported come from data collected during the period of 19 months from April 2012 to October 2013. A total of 2914 honeypots were deployed.

6.4.1 Results

Figure 25 shows the number of honeypots deployed over time. On average 153.4 honeypots were deployed per month with a standard deviation of 67.7 honeypots per month.

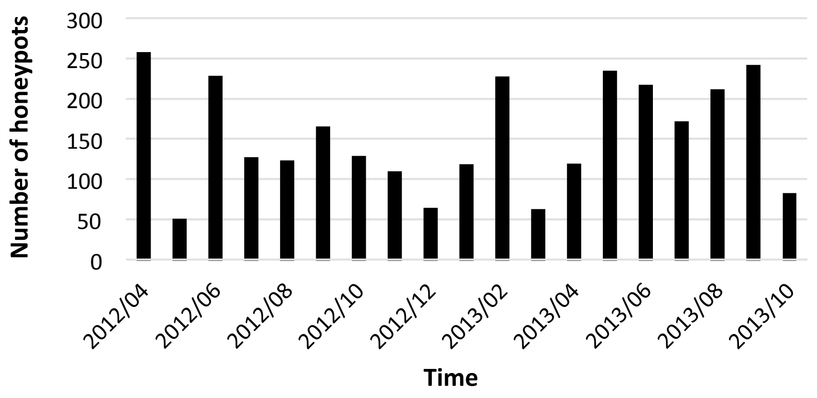


Figure 25. Number of Honeypots Deployed over Time

Figure 26 shows the distribution of the number of honeypots deployed over time for each of the four honeypot configuration types employed within the current design. A total of 710 honeypots of Type 0, 763 of Type 1, 694 of Type 2 and 747 of Type 3 were deployed. We observe that even though the overall number of honeypots deployed varies over time, the repartition among the four types remains consistent since we allocated the treatment conditions randomly.

For Type 0, the average number of honeypots that were deployed per month is 37.4 honeypots with a standard deviation of 15.7. The averages and standard deviations for Types 1, 2, and 3 are respectively 40.2/17.1, 36.5/19.9 and 39.3/18.6. These results support that randomization was successfully applied at the point of deployment. As such, any identified differences across treatment groups that follow can be attributed to the applied treatments, as all other characteristics pertaining to the attacker (both observed and unobserved) should be balanced in expectation across the four groups.

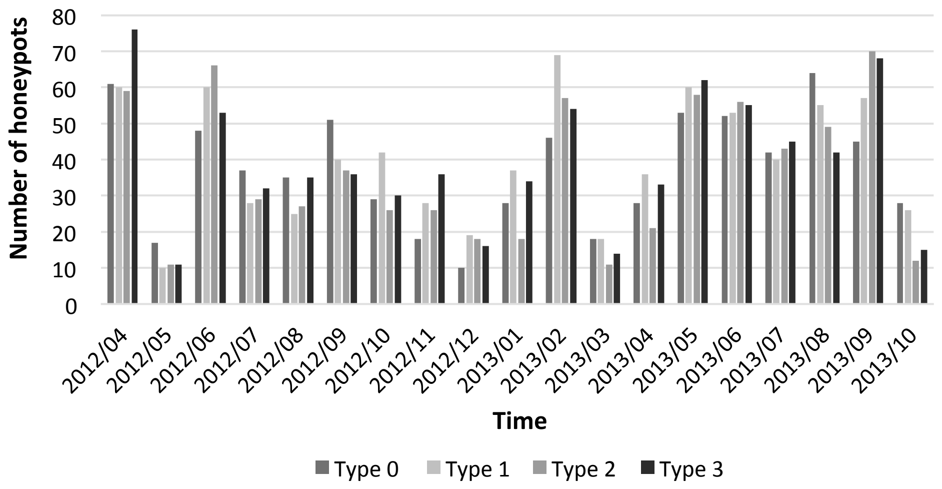


Figure 26. Number of Honeypots Deployed over Time by Honeypot Type

6.4.2 Combined Crimes Committed

This chapter focuses on the crimes committed following successful entry to the respective honeypot, i.e., attacks launched from the honeypots towards external computers. Specifically, an attacker commits a crime using the compromised honeypot when communicating in a particular way with some targets outside the organization network. A total of 611 crimes were committed, representing an average of 0.210 crimes per honeypot deployed.

Figure 27 shows the distribution of the number of crimes committed per honeypot. Among the honeypots that have committed crimes, we observe that most honeypots contained one crime (50 honeypots – 58.1%), two crimes (10 honeypots – 11.6%), or three crimes (7 honeypots – 8.1%). Thus, note the general positive skew in the data with most of the honeypots containing less than 4 crimes and 1 crime serving as the modal category. However, one honeypot was involved in 41 crimes, one in 59 crimes, one in 72 crimes and even one in 183 crimes. These 183 crimes resulting from a single honeypot were reconnaissance attacks against specific ports and targets; we have flagged this data point as an outlier. Another honeypot committed 72 crimes; these crimes consist of 52 DoS attacks (against 52 distinct targets) and 20 reconnaissance attacks. Since the majority of these crimes are targeted DoS attacks, we will not flag this data point as an outlier, and retain this observation for all further analyses. Henceforth, we will often present results with and without the flagged outlier of 183 reconnaissance attacks committed on one honeypot.

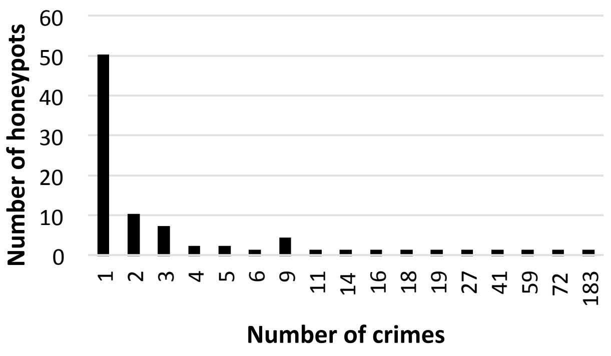


Figure 27. Number of Honeypots and Number of Crimes

Figure 28 and Figure 29 show the number of crimes (with and without the outlier) committed over the 19 months of data collection. When including the outlier, an average of 32.2 crimes/month with a standard deviation of 51.8 crimes/month were observed. When excluding the outlier, these values decrease to an average of 22.5 crimes/month and a standard deviation of 32.5 crimes/month. Even though the number of crimes varies over time, there is no obvious trend suggesting a difference of attacker behavior towards the honeypots based on the number of crimes. We want to verify that the population of attackers targeting the honeypots is not qualitatively changing over time. We are making sure that the differences we may observe are due to the applied treatment and not a change of attackers over time. The Augmented Dickey-Fuller trend test [DIC79, CHE95] is a commonly used test for that purpose [ELD01]. When applying the Augmented Dickey–Fuller trend test on the ratio of the number of monthly crimes divided by the number of monthly honeypots deployed, we find no statistically significant trend or unit root. We also analyze the number of crimes committed per honeypot configuration observed over time.

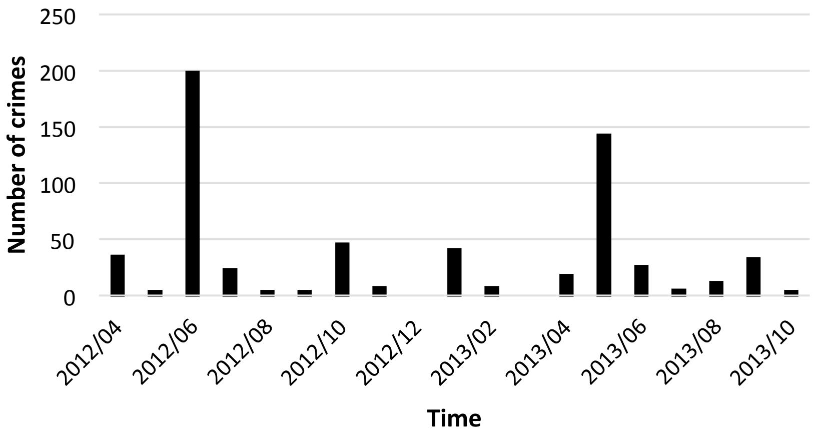


Figure 28. Number of Crimes over Time (With Outlier)

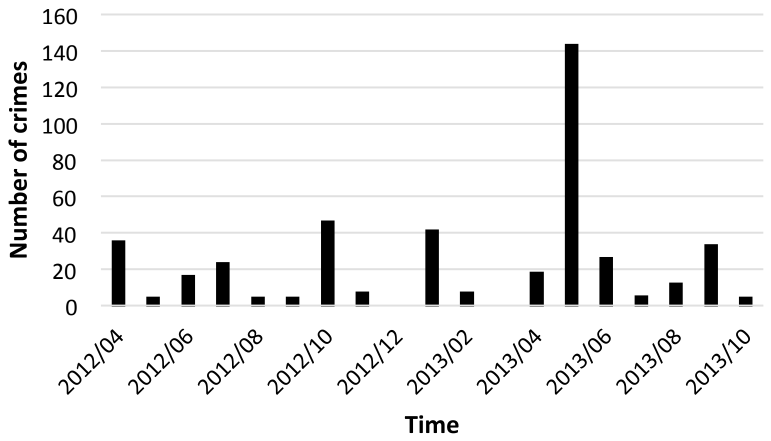


Figure 29. Number of Crimes over Time (Without Outlier)

Figure 30 and Figure 31 show, for each treatment condition, the number of crimes (with and without the outlier) committed over time. For Type 0, when including the outlier, an average of 14.5 crimes/month with a standard deviation of 42.6 crimes/month were observed. When excluding the outlier, these values decrease to an average of 4.84 crimes/month and a standard deviation of 8.29 crimes/month.

The averages and standard deviations for types 1, 2, and 3 are respectively 6.47/15.7, 4.53/6.04 and 6.68/16.8. For each honeypot type, there is no obvious trend suggesting a difference of attacker behavior towards the honeypots based on the number of crimes. When applying the Augmented Dickey–Fuller test on the ratio of the number of monthly crimes divided by the number of monthly honeypots deployed, for each honeypot configuration, we find no statistically significant trend or unit root. This analysis lends support that, based on the number of crimes committed (overall and for each honeypot configuration type), the data collection duration of 19 months did not lead to a difference of attacker behavior.

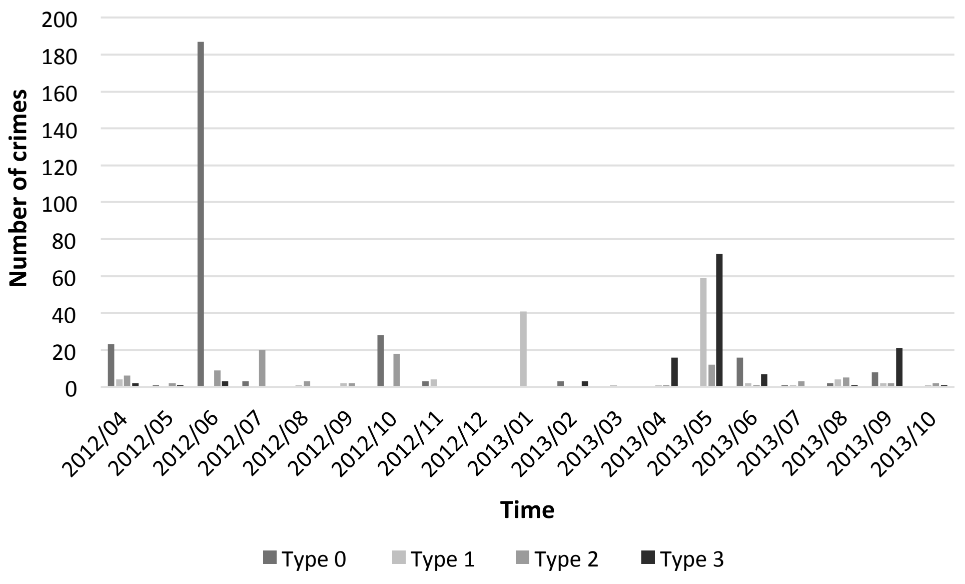


Figure 30. Number of Crimes per Honeypot Type over Time (With Outlier)

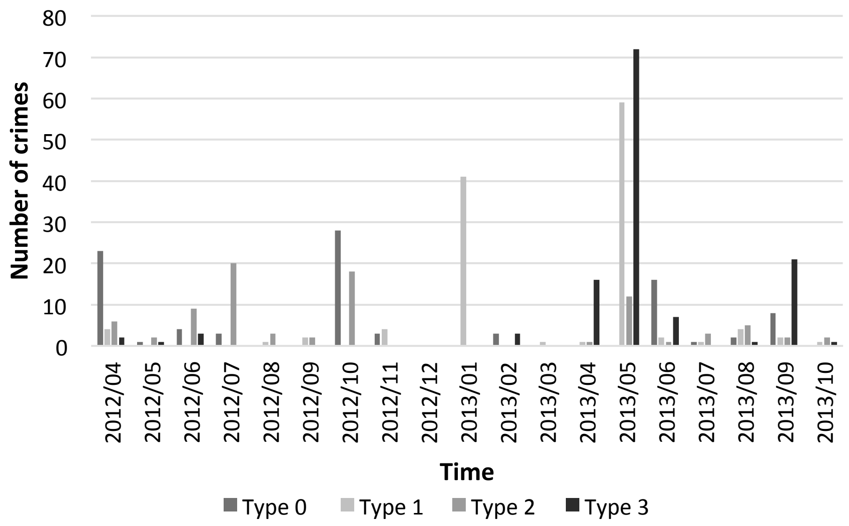


Figure 31. Number of Crimes per Honeypot Type over Time (Without Outlier)

Table 17 contains the number of deployed honeypots and the number of committed crimes for each honeypot configuration. We observe that the number of crimes varies between honeypot configurations. If we calculate the crime rate (i.e., number of crimes observed divided by the number of honeypots deployed and multiplied by 100) per 100 honeypots deployed for each honeypot configuration, we observe that the surveillance banner and surveillance tools seem to have a deterrent effect. This is confirmed with a chi-square test with a p-value of 0.00014.

Table 17. Crime Rates (With Outlier)

HP	# HPs deployed	# Crimes	Crime Rate
Type 0	710	275	38.7
Type 1	763	123	16.1
Type 2	694	86	12.4
Type 3	747	127	17.0
Total	2914	611	21.0

As previously seen, we have an outlier among the honeypots of Type 0 where one honeypot committed 183 crimes. When removing the outlier, we obtain the results shown in Table 18. Since the crime rates across conditions varies between 12.4 and 17.0, the surveillance banner and tools do not appear to have an impact on the overall crime rate. Indeed, the p-value rises to 0.79 with the exclusion of the outlier, which suggests the previous finding was driven exclusively by the presence of an outlier in the control condition.

For the four honeypot configurations, we applied a Kruskal-Wallis H test on the number of committed crimes per honeypot. When including the outlier, we obtain a p-value of 0.932. Without the outlier, the p-value becomes 0.98. In both cases, we fail to reject the null hypothesis that the honeypot configuration does not have an impact on the number of crimes committed on a honeypot.

Table 18. Crime Rates (Without Outlier)

HP	# HPs deployed	# Crimes	Crime Rate
Type 0	709	92	13.0
Type 1	763	123	16.1
Type 2	694	86	12.4
Type 3	747	127	17.0
Total	2913	428	14.7

We also analyzed the time of the first crime committed on each honeypot. The Kruskal-Wallis H test led to a p-value of 0.717. Thus, we fail to reject the null hypothesis that the honeypot configuration does not have an impact on the timing of the first crime committed on each honeypot.

Table 19 shows for each honeypot configuration the number of honeypots deployed as well as the number of honeypots that were used for committing at least one crime (i.e., use of this honeypot to launch an attack towards an external target). The type of honeypot configuration does not seem to have a significant impact since the ratio varies between 2.2% and 3.7%. Indeed the p-value of the chi-square test is 0.90. When considering all the honeypots deployed, 2.95% were used to commit a crime. This number is very low since the honeypot does not contain any true ‘honey’ (e.g., credit card number, social security number copies of passport, bank documents) and its only value should theoretically lay in using it to launch attacks against other targets. Observing that this happens in only about 3% of the cases is an intrinsically noteworthy finding. Additionally, one should note that this percentage is consistent across the different honeypot types.

Table 19. Crime Ratio

HP	# HPs deployed	HPs with at least one crime	Ratio
Type 0	710	25	0.0352
Type 1	763	17	0.0223
Type 2	694	26	0.0375
Type 3	747	18	0.0241
Total	2914	86	0.0295

To better understand such low crime rates, we indicate in Table 20 the number of honeypots that were involved in malicious activity (i.e., were used by the attacker) based on the observation of any keystroke activity. The ratios in Table 20 indicate, for each honeypot configuration, the percentages of honeypots containing some malicious activity. We observe that the percentages vary between 63% and 64.2% (overall 63.8%). This narrow range in the proportion of honeypots containing at least some malicious activity across the four types is in line with a priori expectations given the appropriate implementation of randomization. These numbers show that about one third of the honeypots that were compromised (where credentials were obtained) were not even used in any development of an attack. Attackers did not use them without knowing their potential value. This might indicate that attackers have a large pool of compromised computers and do not use all the ones they broke into.

Table 20. Malicious Activity Ratio

HP	# HPs deployed	HPs with malicious activity	Ratio
Type 0	710	456	0.642
Type 1	763	481	0.630
Type 2	694	448	0.645
Type 3	747	474	0.634
Total	***2914***	***1859***	***0.638***

Two other important metrics are introduced for the present analysis pertaining explicitly to those honeypots that contain some malicious activity: the probability that at least one crime has been committed (i.e., X>=1) and the rate at which these honeypots engaged in crime. For each of the honeypot configurations, these numbers are provided in Table 21.

We observe that the probability that at least one crime has been committed varies between 3.5% and 5.8% across the honeypots configurations. As depicted in Table 21, Type 1 and Type 3 have slightly lower percentages than Type 0 and Type 2 with Type 2 having the highest percentage of the four conditions with 5.80%. So the honeypot configuration type does not seem to have an impact on the probability that at least one crime has been committed, as the chi-square test produces a p-value of 0.83. The overall probability of at least one crime on these honeypots is 4.6%, which is an otherwise surprising observation, as one would expect the probability of committed crimes from the honeypots to be higher given the aforementioned lack of honey on these systems.

Table 21. Committed Crimes

HP	Prob. at Least One Crime Committed	Rate of Crimes Committed
Type 0	25/456 (5.48%)	275/456 (0.603) (with outlier) 92/456 (0.202) (without outlier)
Type 1	17/481 (3.53%)	123/481 (0.256)
Type 2	26/448 (5.80%)	86/448 (0.192)
Type 3	18/474 (3.80%)	127/474 (0.268).
Total	***86/1859 (4.63%)***	***611/1859 (0.329) (with outlier)*** ***428/1859 (0.230) (without outlier)***

When considering the rate of crimes committed per honeypot with malicious activity, we observe that the rate is 0.603 crimes committed per honeypot for Type 0 while including the outlier. This rate falls to between 0.192 and 0.268 crimes per honeypot in the presence of a surveillance banner or surveillance tools. As such, one might conclude that surveillance mechanisms evoke a deterrent effect within this sample, and this conclusion is affirmed by a chi-square test with the p-value approaching zero. However, upon removal of the outlier, Type 1 and Type 3 honeypots have higher rates of crimes committed per honeypot than the control group. The overall rate becomes 0.23 crimes committed per honeypot with malicious activity, and the chi-square test is no longer statistically significant with a p-value of 0.59, and thus does not support the conclusion that surveillance evokes a deterrent effect on the rate of crimes committed.

As discussed in the previous section, the surveillance mechanisms under assessment consist of a banner retaining surveillance content and two surveillance tools. The banner appears upon entry each time the hacker accesses the system. However, we cannot prove that the attacker saw it, read it, nor understood it (e.g., the attacker might not be familiar in English). For the tools, we also have no guarantee attackers have identified them. When analyzing the session content and looking for the “ps” command that indicates which processes are running, we found the following honeypots had used “ps” at least once: Type 0, 2/25 (8.0%), Type 1, 2/17 (11.8%), Type 2, 9/26 (34.6%) and Type 3, 4/18 (22.2%), leading overall to 17/86 (19.8%). So only for 20% of the honeypots that have committed a crime “ps” was used to check which processes were running on the system. Moreover, this does not prove that attackers found the presence of the surveillance tools but only that attackers were interested in the running processes. These findings have the potential to downward bias any potential statistically significant effects that might otherwise be observed with regard to the effect of surveillance means on crime outcomes. E.g., the surveillance processes only have the potential to elicit an effect on attacker behavior amongst this 20% subsample that actually brought up the list of processes.

We define a coordinated crime as any crime during which several other hosts are contributing to the same criminal event. Even though this cannot be determined with certainty, it is possible to identify the honeypots generating Internet Relay Chat (IRC) traffic. IRC is known to be used by Botnet to command and control several compromised hosts [ZHU07]. We postulate that any crime for which we observe IRC traffic right before the start of the crime or during the crime provides evidence of a coordinated crime.

Table 22 shows the total number of honeypots with at least one crime as well as the number of honeypots containing some IRC traffic for each honeypot type. We observe that overall, 80% of the crimes were coordinated. When focusing on specific honeypot types, the percentage of coordinated crimes varies between 67% and 88%. However, this disparity is not found to be statistically significant across conditions as the chi-square test returns a p-value of 0.36. The preponderance of coordinated crimes is stark given the lack of sophistication inherent to the observed crimes that would otherwise not warrant this observed level of coordination.

Table 22. Coordinated Crimes

HP	# HPs with at least one crime	Coordinated	Ratio
Type 0	25	20	80.00
Type 1	17	14	82.35
Type 2	26	23	88.46
Type 3	18	12	66.67
Total	***86***	***69***	***80.23***

6.4.3 Classification of Observed Crimes

During data collection, we observed four of the five types of crimes described in Section 5.4.2 of Chapter 5:

- ***Reconnaissance activities:*** *389 instances over 79 honeypots*
- ***Denial of Service attacks:*** *180 instances over 10 honeypots***.** In addition to the two different types of Denial of Services attacks observed in a previous study we noticed the unusual use of transport protocols such as Combat Radio User Datagram Protocol characterizes: One honeypot used 240 different transport layer protocols.
- ***Brute force attacks:*** *40 instances over 12 honeypots*
- ***Phishing attacks:*** *2 instances over 2 honeypots*

Since phishing attacks were only observed twice during the data collection period, we will remove them and will focus on the most frequently committed crimes: reconnaissance, DoS and brute force attacks.

6.4.3.1 Reconnaissance Attacks

We first assess whether we observe a trend for the number of reconnaissance attacks over the 19 months of data collection. When applying the Augmented Dickey–Fuller trend test on the ratio of the number of monthly reconnaissance attacks divided by the number of monthly honeypots deployed we find no statistically significant trend or unit root in the data. We also applied this test on the same ratio for the data collected for each honeypot type and also found no statistically significant trend or unit root in the data. Table 23 and Table 24 contain the number of deployed honeypots and the number of reconnaissance attacks (with and without the outlier) for each honeypot type. We calculate the crime rate for reconnaissance attacks (i.e., number of crimes observed divided by the number of honeypots deployed and multiplied by 100) per 100 honeypots deployed for each honeypot type. We observe a stark effect for the surveillance mechanisms on the crime rate while including the outlier, but this effect disappears when excluding the outlier. The p-values from chi-square tests confirm these observations with 1.56E-11 (with outlier) and 0.30 (without outlier). Thus, when excluding the outlier, we fail to reject the null hypothesis that the honeypot type does not have an impact on the rate of reconnaissance attacks launched from the honeypots.

Table 23. Reconnaissance Attacks Rates (With Outlier)

HP	# HPs deployed	# Reconnaissance Attacks	Reconnaissance Attack Rate
Type 0	710	259	36.5
Type 1	763	27	3.54
Type 2	694	50	7.20
Type 3	747	53	7.095
Total	***2914***	***389***	***13.3***

Table 24. Reconnaissance Attacks Rates (Without Outlier)

HP	# HPs deployed	# Reconnaissance Attacks	Reconnaissance Attack Rate
Type 0	710	76	10.7
Type 1	763	27	3.54
Type 2	694	50	7.20
Type 3	747	53	7.095
Total	***2914***	***206***	***7.07***

For the four honeypot types, we applied a Kruskal-Wallis H test on the number of reconnaissance attacks per honeypot and obtained a p-value of 0.542 (with outlier) and 0.704 (without outlier). Thus, we fail to reject the null hypothesis that the honeypot type does not have an impact on the rate of reconnaissance attacks launched from each honeypot.

We also analyzed the time of the first reconnaissance attack launched on each honeypot. The Kruskal-Wallis H test led to a p-value of 0.696. Thus, we fail to reject the null hypothesis that the honeypot type does not have an impact on the timing of the first reconnaissance attack launched from each honeypot.

Finally, we analyzed the time associated with each reconnaissance attack launched from each honeypot. For the sake of brevity, these data are not displayed here, but are available upon request from the author. The Kruskal-Wallis H test led to a p-value of

0.102 (with outlier) and 0.08 (without outlier). Thus, we fail to reject the null hypothesis that the honeypot type does not have an impact on the timing of the reconnaissance attacks launched from each honeypot.

In sum, we fail to find a statistically significant effect of surveillance mechanisms on relevant metrics associated with reconnaissance activity launched from the honeypots.

6.4.3.2 *Denial of Service Attacks*

We first assess whether we observe a trend for the number of DoS attacks over the 19 months of data collection. When applying the Augmented Dickey–Fuller trend test on the ratio of the number of monthly DoS attacks divided by the number of monthly honeypots deployed we find no statistically significant trend or unit root. We also applied the test on the same ratio for each honeypot configuration and also found no statistically significant trend or unit root.

Table 25 contains the number of deployed honeypots and the number of DoS attacks for each honeypot type as well as the crime rate for DoS attacks. We observe a stark effect for the surveillance banner and surveillance tools on the crime rate pertaining to DoS attacks. This is confirmed with a chi-square test with a p-value of 0.0047. Thus, we reject the null hypothesis that the honeypot configuration does not have an impact on the rate of DoS attacks launched from each honeypot. However, we should note that this effect is in a direction contrary to a priori expectations as DoS attacks are only observed for those honeypots containing surveillance mechanisms with no DoS attacks launched from honeypots in the control group.

Table 25. DoS Attacks Rates

HP	# HPs deployed	# DoS Attacks	DoS Attack Rate
Type 0	710	0	0.0
Type 1	763	95	12.45
Type 2	694	33	4.755
Type 3	747	52	6.96
Total	***2914***	***180***	***6.18***

For the four honeypot configurations, we applied an Kruskal-Wallis H test on the number of DoS attacks per honeypot and obtained a p-value of 0.108. Thus, we fail to reject the null hypothesis that the honeypot type does not have an impact on the number of the DoS attacks launched from each honeypot.

We also analyzed the time of the first DoS attack launched on each honeypot. The Kruskal-Wallis H test led to a p-value of 0.652. Thus, we fail to reject the null hypothesis that the honeypot type does not have an impact on the timing of the first DoS attack launched from each honeypot.

Finally, we analyzed the time associated with each DoS attack launched from each honeypot. The Kruskal-Wallis H test led to a p-value of 2.10E-07. Thus, we reject the null hypothesis that the honeypot type does not have an impact on the timing of the DoS attacks launched from each honeypot.

In sum, the assignment to the four honeypot types included in this analysis appears to have an effect on the rate at which attackers engage in DoS attacks. However, this is an augmentative effect wherein surveillance mechanisms increase, and in fact produce, the DoS attack rates as compared to the control condition. This runs counter to deterrence-based expectations and should be explored further in future analyses.

6.4.3.3 *Brute Force Attacks*

We first assess whether we observe a trend for the number of brute force attacks over the 19 months of data collection. When applying the Augmented Dickey–Fuller trend test on the ratio of the number of monthly brute force attacks divided by the number of monthly honeypots deployed we find no statistically significant trend or unit root. We also applied the test on the same ratio for each honeypot configuration and also found no statistically significant trend or unit root.

Table 26 contains the number of deployed honeypots and the number of brute force attacks for each honeypot type. If we calculate the crime rate (i.e., number of crimes observed divided by the number of honeypots deployed and multiplied by 100) per 100 honeypots deployed for each honeypot type, we do not observe a clear effect for the surveillance banner and surveillance tools on the respective crime rate. This is confirmed with a chi-square test with a p-value of 0.243. Thus, we fail to reject the null hypothesis that the honeypot configuration does not have an effect on the rate of brute force attacks launched from each honeypot.

Table 26. Brute Force Attacks Rates

HP	# HPs deployed	# Brute Force Attacks	Brute Force Attack Rate
Type 0	710	15	2.11
Type 1	763	1	0.131
Type 2	694	2	0.288
Type 3	747	22	2.94
Total	***2914***	***40***	***1.37***

For the four honeypot types, we applied a Kruskal-Wallis H test on the number of brute force attacks per honeypot and obtained a p-value of 0.238. As such, we fail to reject the null hypothesis that the honeypot type does not have an impact on the rate of brute force attacks launched from a honeypot.

We also analyzed the time of the first brute force attack launched on each honeypot. The Kruskal-Wallis H test produced a p-value of 0.33. Thus, we fail to reject the null hypothesis that the honeypot type does not have an impact on the timing of the first brute force attack launched from each honeypot.

Finally, we analyzed the time associated with each brute force attack launched from each honeypot. The Kruskal-Wallis H test produced a p-value of 0.024. Thus, we reject the null hypothesis that the honeypot type does not have an impact on the timing of the brute force attacks launched from each honeypot.

In sum, we generally fail to find a statistically significant effect of a surveillance banner and surveillance processes on brute force attacks according to the highlighted metrics. The lone exception to this is with regard to the timing of the first brute force attack, which warrants further study.

6.5 Discussion

From a statistical point of view, most of the results presented are tantamount to null effects. However, the simple fact that not all of the above tests resulted in null effects suggest that there may be a mechanism at play connecting the presentation and application of surveillance content with attacker behavior. This opens the door to further research to better understand this procedure and whether deterrence is applicable given the finding regarding DoS attack behavior produced from those honeypots retaining surveillance content.

In another study focusing on attack sessions and keystrokes, we observed that the presence of a surveillance banner reduced the probability of commands being typed in the first attack session. In addition, the presence of keystrokes in the following attack sessions was conditioned by the presence of the surveillance banner and keystrokes in the first session. This study has been published in [MAI14].

The framework we developed could handle up to 300 deployed honeypots at the same time. We often did not reach the limit of these 300 honeypots. The goal of the long data collection period was to ensure a high number of honeypot deployments and ideally a high number of observed crimes. We show in this chapter that only about 3% of the honeypots committed at least one crime; Even though 2,914 honeypots were deployed, only 86 of them were involved in committing at least one crime. Many were involved in committing several crimes leading to a total of 611 crimes committed during the data collection period. When applying statistical tests, we observed that most led to a null effect. In some case very few crimes had been committed (e.g., 1 brute force attack for Type 1 honeypot configuration). This raises an issue with regard to what constitutes an adequate sample size for analyzing such rare events. Even a data collection period of over 19 months with a framework handling a potentially larger farm of honeypots might not be sufficient from a statistical point of view. The impetus behind the development of empirical studies is to obtain a better understanding of the attack threat. As such, it may be found permissible to derive some conclusions and substantive interpretations in spite of the aforementioned limitations to the present study.

6.6 Limitations

One limitation of the study is that data were collected on one specific network. We do not claim that these results are generalizable to other networks, as the same study on a different network might lead to different results and conclusions. This is a common limitation for field experiments wherein the external validity is often limited, but is countered by the high internal validity inherent to randomized experiments that enable the identification of potentially causal effects.

Another limitation is that we do not claim these results will remain true over time. This study was conducted over a specific period of time, and as such, may retain limited retrospective and prospective application. The attack behavior can rapidly change, which necessitates that our findings should be revisited by future research.

The data collection period is rather long: 19 months. One concern is that the crime behavior might have changed during that time. This is why we ran some trend tests for the overall number of crimes (for all honeypots and for each of the honeypot configuration types) as well as for the most frequently observed crimes (reconnaissance, DoS, and brute force attacks) (for all honeypots and for each of the honeypot configuration types). In each case, we did not find any statistically significant trend or unit root in the data. These observations help mitigate the potential bias inherent to this limitation related to the long data collection period.

The present study does not provide any insight on the attacker. We concede that we cannot guarantee that each miscreant using the framework is human. Conversely, we cannot guarantee that each miscreant using the framework is an automated bot. The keystroke dataset contains the delays in milliseconds between each keystroke. Large and irregular intervals would lend themselves toward a human typing the commands. Short and regular intervals would suggest that the session is controlled by an automated script. We find evidence of both types, but note that this does not definitively prove a session was conducted by a human or a bot. We use this as support that there are likely a nontrivial number of bots and a nontrivial number of humans within our dataset. This is relevant within this context for providing the aggregate effect of such a policy implementation on a real-world computer network. This still increases the probability of a type II error within the present analysis due to this lack of differentiation between computer and human users, but that does not diminish the relevance of these analyses due to this serving as an effective test of a social-science driven policy.

The configurations applied to the honeypots also present some limitations. As previously mentioned, we have no guarantee that the attackers read the banner or understood it since the message displayed is in English. Attackers from non-English speaking countries may not understand the message announcing the surveillance. Moreover, we also do not know whether attackers checked for the monitoring tools even when they listed the processes running on the honeypot.

In order to prevent attackers from detecting the nature of the compromised hosts, the honeypots behavior should be close to a common UMD operated system. The means to convey surveillance, i.e. the banner announcing the surveillance processes, and the monitoring tools were selected based on several constraints. For example, a banner with counter attach threats is against the UMD policy. A message announcing the monitoring tools in the middle of a session is not a usual behavior for a computer system.

6.7 Conclusions

This chapter focuses on one aspect of cybercrime deterrence: the announcement and existence of surveillance mechanisms. More specifically, we investigate whether computer focused crimes are impacted by a surveillance warning banner or surveillance tools.

A farm of four honeypots was configured as follows: Type 0) no surveillance banner nor tools, Type 1) a surveillance banner but no surveillance tools, Type 2) no surveillance banner but surveillance tools, and Type 3) a surveillance banner and surveillance tools. Following a brute force attack on Secure Shell, attackers were randomly assigned to one of these treatment conditions and were granted access to the target system for 30 days. Computer focused crimes were identified through the network flows. The impact of a surveillance banner and/or surveillance tools was analyzed based on the number of crimes, time of the first crime, time distribution for all crimes and the crime rates for the most frequently observed crimes.

We observed that none of the honeypot configurations had a statistically significant impact upon these metrics when considering all crimes, but that some impact was measured for some of the most frequently observed crimes.

Chapter 7 – Effects of a Banner on the Commands Typed by Attackers: Differences across Countries

7.1 Introduction

Chapter 6 discussed the effect of a surveillance banner and surveillance mechanisms on the attacks launched by a target computer following a successful compromise by attackers. We observed that surveillance had no impact on the number of crimes, time of the first crime, time distribution for all crimes and the crime rates for the most frequently observed crimes when considering all crimes but that some impact was measured for some of the most frequently observed crimes.

7.2 Research Questions and Hypotheses

In this chapter we aim to empirically study the following research questions:

c) *Is there a variation in the probability that an attacker would enter commands depending on the country of origin from which an attack is launched?*

d) *Do attackers from different countries vary in their use of system activity commands in the presence of a surveillance banner?*

For this study we analyzed 1) the data from the session and deployment tables to identify the country of origin of all the sessions per honeypots, and 2) the keystrokes processed into lists of commands to study the attackers' actions. We provided details on these data in Section 3.3.2.4 of Chapter 3.

Theoretical and empirical research from a variety of social science disciplines ranging from psychology to criminology to business ethics continues to show that the effects of sanctions are not constant across individuals in a given population [ARC09], [LOU12], [THO3]. A review of the current research shows that in the physical world, criminals often display different responses to deterrence mechanisms.

In this chapter, we investigate whether a surveillance banner alters the behavior of attackers based on their country of origin (based on the observed IP address). Using several metrics, such as commands typed by attackers, to measure attackers' behavior we explore whether attackers originating from the United States, China, Romania, Republic of Korea, and Germany display significant differences in specific session characteristics during an attack based on whether or not they were exposed to a surveillance banner.

We specifically focus on the possibility that attackers from different countries may respond differently to a surveillance banner as factors such as cultural differences existing across nations, as well as differential assessments regarding the likelihood of punishment may result in varying responses to a particular sanction threat.

For example, in one country, the presence of a surveillance banner may serve as a valid signal of a threat, and accordingly, attackers originating from that country would internalize a heightened risk of apprehension and punishment, reducing their adverse behavior in response. However, in another country, a banner may be perceived in an opposing manner. For instance, attackers originating in countries geographically separated from the target may perceive a lower likelihood of apprehension and therefore, may not be deterred by a threat.

Moreover, in certain countries, rather than eliciting a deterrent effect, a banner may instead generate feelings of defiance and a willingness to oppose the threat posed by an authority. Consequently, the banner may increase the propensity of an attacker to engage in adverse behavior [SHE93].

Despites the anonymity offered by the Internet and because of their human nature, attackers exhibit a rational decision-making process [PNG09]. They will attempt to maximize rewards while minimizing the risks of being detected. Because of the proximity of the attackers originating from the United States, we hypothesize that *U.S. attackers, more easily identifiable and prosecutable, would be deterred by surveillance.* More specifically, *U.S. attackers understanding a surveillance announcement banner would look for effective surveillance cues including monitoring tools*.

7.3 Experimental Design

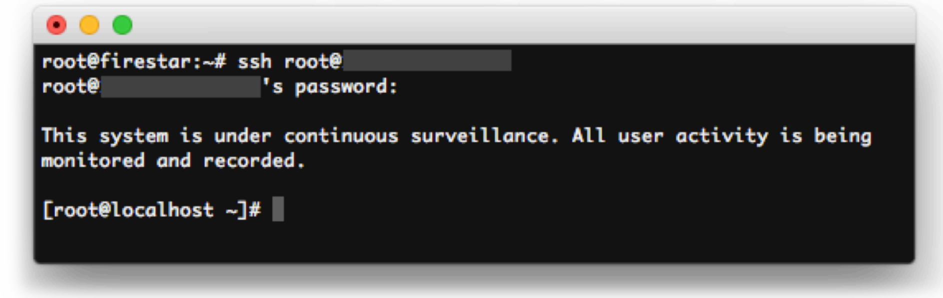
Attackers, who have been identified according to their IP addresses, are randomly attributed one of the configuration types listed in Table 27. The configuration randomly assigned to the honeypot container involves a two (banner vs no banner) x two (processes vs no processes) design:

- The display of a banner after a SSH login informing the user that the system is under surveillance (Figure 32),
- The presence of surveillance tool processes (Figure 33)

Figure 33 shows two different surveillance processes. One is called *zabbix_agentd*, which is the agent of Zabbix an open source monitoring solution. The other is a script named "*monitor*". "*monitor*" updates and saves a file every minute containing the disk usage, the system uptime, the available memory, the users' logins and the running processes. The attacker has access to these files.

Table 27. Honeypot Configuration Types

Honeypot Type	Surveillance Banner	Surveillance Processes
0	No	No
1	Yes	No
2	No	Yes
3	Yes	Yes

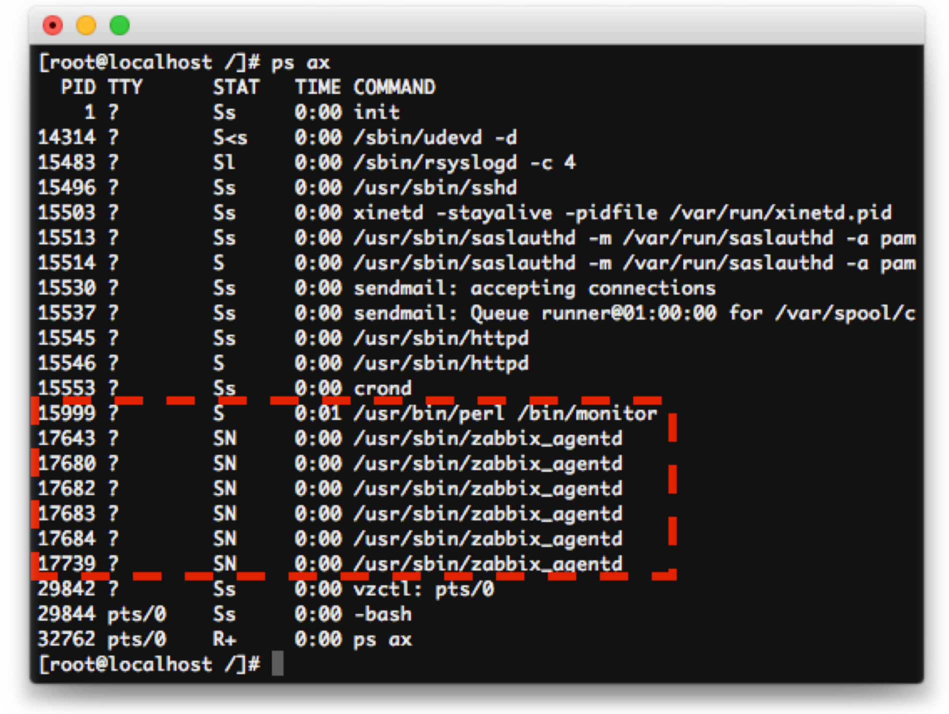


```
root@firestar:~# ssh root@[redacted]
root@[redacted]'s password:

This system is under continuous surveillance. All user activity is being
monitored and recorded.

[root@localhost ~]#
```

Figure 32. Surveillance Banner Displayed



```
[root@localhost /]# ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 init
14314 ?        S<s    0:00 /sbin/udevd -d
15483 ?        Sl     0:00 /sbin/rsyslogd -c 4
15496 ?        Ss     0:00 /usr/sbin/sshd
15503 ?        Ss     0:00 xinetd -stayalive -pidfile /var/run/xinetd.pid
15513 ?        Ss     0:00 /usr/sbin/saslauthd -m /var/run/saslauthd -a pam
15514 ?        S      0:00 /usr/sbin/saslauthd -m /var/run/saslauthd -a pam
15530 ?        Ss     0:00 sendmail: accepting connections
15537 ?        Ss     0:00 sendmail: Queue runner@01:00:00 for /var/spool/c
15545 ?        Ss     0:00 /usr/sbin/httpd
15546 ?        S      0:00 /usr/sbin/httpd
15553 ?        Ss     0:00 crond
15999 ?        S      0:01 /usr/bin/perl /bin/monitor
17643 ?        SN     0:00 /usr/sbin/zabbix_agentd
17680 ?        SN     0:00 /usr/sbin/zabbix_agentd
17682 ?        SN     0:00 /usr/sbin/zabbix_agentd
17683 ?        SN     0:00 /usr/sbin/zabbix_agentd
17684 ?        SN     0:00 /usr/sbin/zabbix_agentd
17739 ?        SN     0:00 /usr/sbin/zabbix_agentd
29842 ?        Ss     0:00 vzctl: pts/0
29844 pts/0    Ss     0:00 -bash
32762 pts/0    R+     0:00 ps ax
[root@localhost /]#
```

Figure 33. Result of the `ps ax` command

7.4 Results and Analysis

7.4.1 Results

This analysis focuses on data collected over a 31-month period from March 31, 2012 until October 29, 2014. During this period of analysis, 5,231 deployments occurred along with a total of 49,149 sessions from 103 different countries. The database used the countries and country codes from the International Organization for Standardization (ISO) 3166. For some IP addresses, the mapping to a country was not possible due to geo-localization restrictions, resulting in a blank country of origin for that session. These sessions are categorized as *Unknown*. The geo-localization database also flagged the known anonymous public proxy servers, these sessions are also categorized as *Unknown*.

As mentioned in [STU12], the IP address used during the brute-force phase before the honeypot deployment often originates from different countries compared to the ones used during attack sessions. Our results also show such differences regarding the IP address origins. In addition, the brute-force, also known as dictionary, attack is often automated and thus no attacker can actually read the surveillance banner at this stage. Consequently, the surveillance banner can only be seen starting with the first session when the attacker logs onto the honeypot after its deployment. We focused on the sessions' origin countries, not the deployment origin countries.

The number of sessions per country ranges from 19,653 (United States) to one (Azerbaijan, Ghana, Latvia, New Caledonia, Nicaragua, Papua New Guinea, Seychelles, United Arab Emirates, and American Samoa). Table 28 shows the five countries with the highest originating session counts: the United States, China, Romania, the Republic of Korea and Germany. The next highest number of session corresponds to the category *Unknown*. The other countries account for fewer sessions. For the remainder of the chapter, we will focus on the five countries with the highest number of sessions.

Table 28. Number of Sessions per Country (Top 5)

Country	Number of Sessions
United States	19,653
China	13,420
Romania	3,845
Republic of Korea	1,395
Germany	1,267

7.4.2 Analysis

We focused our analysis on the impact of the surveillance banner, as very few attackers issued the correct commands that would display the surveillance processes and reveal the existence of surveillance tools. For the following study, we merged honeypots types 0 and 2 (No Banner), and honeypot types 1 and 3 (Surveillance Banner).

7.4.2.1 Total Number of Sessions

We first started to look at the number of sessions per country and per honeypot configuration, i.e., with surveillance banner or without a surveillance banner. We believed that the surveillance banner deters the attackers and that it impacts the number of sessions. We expected to see fewer sessions on the honeypots displaying the surveillance banner than the ones not displaying any kind of text. It translates into the attacker not coming back on the honeypot after seeing the surveillance banner.

Table 29 shows the total number of sessions for the honeypots with and without a banner for the five considered countries. We want to assess whether deterrence has the same effect for these five countries. Therfore, we applied a χ-square test to assess whether the impact of the banner on the number of sessions depends on the country. With a p-value of 1.E-99, the χ-square test shows that the number sessions on the honeypots displaying a banner or not displaying a banner is independent of the country.

From Table 29, we see that the deterrence effect would lead to a smaller number of sessions when a banner is being displayed. This is the case for the United States and the Republic of Korea. However, the opposite is observed for China, Romania and Germany.

Table 29. All Sessions

Country	Banner	No Banner
United States (19,653)	9,395	10,258
China (13,420)	7,204	6,216
Romania (3,845)	1,966	1,879
Republic of Korea (1,395)	442	953
Germany (1,267)	863	404

7.4.2.2 Number of Sessions with Keystrokes

Depending on the presence of a banner, the number of sessions is not independent from where the attack was launched. In addition, the banner may impact the actions done by the attackers during these attack sessions. Our dataset contains the keystrokes typed by the attacker during the attack session. Our assumption is that the banner impacts the presence of keystrokes during an attack session. We believe that if deterred by the surveillance banner, the attacker would not type anything and leave the system to avoid detection.

We counted the number of sessions with keystrokes, i.e. attack sessions where the attacker issued commands to his or her honeypot after deployment. Table 30 shows for each of the top five countries the number of sessions with keystrokes split between honeypots displaying or not displaying a banner.

Table 30. All Session with Keystrokes

Country (sessions with keystrokes)	**Banner** (sessions with keystrokes)	**No Banner** (sessions with keystrokes)
United States (494)	236	258
China (730)	349	381
Romania (1713)	871	842
Republic of Korea (34)	16	18
Germany (288)	146	142

We applied a χ-square test to assess whether the impact of the banner on the number of sessions with keystrokes depends on the country. With a p-value of 0.58, we cannot reject the hypothesis that number of sessions with keystrokes is independent of the country.

From Table 30, we see that the deterrence effect would lead to a higher number of keystroke sessions for honeypots without a banner. This is the case for the United States, China and the Republic of Korea but not for Romania and Germany. Thus, these results show that the surveillance banner does not have a systematic deterrence effect on whether or not the attackers type during the attack sessions.

The attacker may not be deterred anymore when he or she realizes the honeypot is still compromised, i.e., the host is still online and the compromised user account has still the same password. It may indicate that the system is not under surveillance and thus the attacker can actually perform malicious activities on the compromised honeypot and ignore the banner. For this reason, we also looked at the first attack sessions only.

Table 31 shows for each of the five countries the number of first sessions with keystroke with banner and without banner. Since the observations are lower than 5 for the Republic of Korea, we need to exclude these results when applying a χ-square test. Therefore, we applied a χ-square test to assess whether the impact of the banner on the number of first sessions with keystrokes depends on the country for the United States, China, Romania, and Germany. With a p-value of 0.68, the χ-square test, we cannot reject the hypothesis that number of first sessions with keystrokes is independent of the country.

From Table 31, we see that the deterrence effect would lead to a higher number of first sessions with keystrokes for honeypots without a banner. This is the case for the United States and Romania but not for China, the Republic of Korea and Germany. Thus, these results show that the surveillance banner does not have a systematic deterrence effect on whether or not the attackers type during the first attack sessions. Even though the differences are small, these results seem to point towards a difference in behavior between attacks from the United States and China.

Table 31. First Sessions with Keystrokes

Country (1st session w/keystrokes)	**Banner** (1st session w/keystrokes)	**No Banner** (1st session w/keystrokes)
United States (86)	40	46
China (50)	27	23
Romania (192)	92	100
Republic of Korea (4)	3	1
Germany (30)	17	13

7.4.2.3 System Activity Commands

An attacker who receives a banner announcing the presence of a surveillance system may decide to look if it is indeed true and maybe disable it. He or she might want to check if someone is logged in to watch him or her or if the stolen account has ever been used. We concentrated the analysis of the keystrokes on the following commands:

- *ps:* This command shows the processes currently running on the system.
- *top:* This command also shows the processor usage in real time per processes.
- *who:* This command displays the current logged in users on the system.
- *ls:* This command lists the files present in a directory.

If someone else is using the system, the *who* command may return one or more other users logged into the system. Also, someone actively using a system leaves traces. A home directory with no files clearly indicates that the user is not really active and barely uses the system.

Table 32 presents, for all five countries, the number of sessions with system activity check commands displaying a surveillance banner and the one without banner.

Table 32. All Sessions with System Activity Commands

Country (sessions with commands)	**Banner** (sessions with commands)	**No Banner** (sessions with commands)
United States (211)	100	111
China (103)	62	41
Romania (859)	420	439
Republic of Korea (19)	6	13
Germany (212)	105	107

We applied a χ-square test to assess whether the impact of the banner on the number of sessions with system activity commands depends on the country. With a p-value of 0.10, the χ-square test, we cannot reject the hypothesis that number of sessions with system activity commands is independent of the country.

The results of the banner and no banner configurations comparison for each country show a positive difference (i.e., higher number of sessions for honeypots having a banner compared to the ones without one) between the configuration with banner and the one without banner for the sessions from China only. According to the results it seems that attackers from China check for the system activity when a banner is presented upon login. This result is interesting since the cyber security community usually assumes that attackers who launched an attack outside the United States would not be influenced by any deterrence approach.

For the same reasons as the keystrokes analysis, we then focused on the first attack session. Table 33 contains the number of the first sessions with system commands displaying a banner and the ones not presenting a surveillance banner. Since the observations are lower than 5 for the Republic of Korea, we exclude these results when applying a χ-square test. Therefore, we applied a χ-square test to assess whether the impact of the banner on the number of first sessions with system activity commands depends on the country for the United States, China, Romania, and Germany. With a p-value of 0.17, the χ-square test, we cannot reject the hypothesis that number of first sessions with system activity commands is independed on the country.

Expected results are higher for the banner configuration than the no banner ones. This is indeed the case for the observed attacks from China. However, the opposite is observed for the United States, Romania, and Germany. These results are surprising since we would have expected the United States and China to have the opposite results. For the attacks initiated in the United States, attacks were expected to be influenced by the banner.

Table 33. First Sessions with System Activity Commands

Country (1st session w/commands)	**Banner** (1st session with w/commands)	**No Banner** (1st session with commands)
United States (29)	11	18
China (20)	13	7
Romania (93)	45	48
Republic of Korea (1)	0	1
Germany (18)	6	12

7.5 Limitations

As previously mentioned, we cannot guarantee that all attackers read the banner or understood it. Also, we cannot differentiate with certainty a bot from a human being.

To prevent the attackers from detecting the nature of the compromised hosts, the honeypots behavior should be close to a regular UMD operated system. The means to convey the surveillance, i.e. the banner announcing the surveillance processes, and the monitoring tools were selected based on several constraints. For example, a banner with counter attach threats is against the UMD policy. A message announcing the monitoring tools in the middle of a session is not a usual behavior for a computer system.

The cybersecurity community has often avoided presenting results based on the IP address since many IP addresses can be spoofed. Authors in [STU12] did show that attackers launching brute force attackers and then compromising a honeypot use two different IP addresses. The experiment described uses randomized assignment of honeypots, which should mitigate some external factors like the use of spoofed IP addresses. Other limitations of the experiment are the duration (31 months), and the location (single location of a US public university).

7.6 Conclusions

The presented study used data collected over 31 months from two honeypot configurations assigned to attackers: Control (no banner) or banner condition. Following a brute force attack on SSH, attackers were randomly assigned a configuration and were granted access for 30 days. We focused on the commands typed by attackers and disaggregate the dataset based on country of origin of the attack, concentrating on the most frequent countries (i.e., United States, China, Romania, Republic of Korea, and Germany).

We explored various metrics: total number of sessions, number of sessions with keystrokes, number of first sessions with keystrokes, system activity commands typed in any session or in the first session. We applied χ-square tests to assess the impact of the presence of the banner and the country from where the attack came from.

In the number of attack sessions, we could reject the hypothesis that the impact of the banner is independent for the five countries. In addition, for all sessions, the use of a banner altered behavior originating in China, Romania and Germany. When focusing on specific commands that provide information on the attacked computer (i.e., ls, who, top, ps), the banner only has an impact on all sessions for attacks from China.

As expected we observed mixed results. The presence of the banner mainly did not have an effect.

While the current study contributes to literature of deterrence on the Internet, this study led to many questions that should be examined by future research that will require additional experiments. For instance, future research should examine questions such as, would other deterrence approaches lead to the same results than the banner? Why did some attacks from China lead to counter-intuitive results? How can we more precisely characterize the attacks from different countries? Are attacks global (since we are now all interconnected) or local (based on the cultural environment of the attacker)?

Chapter 8 – Conclusions

8.1 Summary

Chapter 3 described the distributed honeypot network architecture deployed at the University of Maryland as well as the honeypot framework developed for the cybercrime project. This framework was used to support different honeypot-based experiments, aimed at understanding the attackers' behavior on a system following a "successful" compromise.

The empirical study presented in Chapter 4 determined the relative skill levels of the attackers according to a set of ten criteria. We also classified the different malicious software uploaded and installed on the compromised honeypots according to the software's identified purpose. We showed that the main motivation behind attack is to install IRC-based botnets. We collected evidence to show that about 15% of attackers shared honeypot access with at least one other attacker. We noted that changing password was a frequent action for attackers (77% of attackers updated the target computer password). Only skilled attackers created a new user, but most attackers checked for the presence of users on the system. We also noticed that the configuration of the target did not seem to impact the type of attacker launching the attack.

Chapters 5 and 6 were more focused on the crimes committed (i.e., launched attacks) by the honeypots following a compromise. More specifically in Chapter 5 we focused on three specific dimensions of computer focused crimes: 1) whether the crime was destructive or not, 2) whether the victim of the crime was a target of choice or a target of opportunity, and 3) whether the attack was coordinated or not. We empirically assessed whether 1) the size of the hard drive, 2) the size of the memory, 3) the size of the bandwidth, and 4) the presence of a warning message impacted the type of crime committed. The different empirical studies led to the following conclusions:

- A warning banner is non dissuasive for any of the observed crimes,
- High bandwidth favors any of the observed crimes besides non-destructive crimes,
- High disk space favors destructive crimes, crimes against targets of choice, and crimes involving coordinated attacks,
- Low disk space and high memory space favor non-destructive crimes, and
- Low memory size favors destructive crimes, crimes against targets of opportunity, and crimes not using coordinated attacks.

Chapter 6 highlighted one aspect of cybercrime deterrence: the announcement and existence of surveillance mechanisms. More specifically, we investigated whether computer focused crimes are impacted by a surveillance warning banner or surveillance tools. A farm of four target computers was configured as follows: Type 0) no surveillance banner nor tools, Type 1) a surveillance banner but no surveillance tools, Type 2) no surveillance banner but surveillance tools, and Type 3) a surveillance banner and surveillance tools. The impact of a surveillance banner and/or surveillance tools was analyzed based on the number of crimes, time of the first crime, time distribution for all crimes and the crime rates for the most frequently observed crimes. We observed that none of the target configurations had a statistically significant impact upon these metrics when considering all crimes, but that some impact was measured for some of the most frequently observed crimes.

Chapter 7 presented a study on the impact of a surveillance banner on the attackers' behavior on the honeypot and, more specifically, the differences of behavior depending on the attacker's country of origin. Five countries were studied based on their frequency: China, Germany, Republic of Korea, Romania and United States. Various metrics were employed to identity variations in the attackers' behavior: total number of sessions, number of sessions with keystrokes, number of first sessions with keystrokes, system activity commands typed in any session or in the first session. It was determined that the surveillance banner had no statistically significant effect, however when focusing on specific commands that provide information on the attacked system (i.e., ps, ls, who, top), we noticed that the display of such a banner has an impact on the number of sessions for attacks from China.

8.2 Contributions

8.2.1 Technical Contributions

Chapter 3 presented a distributed honeypot network architecture designed to support honeypot-based research experiments. Factors that were considered in the design included:

- Central, organized and secured data collection
- Safety mechanisms to contain attacks and isolate Honeypots from management and regular networks
- Routing methods to forward remote location honeypot traffic

- Good practices to design and deploy honeypot-based experiments via a security and deployment policy

The cybercrime framework also described in Chapter 3 allowed the deployment of similar experiments involving high interaction honeypots, each experiment randomly exposed attackers to different treatment. This framework design introduced a novel method to:

- Deploy up to three hundred honeypots per experiment due to a lightweight virtualization method
- Ease attackers' access to the honeypots independent from the dictionary they use during the brute-force phase.

The framework also provides a central data collection and storage database.

8.2.2 University of Maryland Security

Both honeypot-based architectures provide valuable information on the attackers and the respective attacks they launch from compromised systems. First, it is not always possible to perform a forensic analysis on campus systems to understand how the device was compromised and to identify the nature of the malicious software implanted on it. Since we have a good knowledge of a "clean" honeypot and we keep an image of the honeypots, it is easy to find malware and the modifications made on the system. Second, from the keystroke dataset and the network flow analysis we can extract:

- IP of the attackers: These IPs can be flagged as malicious and blocked if the traffic generated interferes with the correct operation of the University's network and systems.

- URL used by the attackers to download malicious software on the honeypot can be also collected and used to populate the Intrusion Detection Systems (IPS), a network security device blocking attacks, list of malicious links. The IPS can then flag compromised UMD computers attempting to download malicious software.
- Network flow records of the attacks launched by a compromised honeypot or specific malicious activity from the framework. We then try to identify these attacks in the UMD netflow records.

8.2.3 Science of Cybersecurity

The empirical studies in Chapters 5, 6 and 7 showed that the warning and surveillance aspects of the deterrence have mainly no significant effect on the attackers' behavior. Despites these negative results, the studies followed a rigorous scientific approach to test our hypothesis: we formulated research questions and designed methodologically rigorous experiments to generate sufficient data to perform strong statistical analyzes.

8.3 Limitations

The primary limitations of the presented studies reside in the validation. The validation requires both the replication and reproduction of the results. To replicate the results the same experimental design and methodology should be used in another location. Reproducing the results implies developing another experiment to collect the data and another methodology to analyze them. Both replication and reproduction require 1) the deployment of sensitive and risky experiments, and 2) the sharing of security data.

In addition, the re-analysis of the results may not be possible as the security context is constantly changing. New attacks and vulnerabilities appear every day.

8.4 Future Work

One problem that is not totally addressed by this work is the replication of the experiments in another location. Despite the current ongoing collaborations with different universities and companies, the number of external IP addresses is not sufficient to replicate the experiments at the same scale. We will need to develop collaborations so that the studies described in Chapter 4, 5, 6 and 7 can be replicated and the results compared.

To completely cover the deterrence theory, we will need to test the effect of barriers on the attackers and attacks. We suspect that introducing obstacles to the compromise and exploitation of the honeypots may impact the behavior of attackers.

A growing number of attacks are now automated. We currently use the timestamps to identify sets of commands that have been typed "too quickly" and some other keystrokes characteristics such as UP-ARROW or BACKSPACE. This method presents a few limitations of its own: it is possible that attackers use a tool that replicates sessions, in that case UP-ARROW and such will be replicated as well. Network delays can impact the timestamp and prevent the detection of automated sessions. In that area, additional work is necessary to identify other characteristics of automated sessions independent from the keystrokes timestamps.

Future work will consist in developing new experiments in close collaboration with various fields in the social sciences such as criminology, economics, or psychology.

8.5 Conclusion

Over the last five years I have been involved in several research projects that can be grouped according to three aspects:

- A collaboration with City University in London on antivirus detection, regression and label changes,
- The development of a network of honeypots to support large scale empirical studies on malicious data, and
- The design, implementation, and result analysis for several empirical studies in collaboration with criminologists.

The first research project (not described in this manuscript) led to 3 published papers [GAS12, GAS13a, GAS13b], two of which were in highly competitive conferences [GAS13a, GAS13b]. The second research project led to one publication (as first author) [SOB11]. The third research project led to two journal articles [MAI13, MAI14] and two papers in highly competitive conferences [SAL11, SOB12].

In this dissertation, we have shown how to successfully develop empirical experiments in cybersecurity. This has been made possible through the collaboration with the Security Team at the Division of Information Technology.

Appendix A: CyQLNet

CyQLNet is a network architecture aiming at providing a safe computing environment for the researchers and collaborators of the Cybersecurity Quantification Laboratory (CyQL). The network infrastructure is designed to isolate and protect the research resources (data and servers) from the external world but also from the honeypot network operated by the research team while simultaneously providing a means to easily access and manage the data generated by the honeypots.

A.1 Network Architecture

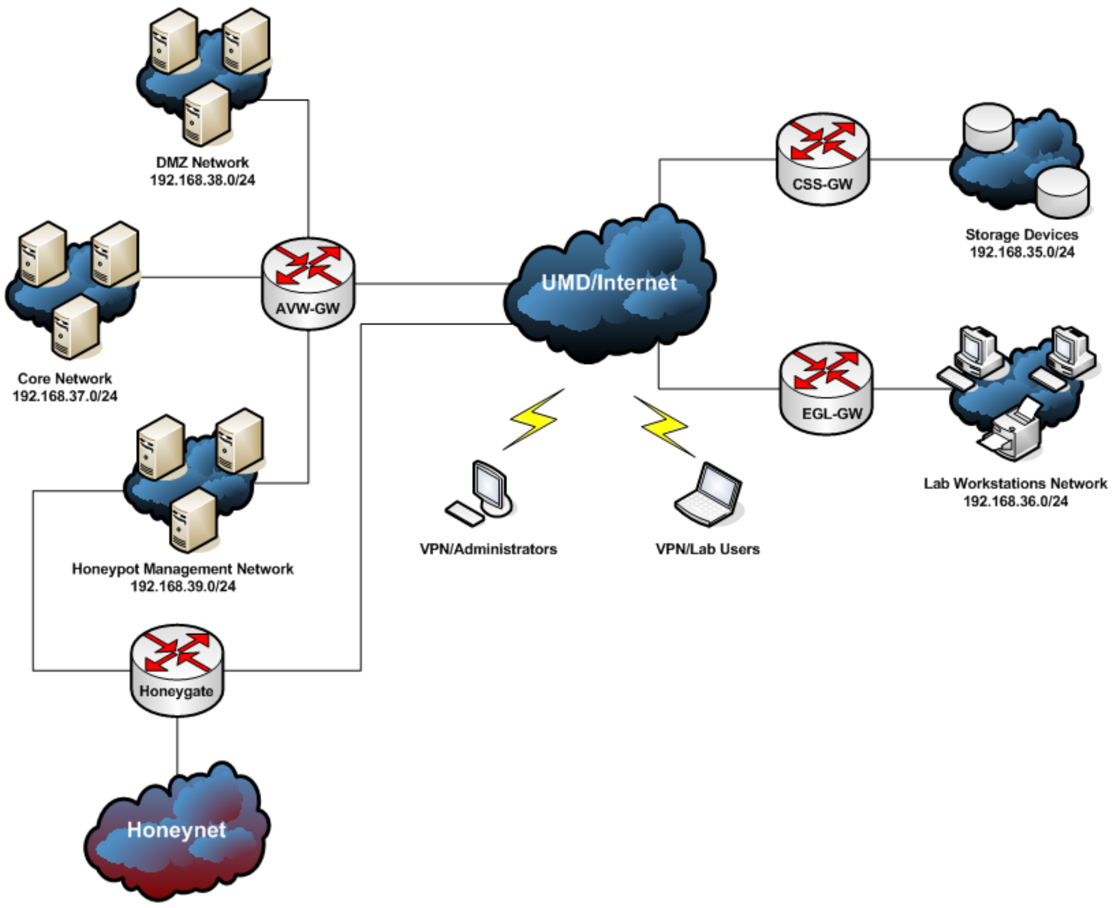


Figure 34. CyQL Network Architecture

As shown in Figure 34, CyQLNet consists of six different networks distributed across three different locations. A rack in the datacenter location is housing all of the servers as well as some storage devices (See Figure 35). The location in Engineering Lab Building (EGL) is the physical lab with the research team's workstations. The Computer and Space Science location is dedicated to storage devices. All of these sites are connected through the University network using secured encrypted communication channels. The gateways EGL-GW and CSS-GW establish a tunnel with AVW-GW. AVW-GW forwards transparently the traffic from the different networks using the tunnels.

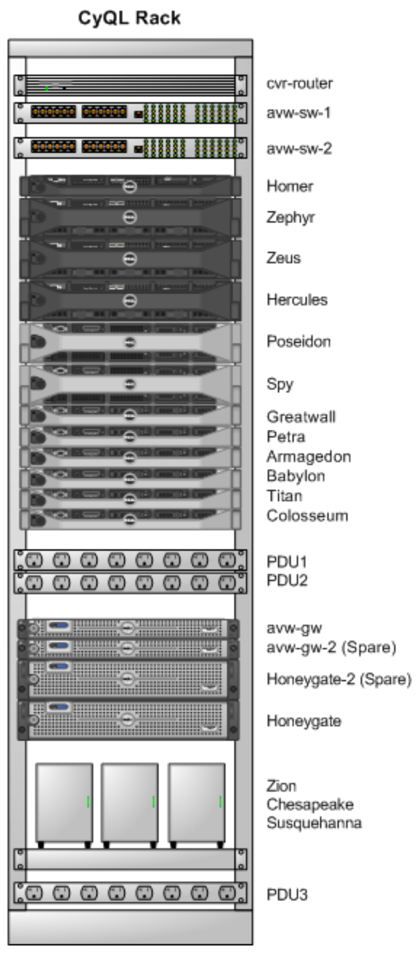


Figure 35. Datacenter Server Rack

A.1.1 Core Network

The core network hosts the lab general-purpose server and the management interfaces of the critical infrastructure devices including switches, network attached storage devices, gateways and server remote access control. It's a private secured network that is not directly accessible from the Internet.

A.1.2 Core Network DMZ

By definition, a DMZ or demilitarized zone is a sub-network that hosts services accessible from an external network such as Internet. In our case, the DMZ network is used to provide access to internal web applications from the Internet without having Internet hosts interacting directly with the servers on the Core Network. The DMZ hosts a web server and a server used to exchange data with outside organizations or hosts.

A.1.3 Honeynet

This network is dedicated to host the public interface of honeypot-based experiments. The machines connected to this network can receive the Internet traffic of different organizations. Its infrastructure is described in Section 3.1 of Chapter 3.

A.1.4 Honeynet Management Network

The UMD Honeynet deployment policy described in Section 3.1.4.2 of Chapter 3 recommends each experiment have a separate private management interface. This interface allows the experiment administrator to access and perform maintenance operations on the honeypots involved. It also permits the download of data collected by the honeypots for backup or processing.

Because of their nature, the Honeypots can get compromised and the attacker(s) can use them to access the internal network. The firewall rules on the main gateway prevent any hosts on the Honeypot Management Network from accessing the other networks and the Internet. Anything that is connected to the Honeynet network and that has a private management interface will be connected to this network. The Honeypot Management Network acts as an insulation layer while allowing the experiment users to manage their honeypot and download their data.

Using a separate interface presents another advantage. Any traffic observed on the Honeynet is considered as malicious. The attackers might detect the honeypot access for management purpose. In addition, the Honeynet traffic would have to be filtered to remove any "legitimate" traffic from the data collection.

A.1.5 Virtual Private Network

As previously mentioned, most of the networks operated by CyQL are private, and thus not accessible from the Internet except for the DMZ.

Virtual Private Network (VPN) technology is used to create a secured tunnel between a remote host (or network) and the main network gateway (AVW-GW) across a public network such as the campus network and the Internet. Once established, the VPN tunnel allows the remote host or network to access the private networks. There are two different VPN services available to access the internal networks of CyQL.

- The Lab Users VPN profile permits access to a specific set of servers within the Core and Honeypot Management Networks.
- The Administrators VPN allows access to every network and device without any restrictions.

A.2 Network Components

A.2.1 Gateways

a) AVW-GW and AVW-GW-2

AVW-GW is the main gateway for CyQLNet. It interconnects all the networks except the Honeynet. It provides Internet access to the networks hosted in the datacenter if permitted by the firewall.

AVW-GW-2 is a spare gateway. It is a replicate of the main gateway ready to take over in case of failure of AVW-GW.

PfSense [PFS15] is based on FreeBSD Operating System and customized to run as a firewall and router. It is entirely manageable through a web interface and offers advanced firewalling and routing functionalities allowing us to connect different sites and isolate the Honeynet.

AVW-GW provides different network services such as:

- IPSec VPN tunnels to interconnect the different locations of CyQLNet.
- VPN services to allow remote access of the networks
- A time server to help synchronize clocks on the different servers of CyQLNet
- Automatic IP address configuration on the Core and Honeypot Management Networks
- Provides Domain Name Services on the Honeypot Management Network

Access Control List

One of the key functionalities of the gateway is its ability to forward traffic from one network to another if permitted by the access control list shown Table 34. The ACLs presented in Table 34 only show the global traffic filtering. A number of firewall rules allow a more fine-grained access control based on host IP addresses and services ports.

Table 34. AVW-GW Access Control List

Source	Destination	Action
Core Network	*	Allow
Honeypot Management Network	*	Deny
Core Network DMZ	*	Deny
EGL Lab Network	Core Network	Allow
EGL Lab Network	Honeypot Management Network	Allow
EGL Lab Network	Core Network DMZ	Allow
EGL Lab Network	CSS Storage Network	Deny
EGL Lab Network	*Internet*	Allow
CSS Storage Network	*	Deny
VPN Administrator	*	Allow
VPN Lab Users	Core Network	Deny
VPN Lab Users	Honeypot Management Network	Deny
VPN Lab Users	Core Network DMZ	Allow
VPN Lab Users	CSS Storage Network	Deny
VPN Lab Users	EGL Lab	Allow

b) EGL-GW and CSS-GW

Both gateways are running the same version of PfSense as AVW-GW and establish tunnels with AVW-GW to interconnect the distant networks across the UMD network.

EGL-GW provides Internet access to the lab workstations and to the different networks allowed by the ACLs shown in Table 34. CSS-GW only allows traffic from specific servers in the Core Network to the storage devices. Since Internet is not necessary in this location, it is disabled.

CSS-GW and EGL-GW provides Domain Name Services (DNS) and IP auto configuration.

c) Honeygate

Honeygate is the main gateway for the Honeynet. It also runs PfSense. Its functionalities are described in Section 3.1.2.1 of Chapter 3.

A.2.2 Switches

All of the network devices are connected to two switches. These switches are using Virtual Local Area Network (VLAN) to partition one network into several logical networks. Depending on the switch port, a VLAN ID tag is added to each frame sent by a device on the network. This VLAN ID indicates which network partition the frame belongs to. Each VLAN is a distinct network, a device on VLAN 1 can only communicate with the other devices on VLAN 1. It is not possible to jump from one network to another.

The VLAN configuration is done port by port on the switches. Most of the times, the VLAN tag is added to the frame once it enters one of the switches and removed when it leaves the switch. Some of the switches ports are called “trunk ports”. These ports are assigned two or more VLANs. The devices attached to these trunk ports are configured to support VLANs and to communicate on different networks. Both switches (*avw-sw-1* and *avw-sw-2*) are connected to each other using trunk ports and share the same VLAN configuration shown in Table 35.

Table 35. Virtual LAN Definition

VLAN ID	Description
2	Core Network
3	Core Network DMZ
4	Honeynet
5	Honeypot Management Network
6	UMD Network

A.3 General Purpose Server

A.3.1 Zeus

Zeus is one of the main servers of the Core Network. It provides core services for the whole network including authentication, directory and file sharing services. It is running Linux Ubuntu Server 10.04 Long Term Support.

a) Active directory

Active Directory (AD) is a Microsoft product used to authorize, authenticate and manage users and computers on a Windows Domain. AD is usually operated on a Windows Domain Controller Windows Server operating system. In our case, we use Samba, a re-implementation of the main Windows file and printer sharing protocols on UNIX systems. Samba version 4 can act as a Domain Controller and provide Active Directory Services for Windows and Linux servers. To do so, a Domain Controller part of an Active Directory relies heavily on the following services:

- Domain Name Services (DNS): A DNS server is in charge of translating a machine name such as www.umd.edu into an IP address usable to contact a host on an IP network.

- Kerberos: Kerberos is an authentication protocol using secret-key cryptography. It provides strong authentication of client and services using tickets.
- LDAP: Lightweight Directory Access Protocol is used to connect to, search and modify a directory. A directory stores and organizes the different network entities such as services, hosts, users and groups.

On the CyQL Network, the Active Directory is used to centrally manage the authentication. As a consequence, each user has one credential that he or she can use to log into workstations, servers and applications.

In addition to the authentication, the Active Directory is used to authorize access to workstations, servers or applications. Authorizations are based on group memberships. This allows us to control who can use a specific server, workstation or services. It is also used to give users additional privileges on some servers or workstations (ability to become administrator of the machine).

b) File Sharing Service

Three different shared directories are available on Zeus and mounted automatically on Windows and Linux workstations and servers.

- Home Directories: Each user has a secured space to store documents and other files.
- Projects Directory: A common space for each research projects.
- Common Directory: A common space for all users to share files.

c) Servers Resources Monitoring

We use *Munin* [MUN15] to collect data on the critical system resources such as disk, memory, network and processor usage. It generates graphs representing a year worth of data points for each of the monitored resources. Munin is used to identify the root cause of performance issues and help with the hardware upgrade decision-making process.

d) Backup

Zeus runs *BackupPC* [BAC15] to backup user files of the File Sharing Service as well lab a set of specific directories on the lab servers. *BackupPC* does not require the installation of a client on each server. It uses Rsync, a file transfer tool, over Secure Shell protocol and public key authentication to access the remote hosts directories to duplicate.

BackupPC creates the backup files on a network storage disk NFS share mounted on Zeus. In its current configuration, *backupPC* performs and incremental backup every night (only the files that changed will be backed up again) and a full backup once a week. We currently keep up to seven days of backups. A deleted file can be recovered for up to seven days but after that delay, it is removed from the backup.

The directories to backup as well as the servers can be easily added through the web interface. The web interface also provides access to the backed up files and their different versions.

Each server in the lab will have its configuration (`/etc` directory) backed up every day.

e) Servers Central Management

The administrator account on Zeus can establish remote administrator sessions via the Secure Shell protocol with every lab computer and server using public key authentication, a password-less method to authenticate users. In addition to being able to get a remote shell without providing a password from the administrator session on Zeus, this mechanism allows us:

- To push new configuration files to all the servers automatically
- To perform maintenance operations automatically by running script on remote hosts.

A.3.2 Poseidon

This server is dedicated to heavy data processing and large database hosting. When a script is time-consuming and requires more resources than a regular desktop, Poseidon is used in order to execute it. The user allowed to access that machine uploads the script or tool along with the data required for the processing task. Poseidon has a fast access to the data generated by the Honeynet.

A.3.3 Zephyr

Zephyr hosts all the lab web applications common to all research projects. Most of these applications are actually accessible from the Internet via the Proxy server. Two types of web applications are hosted on Zephyr. *CAS* [CAS15] and *SCM-Manager* [SCM15] are Java-based applications ran by Apache Tomcat and OpenProject is a Ruby based application ran by Rails/Unicorn.

a) Central Authentication Services (CAS)

CAS is a single sign-on web portal used to authenticate lab users on all the web-based applications. When accessing one of the lab web applications, the user is required to authenticate with CAS once. The user can access other applications without proving his or her identity again.

CAS contacts Zeus Active Directory services to verify the authenticity of the user's login name and password.

b) Source Control Management (SCM) Manager

Software and script developers often use source control software to manage the different revisions of their code. These tools are used to create code repositories. They keep track of the different versions or revisions of the source code. Each revision will be associated with a timestamp and the author of the change. Usually, the different versions can be compared, restored and sometimes merged.

These code repositories can be shared with different developers. We use SCM Manager to provide for the lab users:

- A web interface to create and visualize the different code repositories
- An URL that can be used by the different source control tools to access the repositories remotely
- Authentication and authorization mechanisms to control the access to the different repositories. SCM-Manager uses Active Directory groups to grant users to repositories.

SCM Manager supports Git [GIT15] and SVN [SVN15] source control software.

c) OpenProject

The web-based application OpenProject [OPE15] is an open source project management tool. Each research project will have a dedicated workspace providing tools such as Wiki (for documentation), file sharing, source code repository access, timeline tracking or task tracking. The tools can be added to a project according to the team members' needs. For large projects, it is possible to create sub-projects within a project workspace. OpenProject relies on the lab Active Directory Services to authenticate and authorize users to access the different project workspaces. Group membership will dictate which projects a user can access and modify.

A.3.4 Zabbix Appliance

Zabbix [ZAB15] monitors the different critical resources and applications of each computer system (physical and virtual) such as free disk space, memory, processor usage and the state of the different services provided by the server. Zabbix server, running on the Zabbix Appliance virtual machine, contacts the Zabbix agents installed on all the servers monitored by the tool to gather data on specific system resources and services. The server then saves the data in a database and triggers an alert if all the conditions for a specific situation are met. Most of the time, when a system resource becomes low or an application goes offline, an alert is sent to the administrator. Another email is sent when the situation has been solved.

The visualization of the alerts as well as the configuration of Zabbix server is made possible through Zabbix web interface. This interface is entirely customizable; the data collected by the server can be presented in a table, graph or even a map. It is also possible to customize the data that can be gathered by the server as well as the method to collect them. For example, Zabbix can use network management protocols such as SNMP (simple network monitoring protocol) to get a numerical value or a string indicating the status of a network device that is not running the agent.

A.3.5 DMZ Servers

a) Guard

Guard is a temporary file storage area used to receive or send data from or to an external entity. For example, a UMD network server exports the event logs of a security device for the purpose of a research project every day. These logs are uploaded on Guard at midnight. Later, Poseidon downloads them and erases them from Guard.

Guard only allows Secure Copy Protocol (SCP) sessions, a transfer protocol over the Secure Shell Protocol. Regular SSH sessions are not permitted except from Zeus.

b) Proxy

Proxy is an intermediate server used to access the lab web applications from the Internet. It allows us to protect the core servers. These servers are not directly accessible from the Internet and as a consequence less vulnerable to web attacks.

The proxy contacts the servers running the web applications only when the users have been authenticated. Proxy can be the victim of an attack. The damages would be limited to this machine and would not affect the internal servers.

A.4 VMWare Cluster

Nowadays, the Information Technology industry relies more and more on virtualization. Thanks to cloud technologies, virtual servers can be created or destroyed according to the demand of computing resources. In the past, a physical machine was dedicated to provide one core function, and was not used at 100% of its capacity. Virtualization makes it possible for several hosts to share the same hardware reducing the costs of equipment and minimizing maintenance. Advanced virtualization software also makes it easy to create, duplicate or move virtual machines.

The CyQL Network virtual environment is based on *VMware vSphere ESXi* [VMW15] (ESXi host), a bare-metal hypervisor, and *vCenter*, a central management tool for *ESXi*.

Vmware ESXi runs its own dedicated operating system optimized to only support the core features of the virtualization software. On the host itself, Vmware ESXi provides only basic network configuration options. An *ESXi* host can be standalone and an application called *vSphere* client is used to manage the host and the virtual machines.

An ESXi host can be part of a cluster managed by *vCenter*. When connected to a *vCenter* server, the *vSphere* client allows the management of the different *ESXi* hosts participating in that cluster and the different virtual machines.

vCenter can provide advanced functionalities such as high availability, load balancing, shared storage or shared virtual network interfaces. It is possible to manage fully virtualized datacenters with this tool.

In our case, even though all of the hypervisors are attached to *vCenter*, the network interfaces and the storage are configured locally on each *ESXi* server. In addition to its central management console role, *vCenter* allows us to move or duplicate a virtual machine from one server to another server.

In the current configuration of the *Vmware* Cluster, we count three types of hypervisors dedicated to host specific virtual machines:

- Management Hypervisors: dedicated to servers of the core network
- UMD Honeynet Hypervisors, dedicated to host honeypot virtual machines.
- Cybercrime Hypervisors, dedicated to the Cybercrime project virtual machines. These virtual machines require a different network configuration.

As previously mentioned, the network configuration is local to each hypervisor. Depending on the type of hypervisor the virtual machine is hosted on, it will be possible to connect it on different networks. This is done to preserve networks isolation. Honeypots virtual machines should be connected to the UMD Honeynet and Honeynet Management networks and not on any other network. Table 36 shows the list of hypervisors and their respective network configuration.

Table 36. Hypervisors Network Configuration

Hypervisor	**Core Network**	**Core Network DMZ**	**Honeynet**	**Honeynet Management**	**Cybercrime Management**	**Cybercrime Honeypot**
Management	■	■				
Homer	■	■				
Hercules	■	■				
Cybercrime			■	■	■	■
Babylon			■	■	■	■
Armagedon			■	■	■	■
Titan			■	■	■	■
Honeynet			■	■		
Colosseum			■	■		
Petra			■	■		

A.5 Data Storage Devices

The storage devices are network hard drives used to duplicate the data collected by the different honeypot projects. The disks are split between two locations to ensure redundancy of the data in case of the complete loss of storage devices in one of the locations.

A.5.1 Disk Configuration

a) Access

The data stored on the different network disks are accessible through the UNIX file sharing protocol called Network File System (NFS). Each storage device will export or share a directory that will be mounted on the servers of the core network.

Because of the critical data these disks may host, the storage devices are only accessible by the main data collection server and Zeus. All the backup processes are all initiated by either of these two hosts.

b) Fault Tolerance

All the storage devices have one or more disks. Disk failure is a relevant concern. To prevent data loss, two different levels of Redundant Array of Independent Disks (RAID) mechanism are used to combine multiple drives into one logical disk for the purpose of data redundancy:

- RAID 1 or mirroring, when the drives are in pair and mirrored. Disk 2 is an exact copy of disk 1.
- RAID Level 5, when the data is organized in blocks and these blocks are distributed across different along with an additional parity block. This RAID

level works with 3 and more drives and can withstand one drive failure at a time.

As shown in Table 37, the use of redundancy is disk space consuming. For a total of 52 TB of hard drive storage, 18 TB are used for the purpose of redundancy.

The disk drives could be combined together. This configuration is called RAID 0 or JBOD (Just a Bunch of Disks). The disks are merged into one logical drive and the data blocks are spread across the two drives. This RAID level presents a serious drawback: if one drive fails, all of the data are lost.

Each disk drive could be used independently but distributing the data across the 18 different disks would difficult to manage. Also if one drive fails, up to 4TB of data can be lost.

Table 37. Storage RAID Levels

Device	Disk Configuration	Total Size	RAID Level	Effective Size
Zion	5 x 4 TB	20 TB	RAID 5	14 TB
Yosemite	5 x 4 TB	20 TB	RAID 5	14 TB
Susquehanna	2 x 1 TB	2 TB	RAID 1	1 TB
Chesapeake	2 x 1 TB	2 TB	RAID 1	1 TB
Shenandoah	2 x 2 TB	4 TB	RAID 1	2 TB
Assateague	2 x 2 TB	4 TB	RAID 1	2 TB
Total		*52 TB*		*34 TB*

A.5.2 Backup Policy and Data classification

To identify how data storage and backup should be handled, a data storage and backup policy has been defined. Each dataset is classified according to a level of criticality. This level of criticality will dictate how and where to store the data.

a) Level of criticality "High"

Data that we will not be able to recreate if lost. For example, datasets generated by an experiment.

- This data should be stored on a server with RAID Level 1 minimum
- This data should be duplicated on two other storage devices, in two different locations
- This data should be backed up twice a day.

b) Level of criticality "Medium"

Data that we can re-create but the process can be time consuming. For example, scripts analyzing the data collected by an experiment.

- This data should be stored on a server with RAID Level 1 minimum
- This data should be duplicated on another storage device, in a different location than the server storing that data initially
- This data should be backed up every day at night after business hours

User files, source code repositories, servers configurations, non-critical databases and OpenProject data fall under that category.

c) Level of criticality "Low"

Large volume of data generated by analysis scripts or downloaded again.

- This data should be stored on a server with RAID Level 1 minimum
- Data duplication is optional
- Back up periodically from every day to once a week depending on the changes

Appendix B: UMD Honeynet

B.1 Traffic Filtering

The Honeygate host is designed to be the UMD Honeynet's single point of entry. Per this design, every single packet received from the Honeymole tunnels, the UMD internal network and the UMD border network goes through this gateway. Even though this may be seen as a single point of failure, this design permits traffic shaping and filtering to 1) shut down the Honeynet in case of emergency, 2) block an IP address in the event of an uncontrolled compromise and 3) limit the action of a controlled compromise on the Internet and the UMD network.

Honeygate also maintains a blacklist of IP addresses that should be blocked. These addresses can belong to the Honeynet, but they can also be Internet IP addresses. As shown in Figure 36, there are two main points off traffic inspection on Honeygate:

- In front of each network feed (traffic providers such as Honeymole, border network and UMD Network) where we only block the IP addresses from the blacklist.
- In front of the Honeynet network interface where we block traffic targeting the UMD network and the blacklisted IP addresses.

An IP address can be blacklisted for several reasons:

- A honeypot is compromised beyond control.
- A honeypot is targeting a specific IP address on the Internet. To protect that host, the IP of the victim is blocked.

- An Internet host targets the Honeynet. The IP is blocked to protect the Honeynet.
- Several Internet hosts target one Honeypot, the Honeypot IP is blocked to protect the Honeynet.

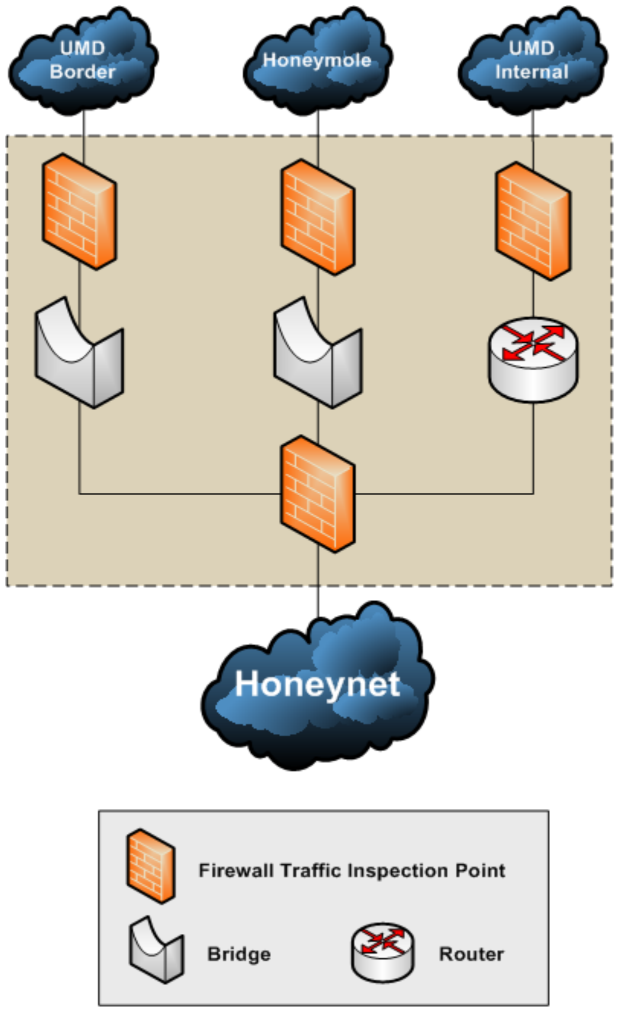


Figure 36. Honeygate Architecture

B.2 Traffic Routing

The traffic of all of the organizations taking part to the UMD Honeynet is directed to the honeypot network using different methods. As depicted in Figure 36, the Honeygate gateway is designed to handle different sources of traffic: direct routing

(UMD Internal), bridge for Honeymole tunnels, and bridge for GRE tunnel. A network bridge connects two network segments. It learns what devices are on each side. The traffic will be forwarded from one segment to another if the source and destination hosts are not on the same network segments. The bridge is transparent and allows us to observe and filter traffic.

B.2.1 Direct Routing

The Honeygate host is advertised as the main router for all the campus internal sub-networks dedicated to the honeypots. In that configuration, the Honeygate host is behaving like a regular router.

B.2.2 Generic Routing Encapsulation Bridge

GRE is a tunneling protocol that permits the Division of Information Technology to tunnel down the traffic from the campus border routers to an intermediate router connected to the Honeygate. Like for the Honeymole servers, the traffic received from this intermediate router is forwarded to the Honeynet via a bridged network interfaces to permit packet filtering.

The traffic from the campus border network is neither filtered nor protected by the University's intrusion detection systems.

B.2.3 Honeymole Tunnel

Honeymole [HON15] is a tunneling program that creates a secure communication bridge between a Honeymole client and server. The client, hosted on the remote location network and the server, hosted on the University of Maryland network, both capture the required traffic to port the external entity honeypot IP addresses to the UMD Honeynet. The Honeymole server at UMD is a virtual machine hosted on one of the Honeynet hypervisors. As shown in Figure 37, the server is connected to the Internet via a Firewall. This firewall only allows the traffic created by the communication tunnel established with the client. The Honeymole server injects the traffic received by the Honeymole client to the Honeygate on one of its bridged interfaces. The network bridge will then forward the traffic to the UMD Honeynet. This server is also connected to the Honeypot Management network to permit administration tasks and monitoring.

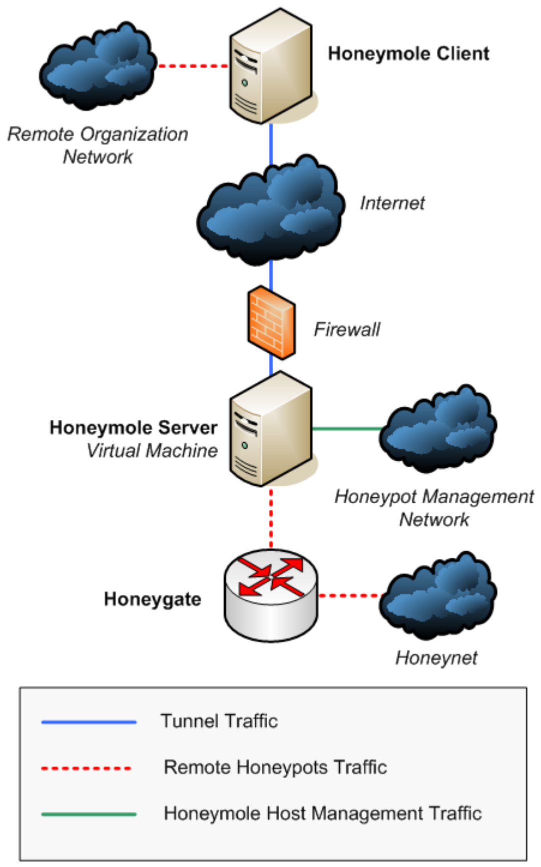


Figure 37. Honeymole Architecture

The Honeymole Server is running on top of a Linux OS. We have a Honeymole Server virtual machine for each remote organization giving us IP addresses for the Honeynet.

B.3 Honeypots Database

In addition to the data collection (Section 3.1.3 of Chapter 3), the data collector Spy hosts the database that keeps track of the different honeypot-based experiments as well as all the UMD Honeynet IP addresses. With about 2,000 IP addresses, it is important to keep track of the various past and current experiments as well as their IP address allocations.

B.3.1 Database Schema

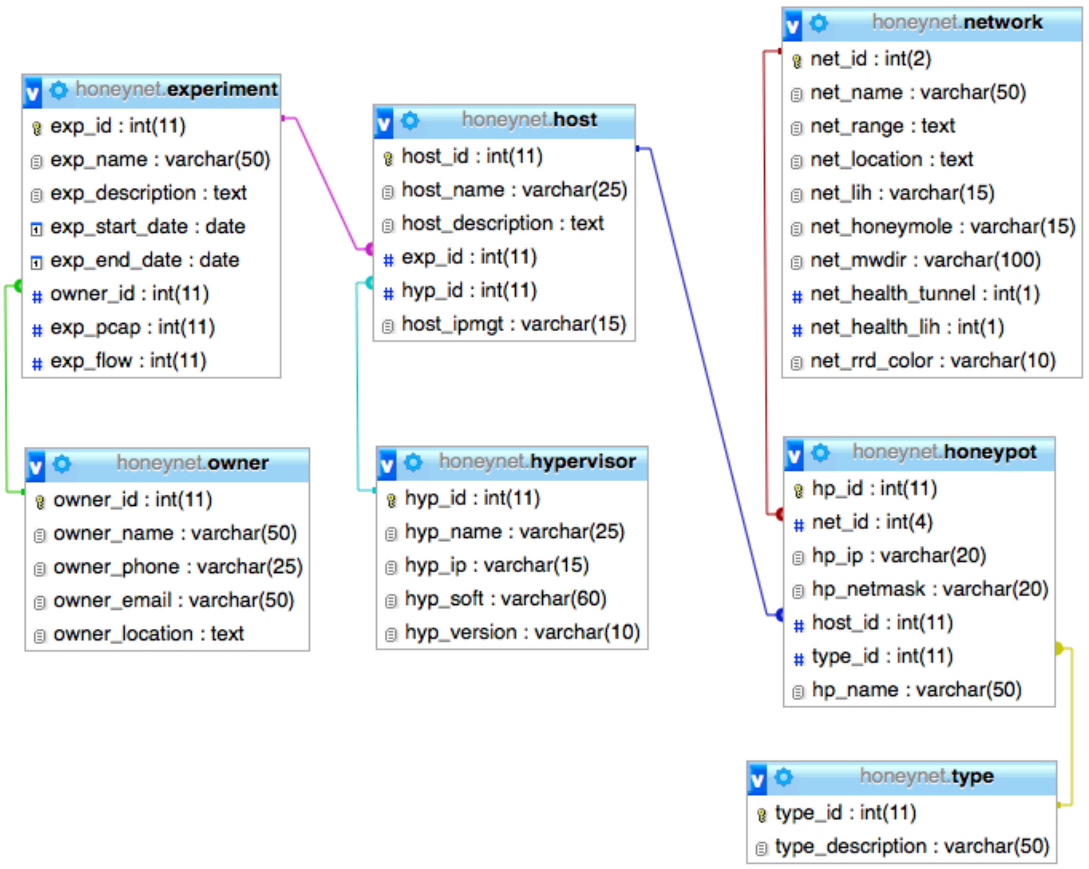


Figure 38. UMD Honeynet Database

As shown in Figure 38, the UMD Honeynet database consists of seven tables providing information regarding the honeypots, where they are hosted (virtual machine), their IP addresses and the experiment they belong to.

a) Experiment and owner tables

The *experiment* table keeps track of all of the experiments including their start and end date, and their description. The fields *exp_pcap* and *exp_flow* are both flags indicating whether the traffic and flow data specific to the experiment should be created during the log rotation.

The *experiment* table is linked with the *owner* table. The owner table contains the contact information of the person responsible for the experiment.

b) Host and Hypervisor tables

The *host* table contains the information related to the virtual hosts running one or several honeypots. Virtual machines are usually hosted on a dedicated physical host called hypervisor. The *hypervisor* table allows us to specify on which hypervisor the virtual machines are located. This table provides for each system its IP address, the software version, and its name.

c) Honeypot, network and type tables

The *honeypot* table provides information including the IP address configuration, the level of interaction (Link with *type* table) and the hosting machine (link with *host* table) for each honeypot deployed on the Honeynet.

The *honeypot* table is also linked with the *network* table. This last table contains the list of the different subnets that compose the Honeynet. For each subnet the table provides information on the IP allocated, details on the location, the IP address of the subnet LIH host, the IP address of the Honeymole host (if applicable) and the location of the malware repository on Spy. In addition to all these, one of the monitoring tools maintains the health status of the Honeymole tunnel and Low Interaction Host. The *net_rrd_color* contains the color code representing a specific subnet in all the graphs generated by the different monitoring tools.

B.3.2 Database Usages

The primary usage of the databases is to be able to track down a honeypot from its IP address. The need to identify a honeypot from its IP address mostly happens during incidents investigations or traffic analyses.

The database is also used to build the traffic filters for each experiment and subnet used during the network traffic log rotation. The database allows us to create a traffic file for each experiment and each subnet automatically. There is no need to modify a script or a configuration file when adding a new experiment or subnet.

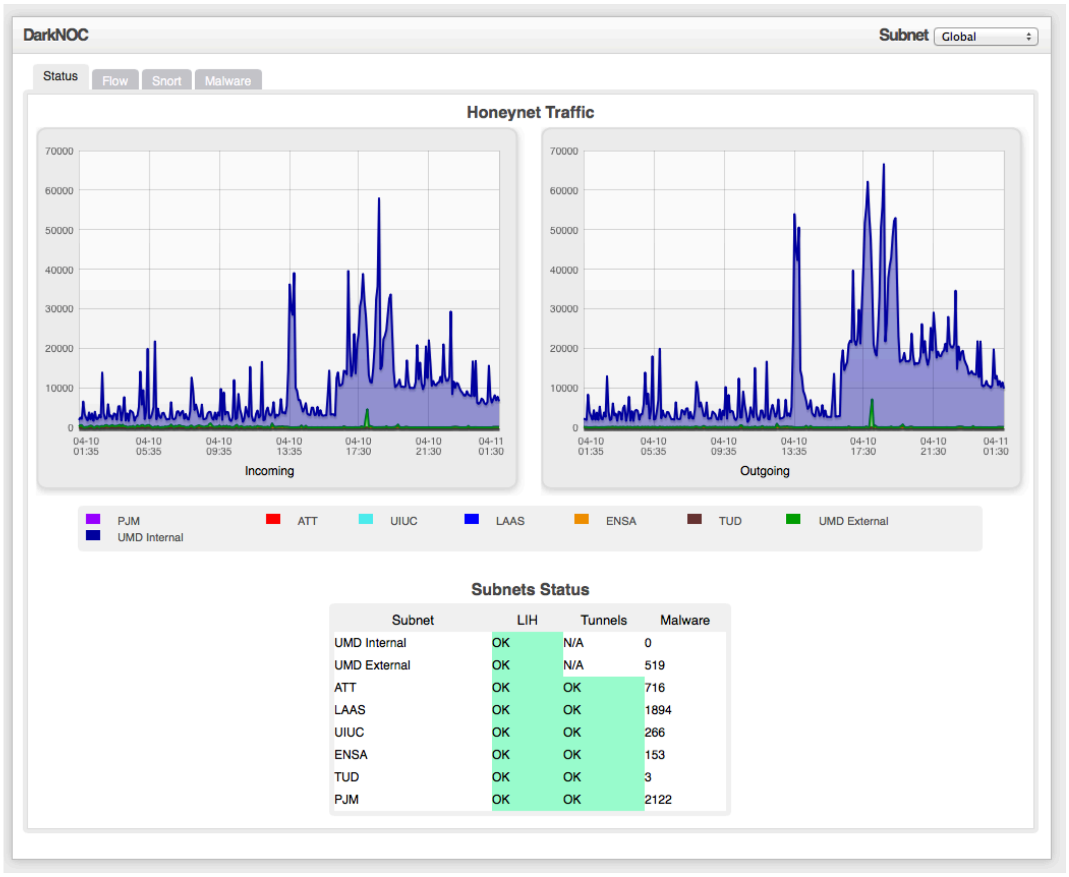
A set of scripts uses the database to generate the IP configuration of the LIH hosts. The scripts extract the IP addresses linked with the different LIH hosts in the database, generate the network file configuration file and push it to the different LIH hosts.

B.4 Monitoring and Alerting Systems

B.4.1 Honeynet Monitoring: DarkNOC

The Honeynet requires constant monitoring to guarantee that protection systems (for example firewalls, traffic shapers) and data collection are operating correctly. The volume of data collected daily can be important and significantly impacts data processing and extraction.

DarkNOC is a solution designed to efficiently process large amount of malicious traffic received by a large honeynet, provide a user-friendly Web interface to highlight potential compromised hosts to security administrators and provide the overall network security status.



Subnet	LIH	Tunnels	Malware
UMD Internal	OK	N/A	0
UMD External	OK	N/A	519
ATT	OK	OK	716
LAAS	OK	OK	1894
UIUC	OK	OK	266
ENSA	OK	OK	153
TUD	OK	OK	3
PJM	OK	OK	2122

Figure 39. DarkNOC Web Interface

The graphical user interface organizes the different data necessary to present a summary of the honeypots activity. Figure 39 shows the homepage of DarkNOC.

The graphical user interface first provides a global view of the activity of the honeypots: the data displayed pertain to all of the subnets. The user then has the possibility to reduce the scope of analysis to one subnet. The web page provided by DarkNOC is divided into four different sections: 1) the subnets status, 2) the flow-based information, 3) the snort events and 4) a summary of the malware collection.

B.4.2 Alerting and Reporting Tools

a) LIH Watch

A first script connects to the Low Interaction Honeypots hosts to check whether the LIH tools (Dionaea and fake SSH server) are running. If not, an email is sent to the administrator and an attempt is made to restart the service. This script is executed every six hours.

b) UMD Hosts Alert

Another script executed at 6am and 6pm looks for any host from the University of Maryland network that has attempted to communicate with the Honeypots. Since the honeynet is not hosting any legitimate traffic, any attempt from a UMD host to connect to a honeypot is suspicious and is investigated.

c) Scanner Detection

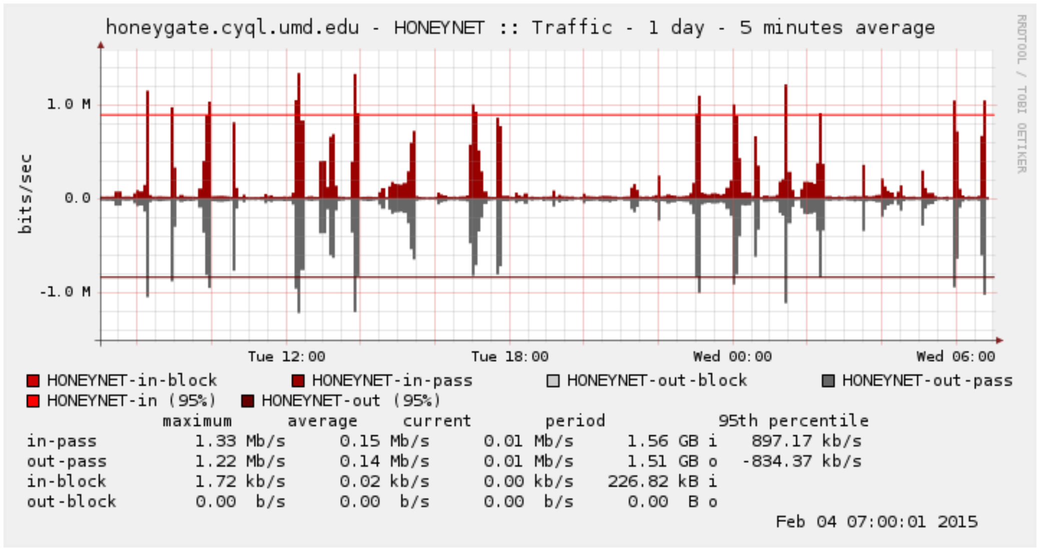
Every hour a script uses the netflow repository to identify potential Secure Shell (SSH) and Remote Desktop (RDP) scanning the Internet. To do so, the flows are aggregated per honeypots and per destination ports. If for port TCP/22 (SSH) or TCP/3389 (RDP) the number of destination IPs exceeds a certain threshold for one honeypot, this honeypot is flagged and an email is sent to the administrator for investigation. This detection is done after traffic filtering. The purpose is to make sure the security measures are still efficient.

d) Phishing Websites Detection

The same script then checks for potentially exploited web servers within the Honeynet. Some attackers often use a compromised honeypot to host a phishing website. The script detects potential phishing websites by looking at the number of different hosts from the Internet trying to access web servers (port TCP/80) on the Honeynet. When such a website is made available online, the traffic towards the web server hosting it significantly increases. The list of potential web servers hosting a website is part of the report sent to the administrator every hour.

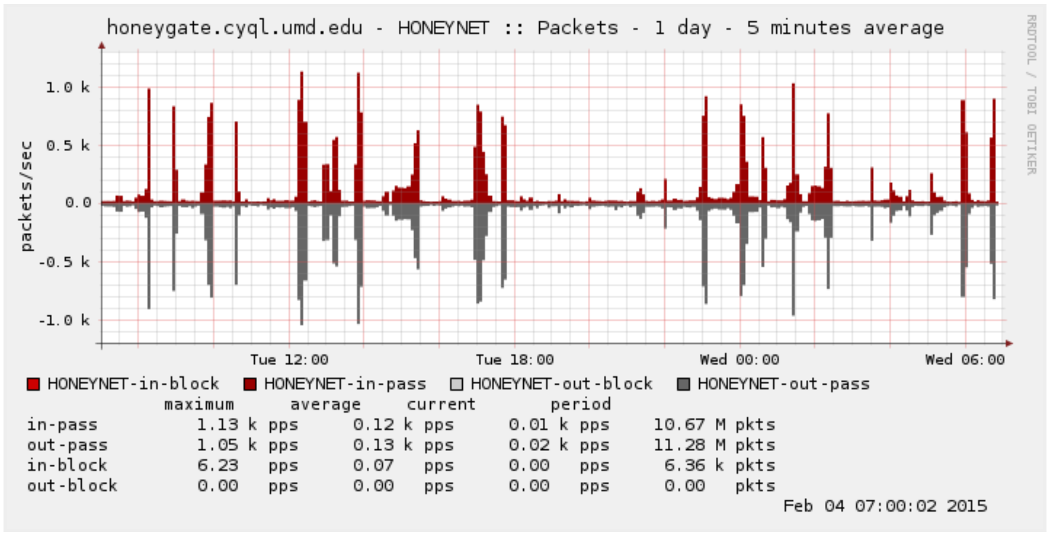
e) Traffic Graph Report

In addition to DarkNOC, the Honeygate also sends a mail report on the Honeynet traffic. It shows the number of bits (Figure 40) and packets (Figure 41) per seconds for the incoming and outgoing honeynet traffic.



```
■ HONEYNET-in-block        ■ HONEYNET-in-pass       □ HONEYNET-out-block       ■ HONEYNET-out-pass
■ HONEYNET-in (95%)   ■ HONEYNET-out (95%)
               maximum       average       current        period          95th percentile
in-pass          1.33 Mb/s     0.15 Mb/s     0.01 Mb/s      1.56 GB i     897.17 kb/s
out-pass         1.22 Mb/s     0.14 Mb/s     0.01 Mb/s      1.51 GB o    -834.37 kb/s
in-block         1.72 kb/s     0.02 kb/s     0.00 kb/s    226.82 kB i
out-block        0.00  b/s     0.00  b/s     0.00  b/s      0.00  B o
                                                                  Feb 04 07:00:01 2015
```

Figure 40. Honeynet Traffic bits per second



```
■ HONEYNET-in-block   ■ HONEYNET-in-pass   □ HONEYNET-out-block   ■ HONEYNET-out-pass
               maximum        average        current         period
in-pass          1.13 k pps     0.12 k pps     0.01 k pps     10.67 M pkts
out-pass         1.05 k pps     0.13 k pps     0.02 k pps     11.28 M pkts
in-block         6.23   pps     0.07   pps     0.00   pps      6.36 k pkts
out-block        0.00   pps     0.00   pps     0.00   pps      0.00   pkts
                                                                  Feb 04 07:00:02 2015
```

Figure 41. Honeynet Traffic packets per second

Under normal situations the outgoing and incoming traffic are symmetrical. If a honeypot gets compromised and starts to scan or use a denial of service attack against an Internet host, the outgoing traffic would be more important than the incoming traffic. On the other hand, if a honeypot were the target of a scan or denial of service attack, the graph would show that more traffic is coming in.

f) Zabbix

Even though Zabbix monitors mostly the core network systems, some honeynet components are also monitored.

Host availability

It is possible to add software packages on PfSense. These packages add functionalities to the gateway. We added the Zabbix agent package allowing us to monitor Honeygate like any other systems. Zabbix will report if the gateway is unreachable. Zabbix will also report heavy processor usage, which is often linked to heavy network traffic.

The Low Interaction Honeypots and Honeymole server virtual machines are also running Zabbix agent and are monitored by Zabbix server.

Honeymole Tunnels State

On the Honeymole server virtual machines, Zabbix will collect the number of running processes for *honeymole*. When a Honeymole tunnel is up, two *honeymole* processes are present in the process list. When the tunnel is down, only one process is listed.

Zabbix will trigger an alert when the number of honeymole processes falls under two.

Network Traffic Collection

Zabbix is configured to collect the size of the current hour network traffic capture file on Spy. When the Honeynet is the target of a large attack or if one compromised host generates a lot of traffic that is not blocked by the gateway, the capture file is significantly larger than usual. The threshold for alert is currently set to 2 GB. When the capture file is bigger than 2 GB, Zabbix triggers an email alert.

B.5 Honeypot Experiments Hosting Environment

The experiment-hosting environment is an important element of the Honeynet. The objective is to provide the hardware resources (physical or virtual) to run all of the honeypot-based experiments. Different solutions have been implemented over the years to maximize the number of honeypots while minimizing maintenance tasks, being able to handle heavy network traffic and not overload the infrastructure.

B.5.1 Physical Hosts

Initially, most of the honeypot hosts were physical computers, but each new implementation of the Honeynet significantly expanded the number of IP addresses available for the honeypots. Resource-wise it was not possible to have one physical honeypot per IP address even if low interaction honeypots were used. Besides, in its current design, each subnet requires at least two machines: One honeymole server and one LIH machine. Just for the infrastructure, almost 15 computers would be necessary.

B.5.2 VMware Server 2

The honeypot virtualization within the Honeynet was first introduced with VMware Server 2.0 [VMS15], a virtualization software suite. VMware Server was installed on top of a Linux Operating System on fifteen physical machines. Depending on the nature of the virtual machines, each node could run up to five virtual machines. 10

years ago, virtualized servers were quite unusual and attacker could guess that they were honeypots, and try to exploit the physical host machine. To prevent the detection of the virtualization both VMware Server and the guest operating system were patched to hide the information that would reveal the nature of the targeted system.

Even though virtualization was allowing us to more than triple the number of honeypots, VMware Server was presenting several issues: the underlying Operating System required maintenance and was also consuming significant computer resources such as CPU time and memory. VMware Server management interface turned out to be very unstable. In addition, the management of the virtual machines was decentralized, there was no central console to create and control the different virtual hosts across the different physical machines.

B.5.3 VMware vSphere

VMware vSphere is the current Honeypot experiment hosting solution. Several nodes of the VMware cluster are dedicated to run the honeypot-based experiments virtual machines. As described in Section 4 of Appendix A, these hypervisors are connected to different networks.

Appendix C: Cybercrime Framework

C.1 Framework Design

In this appendix, we provide more details on the cybercrime framework design. As shown in Figure 42, each cybercrime experiment uses three different types of hosts: a network gateway, a collector host and a set of machines, called OVZ hosts. All of these hosts are virtual machines running on the lab VMware cluster. Because of the specific network configuration that has to be shared across three different hypervisors, the virtual machines for the cybercrime projects are hosted on three specific VMware servers.

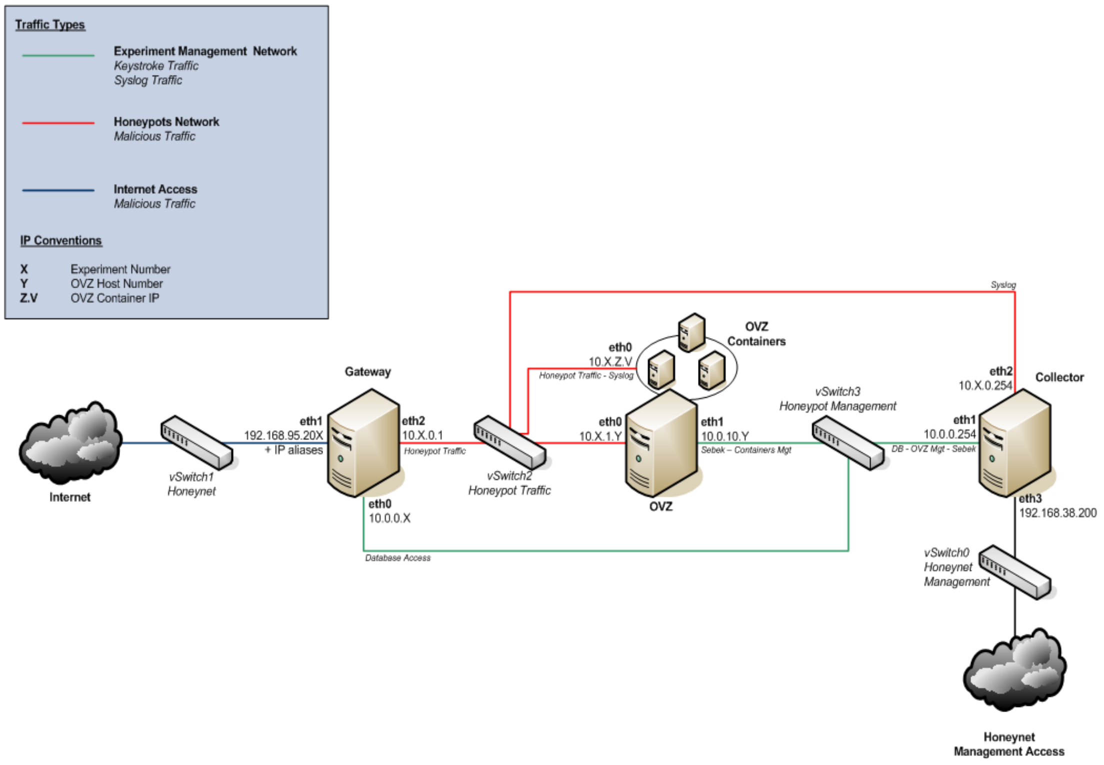


Figure 42. Experiment Design

The gateway is placed between the Internet and the other components of the framework, and is accepting SSH connections on port TCP/22. The OVZ hosts run

OpenVZ [OVZ15], a lightweight virtualization solution for Linux systems. OpenVZ allows us to run in parallel several honeypots per OVZ host. The collector is common to all the cybercrime experiment testbeds. This host aims at centralizing the collected data and the processing.

C.1.1 Collector and Management Host

The Collector host handles several functionalities within the cybercrime framework.

a) Data collection, processing and storage

The collector host is responsible for centralizing and organizing the data generated by the different monitoring tools used on the cybercrime framework. This host receives:

- The authentication events from all the honeypots
- The key logger traces from all the OVZ hosts
- The honeypot deployments from all the gateways

A Perl script processes the raw data and uploads them in a database on a daily basis. The data processing script rebuilds the attackers' sessions on the honeypots from the Syslog authentication logs. It also cleans the keystrokes sent by Sebek and matches them with the attackers' sessions.

b) Central Repository

All of the scripts running on the cybercrime framework are stored in a repository on the collector host. The scripts are automatically distributed to the different components of all of the experiments. The framework is flexible enough to allow experiment specific scripts. In addition to the scripts, the configuration files and the honeypots base images are all stored on the collector host and are all automatically distributed when a change occurs.

A central database also keeps track of the honeypots deployments and provides the gateways the IP to use and the NAT rules associating a honeypot container with a public address.

c) Monitoring

The collector host also monitors the health of the framework: it makes sure that each component is online and that all the monitoring tools are operating correctly. A daily email is generated and sent out to the cybercrime team. It contains information on the data collected as well as the health of each experiment.

The collector host also runs the website that provides access to the live data and status of the framework.

d) Maintenance

The framework provides a set of scripts that performs the daily maintenance operations of the different components. These operations include the removal of honeypot containers reaching the end of lifecycle and the automatic creation of new containers to allow further honeypot deployments.

C.1.2 Network Gateway

The gateway is attached to three different networks. One network interface is connected directly to the Internet and is configured with the 300 public IP addresses made available for the honeypots. The second interface is attached to a private network where all the honeypots containers are connected. The third network is used for management and data collection purposes. The gateway runs a Linux Ubuntu 12.04 operating system. The SSH server is a custom designed C program using the libssh [LIB15]. The fake SSH service returns a SSH successful authentication message to the attackers after a number of brute force attempts. This number is randomly selected between 100 and 200 at the very first login attempt by the attacker for a specific honeypot IP address. At this point each attacker is identified by his or her IP address.

When the expected number of attempts is reached, the C program calls an external script that 1) records the deployment in a database on the collector server, 2) configures the next available honeypot container with the login credentials that “successfully” broke into the system, 3) creates a Network Address Translation rule (NAT) to associate the public IP address targeted with the private IP of the newly configured honeypot container and, 4) attributes and applies one of the configuration types of the corresponding experiment. Once the execution of the script is complete, the SSH server establish a SSH connection with the newly configured honeypot and redirects it to the attacker. This overall sequence of brute-force entry and the resulting procedures discussed above can collectively be termed as a deployment wherein the intruder is successfully assigned to a honeypot with a randomly assigned type or treatment condition. The execution of the script and configuration of the honeypot container only takes a few seconds. In addition to running the fake SSH server and routing the Internet traffic to the honeypots, the gateway also limits the attacks targeting other Internet hosts to prevent their subsequent compromise. This is achieved by rate-limiting the outgoing traffic on specific protocols and ports. The firewall on the gateway limits:

- SSH scans and brute force attempts;
- UDP datagrams to prevent DoS attacks;
- Web and RDP scans.

C.1.3 OpenVZ Hosts

While the gateway manages the traffic and deployment of the honeypots, these honeypots need to be constructed and directly maintained by additional hosts. As we wanted to provide a UNIX environment to the attackers, OpenVZ was deployed on five CentOS 5.4 systems to perform the construction and maintenance tasks necessary for the honeypots to exist and function as intended. We used the stable release of the OpenVZ kernel 2.6.18-164.15.1.el5.028stab068.9 for the deployment.

As mentioned previously, OpenVZ is a lightweight virtualization tool for Linux-based operating systems. An operating system is fundamentally composed of two main elements: the kernel and the user space. The kernel controls the computer hardware and provides the applications executed by the users functions to interact with the hardware. The user space or application space is where all the users processes and software are executed. Each virtual machine on VMware or other full virtualization solutions will have virtual hardware (CPU, memory, network, hard drive…) and will run a complete operating system with kernel and user space. On the other hand OpenVZ, the host operating system will share its kernel and user space. Each container will be a sub-tree of the host operating system process list. Each container will have its own file system, but this file system will be a sub-directory on the host operating system file system.

The advantages of OpenVZ are threefold: first, OpenVZ allows to run several lightweight Linux OSs in parallel on a single OVZ host. After stress-testing the OVZ hosts, we determined that we could easily run up to 60 containers per OVZ host at the same time. With 5 OVZ hosts, this solution gave the ability to run up to 300 honeypots in parallel. Second, OpenVZ provides the tools to easily adjust the containers' configuration including the IP address and the credentials. Moreover, the OpenVZ virtual network interface does not permit the change of IP address within the honeypot container. As a consequence, attackers are not able to change the honeypot IP address following compromised entry. Lastly, attackers cannot interact with other honeypot containers nor with the host operating system. The attacker's actions are restricted to his or her assigned container. Even with root privileges, each container is isolated from the other devices within the design and from the host operating system running on each OVZ host, but we can access the containers (honeypots) file systems and process list.

a) OpenVZ Containers

Each container or honeypot is identified by its CTID (container ID). In our case the CTID is also the Honeypot ID. This identification number is unique across the 5 OVZ hosts of one experiment. A container can have the following states:

- Available, pre-deployed honeypot but not yet in use by an attacker
- In use, fully deployed and configured honeypot, reachable by an attacker via its corresponding public IP address.
- To revert, the honeypot has reached its end of life and awaits the recycling process

b) Template

OpenVZ uses a template to create a container. This template is a compressed archive file containing the containers file system. The template used for this project is built upon Fedora 12 operating system. Each container comes with two servers: a Web and Secure Shell Server.

C.1.4 Network Configuration

As shown in Figure 42, several networks have been created to support the cybercrime experiments proper operations and traffic containment:

- Honeynet: Shown in blue in Figure 42, this network is connected to the gateway to give access to the public IP addresses dedicated to the experiments.
- Honeynet Management Network: Shown in black in Figure 42, this network allows Cybercrime administrators to access the data and the different components of the framework from the users networks (core, lab and VPN networks).
- Experiment Management Network: Shown in green in Figure 42, this network is to separate the management traffic from the malicious once. This is to satisfy the network isolation rule. Besides this ensures that we have a constant access to the OVZ hosts even if an attacker uses all the bandwidth with his or her honeypot.
- Honeypot Network: Shown in red in Figure 42, this network is dedicated to the attackers' traffic (malicious) between the gateway host and the different honeypots or containers.

C.2 Datasets

In this section, we provide more details on the data collection and processing.

C.2.1 Datasets definition

In addition to the Network and the raw network traffic collected by the UMD Honeynet Data Collector, we also gather information relative to the attackers' Secure Shell sessions that include the commands typed by the attackers.

a) Sessions

Syslog [SYS15] is a computer log storage management program. This tool provides a standardized format for log messages generated by systems, software, tools and operating systems.

The Syslog configuration of the honeypots has been modified to send all the logs to the Collector host in addition to the existing local log files. Among all the events received by Syslog, we are particularly interested in the authentication messages generated by the Secure Shell server. These authentication messages provide enough information to produce records in the database pertaining to the beginning and end of the attackers' session, the origin IP address and the username used to authenticate on the system.

Syslog uses the Honeypot Network shown in red in Figure 42 to send all the log messages from the Honeypots to a remote Syslog server running on the Collector. Syslog is the only exception to the network isolation rule. In this case, the data collected from the Honeypots are transiting among malicious traffic. This solution is preferable than connecting the honeypots themselves to a management network.

The remote Syslog server on the Collector host is able to separate the Syslog events from one experiment to another. The Honeypots of each experiment will send their Syslog events to an IP address allocated to the experiment. Depending on that IP address, the Syslog server will be able identify the experiment and route the message appropriately.

On the Cybercrime Framework, the Collector host stores the Syslog events in a database. Depending on the source of the message (its destination IP on the Collector), the appropriate experiment database will be selected to store the message.

```
########################
## Example Experiment ##
########################

########################
# Source
########################
#
source s_src_example {
        udp(ip("10.2.0.254") port(514));
};

########################
# Destination
########################
#
destination d_mysql_example {
        sql(type(mysql)
        host("localhost") username("syslog") password("XXXXXXX")
        database("example")
        table("logs")
        columns("host",   "facility",   "priority",   "level",   "tag",   "datetime",
"program", "msg", "pid")
        values("$HOST_FROM",  "$FACILITY",  "$PRIORITY",  "$LEVEL",  "$TAG",  "$YEAR-
$MONTH-$DAY $HOUR:$MIN:$SEC", "$PROGRAM", "$MSG", "$PID")
        indexes("host", "facility", "priority", "datetime", "program"));
};

########################
# Log path
########################
#
log { source(s_src_example); filter(f_session); destination(d_mysql_example); };
```
Figure 43. Experiment Syslog Configuration

To do so, we have to define a log path in Syslog that includes a source, a filter and a destination. In the configuration file shown in Figure 43 is defined the source *s_src_example* that will select all the events received on the IP address 10.2.0.254, port UDP 514.

The destination rule *d_mysql_example* contains all the information required by Syslog to record events in a specific database.

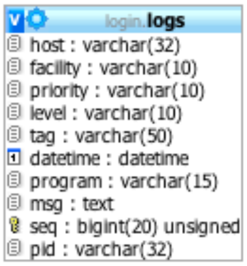
The log path at the end of the configuration file takes all the messages from *s_src_example*, filters them with the rules listed in *f_session* and then saves in the database specified in *d_mysql_example*.

As mentioned previously, Syslog is used to log various system events but we are only interested by successful Secure Shell logins and end of sessions. Filters can be applied on Syslog messages to discard the events we do not want to save in database. Figure 44 shows the set of filters defined in *f_session* used to only keep the following SSH authentication messages:

- `Accepted password for root from X.Y.Z.W port 6045 ssh2`
- `pam_unix(sshd:session): session closed for user root`

```
########################
# Filter
########################
#
filter f_session { match("session closed" value("MESSAGE")) or match("Accepted
password" value("MESSAGE")) or match("Accepted keyboard-interactive/pam"
value("MESSAGE")); };
```

Figure 44. Syslog Filter



```
login.logs
host : varchar(32)
facility : varchar(10)
priority : varchar(10)
level : varchar(10)
tag : varchar(50)
datetime : datetime
program : varchar(15)
msg : text
seq : bigint(20) unsigned
pid : varchar(32)
```

Figure 45. Syslog Table Structure

Each experiment has a *"logs"* table used by Syslog on the Collector to record the Secure Shell authentication. The table structure in Figure 45 allows Syslog to store the following information for each event:

- host: The Honeypot IP address
- facility: The software type that has generated the message
- priority: Combines the facility and the level of security
- level: The severity of the message from info to critical
- datetime: The date and time of the event
- program: Name of the program issuing the message
- msg: The event message (log)
- seq: The Syslog event number
- pid: Process ID of the program issuing the message

This table will be used later by the processing scripts to rebuild the attackers' Secure Shell sessions.

b) Keystrokes

To capture the attackers' keystrokes, we use a key logger from the Honeynet Project called Sebek [SBK15]. Sebek is a kernel module that extracts the keystrokes from the *read* system call. A modified version of the Sebek module has been deployed on each of the OVZ hosts. Since the Linux kernel is shared between all the containers, Sebek can record the keystrokes for all of them at the same time. The module has been modified to support OpenVZ and to add the Honeypot ID in the log. Each Cybercrime experiment OVZ host will send the data captured by the Sebek module to a specific port number on the Collector host through the Honeypot Management network.

On the Collector side, a Sebek listener called *sbk_extract* is started for each experiment. Each instance of *sbk_extract* listens on different UDP ports. The tool will decode the Sebek traffic received from the various OVZ hosts. The output generated by *sbk_extract* is not human readable as it is still in an encoded form. To decode and organize the Sebek output we use a Perl script called *sbk_ks_filter.pl*.

This intermediate script will read the sbk_extract ouput and:

- filter out the unnecessary logs created by the Sebek modules
- extract the attackers keystrokes
- rebuild from the keystrokes the attackers commands by using "[ENTER]" as delimiter

The *sbk_ks_filter.pl* script generates two files. The first one contains the commands issued by the attackers and the other contains the keystrokes. An example of keystrokes and its equivalent commands is shown respectively in Figure 46 and Figure 47. For both of these files, Sebek provides a timestamp, a *VEID* also known as Honeypot ID, the OVZ host IP address, the User ID, the process ID, the file descriptor and i-node of the standard output, and the command name. The keystroke output provides an additional timestamp in milliseconds. This timestamp allows us to see the time difference between each keystroke in milliseconds.

The Sebek module on the OVZ hosts already filters out anything that is not related to Secure Shell keystrokes, but additional filtering is required on the Collector as well. The *sbk_ks_filter.pl* script removes:

- The keystrokes and commands with a VEID of 0. VEID 0 designates the OVZ host itself.
- The keystrokes and commands with an i-node number greater than 10000. OpenVZ generates these "keystrokes" during the honeypot configuration phase.

```
[2014-10-05 10:28:38 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash MS:1412519318193697 ]#w

[2014-10-05 10:28:38 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash MS:1412519318389729 ]#[RETURN]

[2014-10-05 10:28:38 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash MS:1412519318655731 ]#c

[2014-10-05 10:28:38 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash MS:1412519318817674 ]#d

[2014-10-05 10:28:38 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash MS:1412519318859656 ]#

[2014-10-05 10:28:38 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash MS:1412519318925694 ]#/

[2014-10-05 10:28:39 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash MS:1412519319075653 ]#t

[2014-10-05 10:28:39 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash MS:1412519319187660 ]#m

[2014-10-05 10:28:39 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash MS:1412519319375652 ]#p

[2014-10-05 10:28:39 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash MS:1412519319452646 ]#[RETURN]
```

Figure 46. Sebek Filter Keystrokes Output

```
[2014-10-05 10:28:38 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash ]#w

[2014-10-05 10:28:39 Veid:1021 Host:10.0.10.21 UID:500 PID:12998 FD:0 INO:2
COM:bash ]#cd /tmp
```

Figure 47. Sebek Filter Commands Output

C.2.2 Databases

All data collected and processed by the Cybercrime Framework is stored in databases. Each database has its own dedicated database on the database server hosted on the Collector. Each experiment database schema has the same structure as the one depicted in Figure 48.

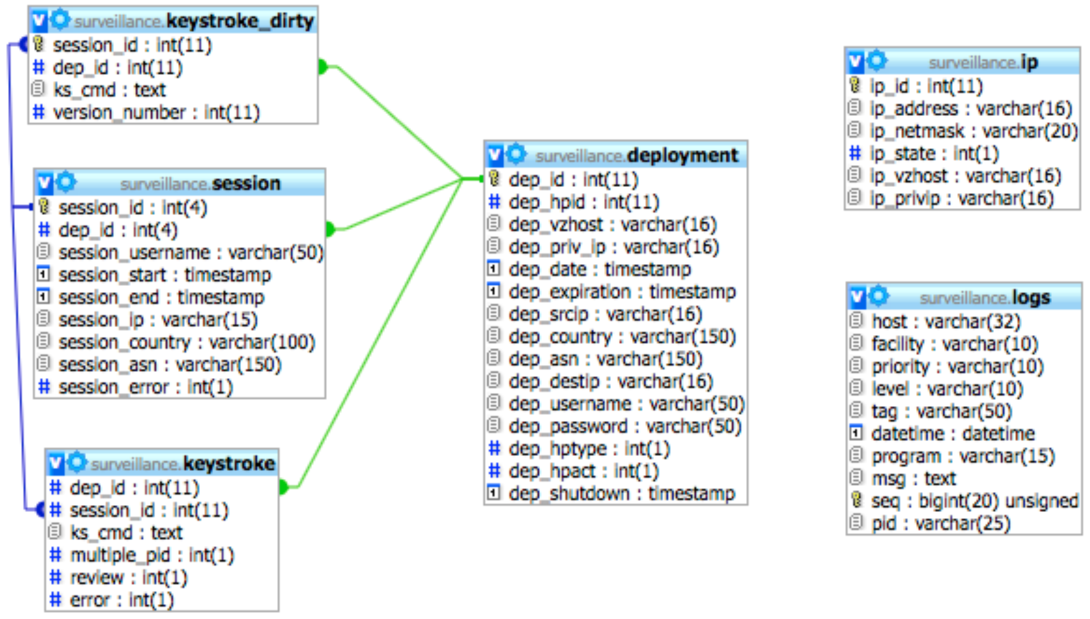


Figure 48. Cybercrime Database Schema

a) Deployment

This table keeps track of all the honeypots deployments. This table is only populated by the Gateway, and it inserts the following information for each successful deployment:

- The Honeypot ID
- The OpenVZ host running the honeypot
- The private IP address of the honeypot
- The deployment timestamp

- The attacker's IP address
- The country of the attacker based on the IP address
- The origin network number based on the attacker's IP address
- The targeted public address on the Honeynet
- The successfully "guessed" login and password
- The honeypot type or treatment number

This table is also updated by the recycling process to keep track of the honeypots that have reached their end of life:

- *dep_expiration* is the honeypot expiration timestamp which is the deployment timestamp plus thirty days.
- *dep_hpact* is a flag to indicate if the honeypot is still active or recycled
    - If *dep_hpact = 0*: The honeypot has been recycled
    - If *dep_hpact = 1*: The honeypot is still active
- *dep_shutdown* is the timestamp of the recycling

b) Session

All the successfully established attacker's Secure Shell sessions are rebuilt in this table from the Syslog authentication logs. From this table it is possible to find all of the SSH sessions for each honeypot along with the following information:

- The username used by the attacker to access the honeypot
- The login and logout times
- The IP of the attacker
- The country and network of origin of the attacker based on his or her IP address

The *session_error* field is used by the data consistency check.

The session and deployment tables are linked by the deployment ID. One deployment can have zero to several sessions. One session corresponds to only one deployment.

c) Keystroke

This table stores all of the commands typed by the attackers during a Secure Shell session. For example, the typed commands shown in Figure 47. One record in this table contains all of the commands along with the header added by Sebek for one session.

The keystroke table is linked with the session table and the deployment table via the session ID and the deployment ID. The relation with the deployment is not necessary since it can be also done through the session table but it allows a quick sort of the keystroke per deployment.

The *multiple_pid*, *review* and *error* fields are used by the data consistency check.

d) Keystroke_dirty

This table keeps track of the changes made by the keystroke cleanup process. Its structure is similar to the keystroke table.

e) Logs

This table is used by Syslog to store the authentication messages from the Secure Shell servers. Its structure and use is described in the previous section.

f) IP

This table is mostly used and updated by the Gateway host. It contains the list of the Honeynet public IP addresses dedicated to a Cybercrime experiment. Each IP address is associated with:

- The netmask used to configure the network interface on gateway

- The IP address state
    - If *ip_state = -1*: This IP is blocked, a firewall rule is added to block the traffic to and from it.
    - If *ip_state = 0*: This IP is available for brute-force and is not associated with a honeypot
    - If *ip_state = 1*: This IP is in use and associated with a Honeypot. In this case the Gateway creates the Network Address Translation rules to forward the traffic for that IP to the correct honeypot identified by its private IP (*ip_privip*).
- The IP of the OVZ host is present for information only. It is a quick way to find the OVZ host running the honeypot in case of an issue.

At boot, the Gateway associates all of the public IP addresses present in the database with the network interface connected to the Honeynet network and uses the state of each IP to initialize the firewall blocks and traffic forwarding for the active honeypots.

During the deployment phase in addition to creating a deployment record in the database, the Gateway updates the corresponding IP record with the new state, the honeypot private IP address and the OVZ host IP address.

During the recycling process, the Collector also updates the IP state and remove the OVZ host and honeypot private IPs.

C.3 Data Processing Script (Session, KS, KS cleanup)

The data processing script is executed at midnight every day, and it performs the following actions:

1) Sebek Filter output files rotation

2) Re-build in database the session information from the Syslog table

3) Process the command and keystroke files of the day and associate them with the sessions

4) Clean the keystrokes

C.3.1 Sebek Rotation

The `sbk_ks_filter.pl` script creates two files called *current.dump* and *current.ks.dump* and both are located in `/cybercrime/data/<experiment>/sebek/`. They respectively contain the commands and the keystrokes of the attackers. The processing script first suspends the keystroke collection for all the experiment and then renames, relocates and compresses the *current* files using the following convention:

```
/cybercrime/data/<experiment>/sebek/YYYY/MM/YYYY-MM-DD_<experiment>.cmd.gz
/cybercrime/data/<experiment>/sebek/YYYY/MM/YYYY-MM-DD_<experiment>.ks.gz
```

Then Sebek is restarted with new `current.dump` and `current.ks.dump` files.

C.3.2 Sessions

For each experiment, the processing script looks for the Syslog logs entries similar to the one shown in Figure 49. This event indicates that an attacker successfully authenticated on a honeypot using Secure Shell. The search is limited to the past seven days by using the log timestamps.

```
Accepted password for root from X.Y.Z.W port 6045 ssh2
```

Figure 49. Secure Shell Accepted Password Message

For each authentication success message found, the processing script will extract:

- The timestamp which is the session start time

- The username
- The IP address
- But also the process ID stored in database

And will look for the session closing message recorded after the session start. The *session closed* message like the one shown in Figure 50 indicates that the attacker logged out of the honeypot. Both *accepted password* and *session closed* events will present the same process ID. This allows us to identify the right *session closed* message. Once found, the record provides the timestamp of the session end.

```
pam_unix(sshd:session): session closed for user root
```

Figure 50. Secure Shell Session Closed Message

A database query quickly identifies the honeypot where the session occurred and returns the deployment ID necessary to create a session in the database. Once the session is rebuilt and all the information gathered, a record is inserted in the session database. If a session with the same characteristics already exists, the record is not created in database to avoid duplicates. As mentioned previously, the script looks at all the sessions from the past seven days and thus is likely to find sessions that have already been inserted in database. But this is necessary to make sure we record the sessions that are lasting several days.

C.3.3 Keystroke and Session Association

With the information provided by Sebek, the only way to match the commands logged by Sebek with a session is to compare the commands timestamps with the session start and end timestamps.

The script reads line by line the previously rotated Sebek files and sorts the logs per VEID also known as honeypot ID. It creates a file for each honeypot ID:

`/cybercrime/data/<experiment>/keystrokes/<hpid>/<honeypot_id>`. This file contains the keystrokes of the honeypot. If the file already exists, the new keystrokes are appended at the end.

The processing script then queries the database to obtain the sessions of the honeypots with keystrokes from the last seven days. For each session, the script identifies all the commands (from the *honeypot_id* file) matching the start and end timestamps.

At this point, if a previous keystroke record exists in the database for that session, the record is deleted and re-created.

For the same reason as the session processing, the script processes the commands for the session of the last seven days. The Sebek log lines that do not fit any session are ignored. It is also worth noting that because of the volume of data they represents, the keystrokes files are not imported in the database.

C.3.4 Keystroke Cleanup

The Session-Keystroke association matching process presents some issues. The timestamps are used to match the keystrokes with a session. It works well when an attacker opens a session, types some commands and closes the session. However if the attacker of one honeypot opens two or more sessions in parallel, the processing script will not be able to distinguish which of the sessions the keystroke logs belong to.

Sebek log header provides a VEID that identifies the honeypot. For each session the process ID (PID) is unique and does not change during the session. So the PID can be used to identify the different sessions with the keystrokes logs. However, when an attacker invokes a program that stays in the foreground and interprets the keyboard entries (such as a text editor) the PID reported by Sebek changes but it is still the same session. These regular PID changes can be identified using the COM field. The COM field reflects the process name that handled the logged keyboard entries. When the PID changes within the same session, the COM field is likely to change as well and match the previously typed command.

For example in Table 38, the attacker was initially on the shell bash and ran the command ftp, a file transfer tool. Ftp does not return to the shell immediately, it offers a prompt and the user can enter ftp commands to download or upload files. Ftp now handles the standard input. As a consequence the PID and the COM fields changed. Once the attacker exited the ftp program, the PID and COM fields went back to their initial value.

Table 38. Keystrokes Commands with “normal” multiple PIDs

Sess. ID	Dep. ID	Keystrokes Commands

696	211	`[2013-10-22 17:37:26 Veid:1788 … PID:25014 … COM:bash ]#ls -a`
		`[2013-10-22 17:37:31 Veid:1788 … PID:25014 … COM:bash ]#ftp`
		`[2013-10-22 17:42:53 Veid:1788 … PID:25033 … COM:ftp ]#bye`
		`[2013-10-22 17:43:26 Veid:1788 … PID:25014 … COM:bash ]#ls -a`

Some attackers sometimes launch a second shell. The PID changes but the COM field does not, however the attacker has to enter “bash” to start the new shell.

Because of this imperfect relation between keystroke and session logs, the database can have keystroke session duplicates.

a) Full duplicates

An example of duplicated keystroke session is shown in Table 39. The commands are the same across the sessions. It usually happens when the sessions have about the same duration and overlap totally. In that case the keystroke record for session ID 18999 will be deleted.

Table 39. Keystroke Session Duplicate

Sess. ID	Dep. ID	Commands

18965	2558	`[2014-05-15 04:50:58 Veid:3549 … PID:6159 … COM:bash ]\#`
		`[2014-05-15 04:52:00 Veid:3549 … PID:6159 … COM:bash ]\#cd .t`
		`[2014-05-15 04:52:00 Veid:3549 … PID:6159 … COM:bash ]\#chmod +x*`
		`[2014-05-15 04:52:06 Veid:3549 … PID:6159 … COM:bash ]\#./inst`

18999	2558	`[2014-05-15 04:50:58 Veid:3549 … PID:6159 … COM:bash ]\#`
		`[2014-05-15 04:52:00 Veid:3549 … PID:6159 … COM:bash ]\#cd .t`
		`[2014-05-15 04:52:00 Veid:3549 … PID:6159 … COM:bash ]\#chmod +x*`
		`[2014-05-15 04:52:06 Veid:3549 … PID:6159 … COM:bash ]\#./inst`

b) Partial duplicate

An example of partial duplicate is shown Table 40. When the session duration is significantly different or when the sessions do not overlap by much, the commands are partially duplicated. In this case we remove the keystroke record for session 19071. It is clear that command has been duplicated from session 19070 and are incomplete.

Table 40. Keystroke Session Partial Duplicate

Sess. ID	Dep. ID	Keystrokes Commands

19070	2553	`[2014-05-15 04:56:58 Veid:3541 … PID:28750 … COM:bash ]#w`
		`[2014-05-15 04:57:02 Veid:3541 … PID:28750 … COM:bash ]#passwd`
		`[2014-05-15 04:57:39 Veid:3541 … PID:67569 … COM:bash ]#perl udp.pl`

19071	2553	`[2014-05-15 04:57:39 Veid:3541 … PID:67569 … COM:bash ]#perl udp.pl`

c) Multiple and mixed PIDs across several sessions

It happens that some attackers open several sessions at the same time with more or less overlap and use them actively to issue commands on the system. The keystrokes get duplicated on the different sessions and depending on the overlaps, the duplicates are usually partial. These keystroke records will naturally show different PIDs confirming the sessions overlap. After verifying that the PID change does not account for a program execution, the processing script will try to separate the keystrokes for each concurrent session. One session will have all the keystrokes for a specific PID and the other sessions' partial duplicates. The partial duplicates in each of the parallel sessions will be removed. When the processing script cannot determine properly the each keystroke sessions, the records get flagged in the database (*multiple_pid* and *error* fields) and the error must be fixed manually.

The *review* field in the keystroke table indicates the processing script has reviewed and if necessary corrected the keystroke record.

C.3.5 Data Consistency Check

Because of the complexity of the programs operating the honeypots, collecting the data, building the datasets and processing them, inconsistencies across the different tables can appear at times. The datasets built by the framework must be trusted as they are being in various research projects. Any inconsistency or error should be spotted quickly, the root cause identified and the problem fixed. Most of the issues raised by the consistency check program are reported to the administrator for manual inspection. The fields *error* in the keystroke table and *session_error* in the session table are both used to flag records with problems.

A consistency check is performed every morning. A report of the issues is sent via email to the administrator. The email is sent only if at least one issue has been detected. Once a session or keystroke record has been flagged it will not be reported again by the following execution of the consistency check tool.

a) IDs Consistency

This step consists in checking that the sessions are still associated with a deployment and that the keystrokes are associated with a session. For the session table, the dep_id of each record is checked against the deployment table to detect orphan sessions. The same for the keystroke table and the session records. Orphan records are removed from the database since they cannot be associated to a deployment or a session.

b) Null sessions with Keystrokes

Null sessions are attackers Secure Shell sessions with a duration of 0 seconds. The attacker logs in and logs out immediately. These sessions are most likely automated; the attacker may want to check that he or she still has access to the compromised system. A scripted bot could also execute a few commands on the honeypot to launch an attack program or a network scan or perform any action on the system in less than a second.

In any cases, a null session cannot have an extended keystroke session. The consistency check program identifies these null sessions with keystrokes, flag them and reports them to the administrator.

c) Keystrokes and Session match

At this step, the tool checks that each command in all the keystroke records is a perfect fit for the associated session. The tool compares the command timestamp with the session start and end timestamps. Here again, the keystroke records with issues are flagged and reported.

d) Session Start and End

This is a simple check, it detects for each session when, according to the database, a session start timestamp is posterior to a session end timestamp.

C.3.6 Data Extraction

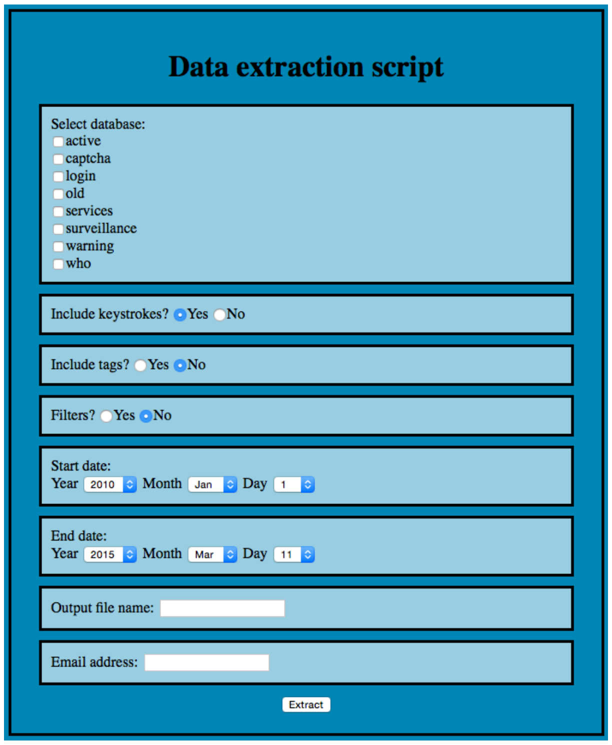
One of the objectives of the cybercrime initiative is to involve different domains of expertise not necessarily linked to engineering or computer science. All of the datasets should be easily accessible by any authorized researcher. The SQL language used to query the cybercrime databases can quickly become a barrier to data access. To solve that issue, some ruby scripts and a web page have been developed to provide a way for researchers to extract the data into text files compatible with most spreadsheet software.

a) Frontend

The frontend application is a web page hosted on the processing server Poseidon. This web page offers the possibility to submit a data extraction job. These extraction jobs are customizable via an online form shown in Figure 51:

- The user can extract the data from one or more experiments (or databases)
- The extraction can include the attackers' commands
- The result files can contain the commands categories (tags)
- The filters allow the user to select specific deployment IDs or session IDs. When yes is selected, the two additional fields shown in Figure 52 appear.
- The user can select a start and end date for the deployments.

- The output field allows the user to customize the output file names.
- The email address is used by the program to notify the user when an extraction job has started and ended. The email provides also a link to download a ZIP archive containing the result files.



Data extraction script

Select database:
- ☐ active
- ☐ captcha
- ☐ login
- ☐ old
- ☐ services
- ☐ surveillance
- ☐ warning
- ☐ who

Include keystrokes? [✓ Yes] No

Include tags? Yes [✓ No]

Filters? Yes [✓ No]

Start date:
Year 2010 Month Jan Day 1

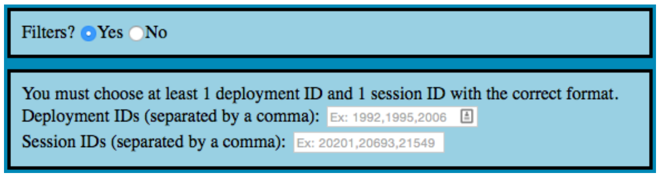
End date:
Year 2015 Month Mar Day 11

Output file name:

Email address:

Figure 51. Data Extraction Online Form

After filling out the online form, the extraction job will be inserted in database. The database keeps track of all the extraction requests with their status (not yet processed, processing and processed).



Filters? [✓ Yes] No

You must choose at least 1 deployment ID and 1 session ID with the correct format.
Deployment IDs (separated by a comma): Ex: 1992,1995,2006
Session IDs (separated by a comma): Ex: 20201,20693,21549

Figure 52. Session and Deployment IDs filters

b) Backend

Two different scripts are involved in the backend. The first one checks the database for new extraction job every five minutes. Once a new job is submitted, this script launches the extraction process with all the options specified by the user. As the extraction process can be resource and time consuming, only one job is executed at a time.

To facilitate the data analysis of the Cybercrime experiments, a modular ruby script has been written to extract the data from the Collector database and aggregate the deployment, session and keystroke information. This extraction script generates four types of output:

- Session Aggregate: This file gathers general information on each deployment as well as some specific characteristics and metrics pertaining to the first nine sessions (if present).
- Keystroke Panel: A list of commonly used and well-known UNIX commands has been built. The panel file will contain the number of times these well-known commands have been executed for each session.
- Union Session Aggregate: This file gathers general information on each deployment as well as characteristics and metrics on the first nine union sessions.
- Union Session Panel: This file gathers general information on each deployment as well as characteristics and metrics for all the union sessions.

Union sessions are cases when an attacker opens multiple Secure Shell sessions at once. The Union session collapse and combines the characteristics of all these sessions into one.

When the extraction script starts processing a job, the user is notified via email and the database maintaining the job queue is updated with the “processing” status. A job in a processing state will prevent the execution of other jobs at the same time.

Once the extraction process is done, the user receives an email with a link to download the ZIP archive containing the output files. The ZIP archive will be available for download during seven days.

C.4 Honeypot Life Cycle

The honeypot lifecycle is determined by the experimental design and also by the technical implementation. As depicted in Figure 53, three phases characterize the attackers actions against the Honeypots. The first one is the scanning phase where the attackers look for SSH servers on the Internet. When the IP address corresponds to a non-deployed honeypot, the experiment gateway receives these scans and replies to them to show that a SSH service is running.

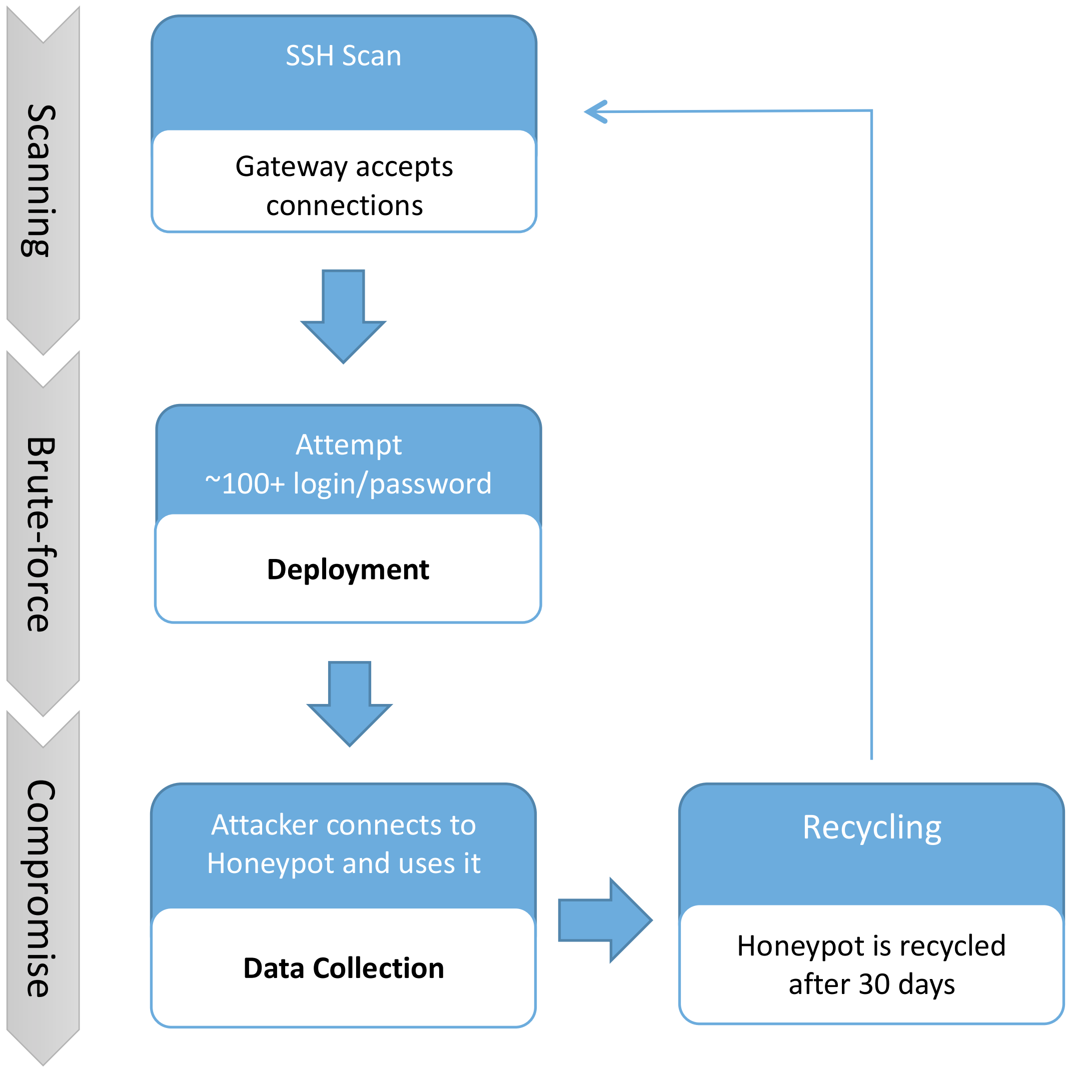

Scanning
Brute-force
Compromise
SSH Scan
Gateway accepts connections
Attempt ~100+ login/password
Deployment
Attacker connects to Honeypot and uses it
Data Collection
Recycling
Honeypot is recycled after 30 days

Figure 53. Honeypot Lifecycle

C.4.1 Brute-force Attack

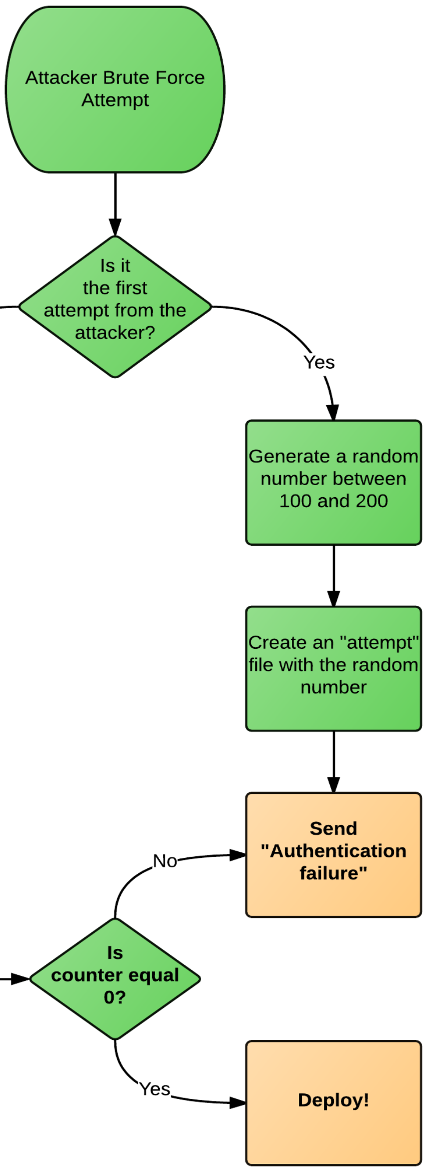

Attacker Brute Force Attempt
Is it the first attempt from the attacker?
No
Yes
Read *Counter* in the attacker "attempt" file
Generate a random number between 100 and 200
Create an "attempt" file with the random number
Send "Authentication failure"
No
Decrease *Counter* by 1
Is counter equal 0?
Yes
Deploy!

Figure 54. Brute Force Attack Attempt

The second phase is the brute-force attack where the attackers try to guess the login and password. The gateway handles these attacks. As shown in Figure 54 once the threshold of guesses is reached, the honeypot is deployed. Once deployed, the attacker can establish a SSH session with the Honeypot using the “guessed” credentials. This is the compromise phase during which we collect most of the data related to one honeypot.

After 30 days, the honeypot is shut down: the public IP address is blocked and the corresponding container is marked for recycling.

C.4.2 Honeypot deployment

The deployment process, shown in Figure 55, handles 1) the network configuration (traffic redirection of a public IP address to a Honeypot private IP address using Network Address Translation (NAT)), 2) the honeypot configuration and 3) the logging to database.

The deployment is initiated by the Gateway following a successful login and password “guess”. The gateway determines the Honeypot ID of the next available Honeypot as well as the honeypot type randomly. It will then contact the OVZ host hosting the container with the corresponding Honeypot ID. The container will be configured to reflect the treatment for the honeypot type given by the gateway.

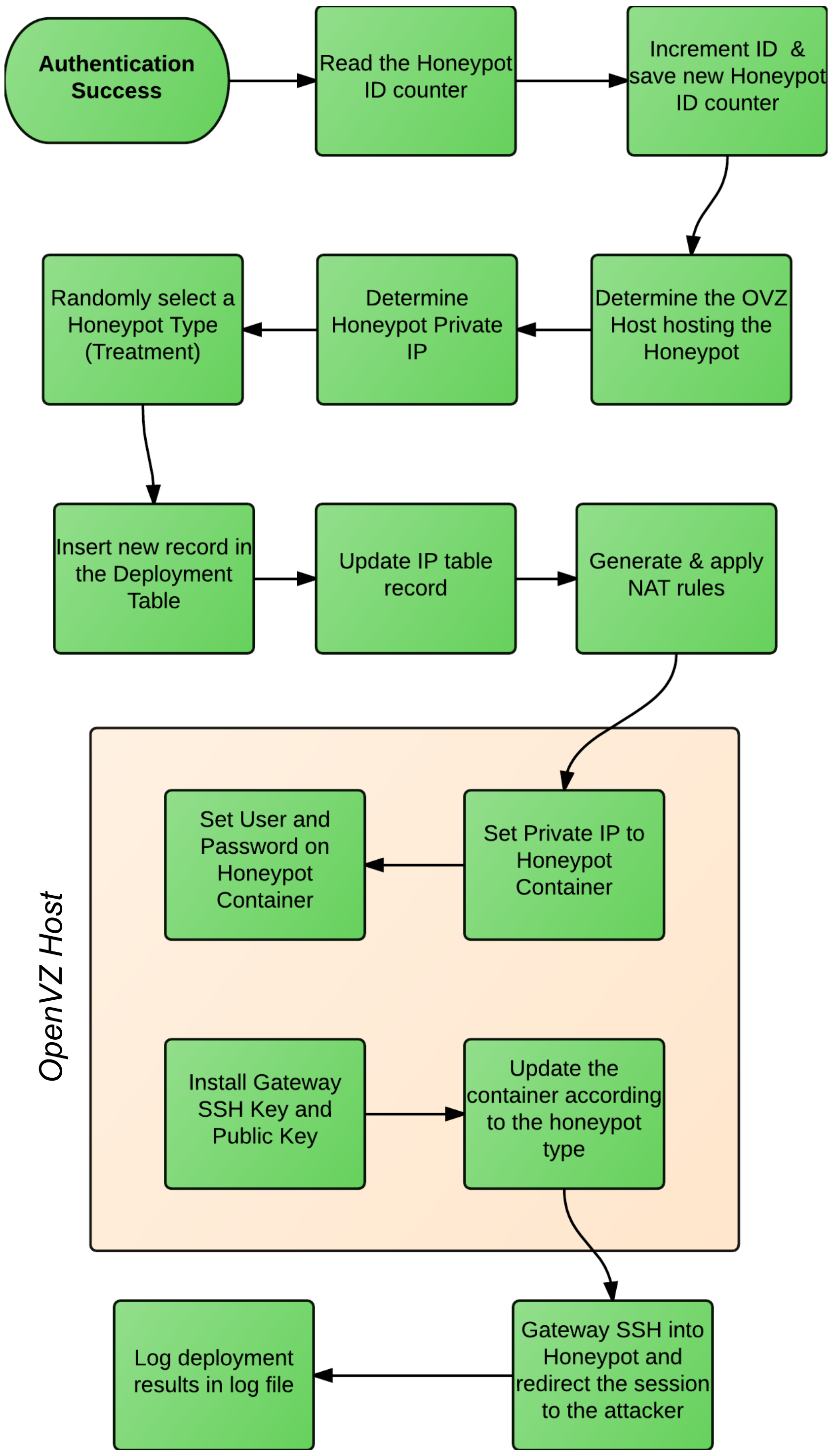

Authentication Success
Read the Honeypot ID counter
Increment ID & save new Honeypot ID counter
Determine the OVZ Host hosting the Honeypot
Determine Honeypot Private IP
Randomly select a Honeypot Type (Treatment)
Insert new record in the Deployment Table
Update IP table record
Generate & apply NAT rules
OpenVZ Host
Set Private IP to Honeypot Container
Set User and Password on Honeypot Container
Install Gateway SSH Key and Public Key
Update the container according to the honeypot type
Gateway SSH into Honeypot and redirect the session to the attacker
Log deployment results in log file

Figure 55. Honeypot Deployment Process

C.4.3 Honeypot Recycling

a) OVZ Containers Recycling

Due to technical issues related to Sebek, the honeypot OVZ containers cannot be recycled right after the 30-day compromise phase. The OVZ host operating system crashes when the Sebek kernel module is loaded and a honeypot container is being stopped for destruction. Once the operating system has crashed, all of the containers stop working and a reboot is necessary. This interrupts the normal operations of the experiment. To prevent this interruption, we wait until the 60 honeypot containers are ready for recycling. The process is divided into two steps. As shown in Figure 56, the recycling script determines whether all of the 60 honeypots on a given OVZ host have reached their end of life or not. If yes, a recycling flag file is created and the OVZ host is rebooted. This part of process is executed every nights.

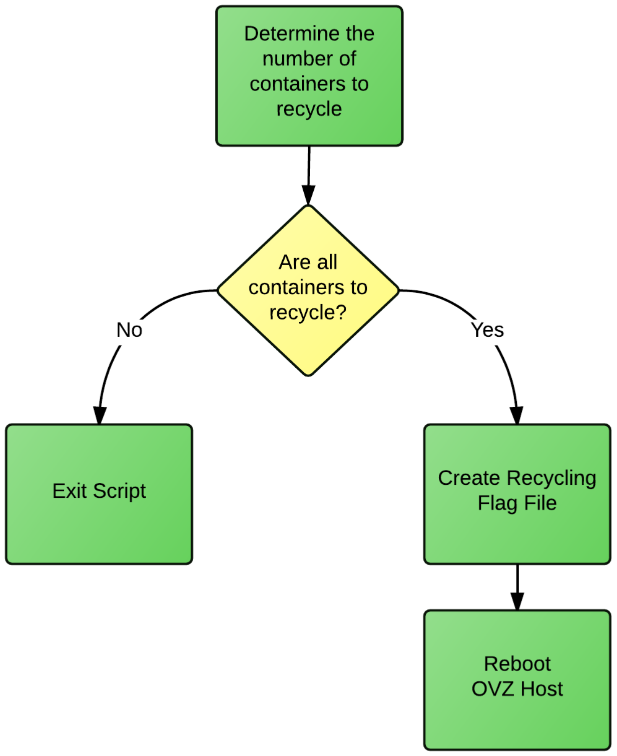

Determine the number of containers to recycle
Are all containers to recycle?
No
Yes
Exit Script
Create Recycling Flag File
Reboot OVZ Host

Figure 56. OVZ Containers Recycling Process Part 1

The second part of the process takes place at the OVZ host's OS (Figure 57). After the reboot, the recycling script detects the flag file. Containers are backed up on the Collector Host and then destroyed. Based on the number of containers per OVZ host (typically sixty) and the number of OVZ hosts (five), new honeypot IDs are determined and a new set of containers is created. The system then continues normal boot operations: it starts the honeypot containers and loads the Sebek module. The framework will detect the creation of these containers and will release the block of IP addresses chosen randomly. At the end of the recycling process, the Sebek module is loaded in memory.

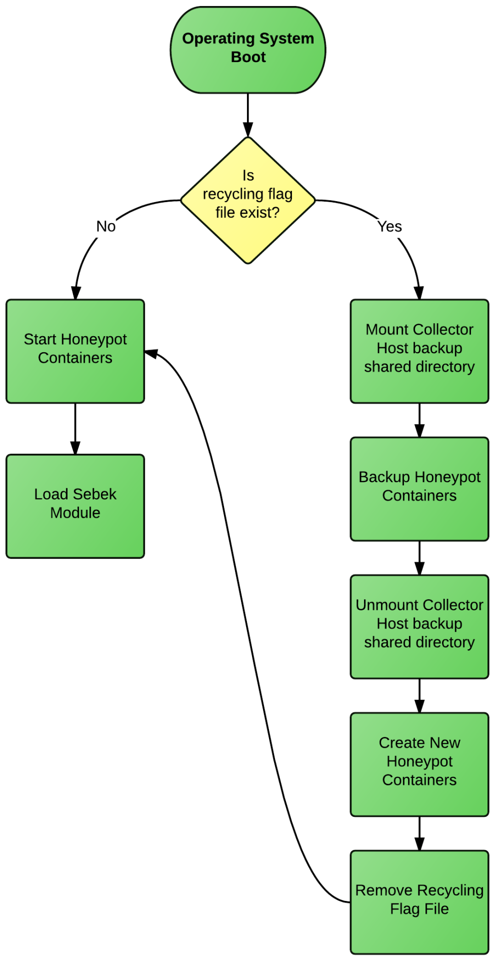

Operating System Boot
Is recycling flag file exist?
No
Yes
Start Honeypot Containers
Load Sebek Module
Mount Collector Host backup shared directory
Backup Honeypot Containers
Unmount Collector Host backup shared directory
Create New Honeypot Containers
Remove Recycling Flag File

Figure 57. OVZ Containers Recycling Process Part 2

b) Honeypot Recycling

The recycling process is executed every night. The Collector host looks for honeypot deployments that have passed their expiration date. For each expired honeypot, the Collector host:

- connects to the corresponding OVZ host and changes the container ID name from *localhost* (default) to *to_recycle*,
- connects to the Gateway to install the firewall rule blocking the public IP address
- updates the IP table in database with the new state
- updates the deployment table to mark the honeypot as inactive and add the shutdown timestamp.

c) IP Unblock

The recycling process blocks the public IP addresses of the honeypots being deactivated. When an OVZ host is entirely recycled, meaning old honeypots destroyed and new ones created, it is necessary to unblock a number of public IP addresses equal to the number of freshly created honeypots.

This process is executed every day in the morning after the recycling process on the OVZ hosts. It identifies the number of IP addresses available to brute-force attacks and checks on each OVZ hosts how many containers are not in use and not marked for recycling:

- If the number of available honeypots is greater than the number of available IP addresses, the script determines the number of IP to release and randomly selects IP addresses from the database. The firewall block is then removed on the Gateway and IP address state is updated.

- If the number is equal, nothing is done
- If the number of available IP is greater than the number of honeypots, in this case the script will block the surplus of IP addresses.

C.5 *Multi-Experiment Architecture*

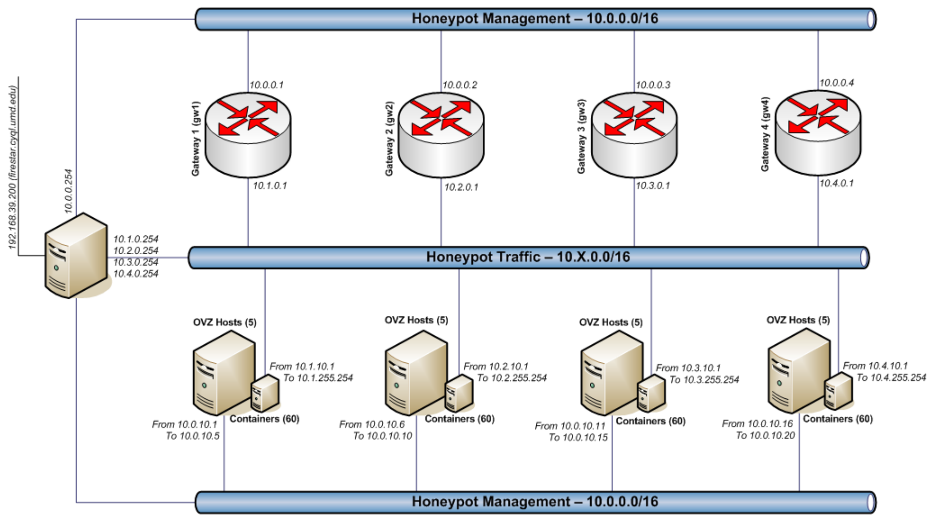

Honeypot Management – 10.0.0.0/16
192.168.39.200 (firestar.cyql.umd.edu)
10.0.0.254
10.1.0.254
10.2.0.254
10.3.0.254
10.4.0.254
10.0.0.1
Gateway 1 (gw1)
10.1.0.1
10.0.0.2
Gateway 2 (gw2)
10.2.0.1
10.0.0.3
Gateway 3 (gw3)
10.3.0.1
10.0.0.4
Gateway 4 (gw4)
10.4.0.1
Honeypot Traffic – 10.X.0.0/16
OVZ Hosts (5)
From 10.1.10.1 To 10.1.255.254
Containers (60)
From 10.0.10.1 To 10.0.10.5
OVZ Hosts (5)
From 10.2.10.1 To 10.2.255.254
Containers (60)
From 10.0.10.6 To 10.0.10.10
OVZ Hosts (5)
From 10.3.10.1 To 10.3.255.254
Containers (60)
From 10.0.10.11 To 10.0.10.15
OVZ Hosts (5)
From 10.4.10.1 To 10.4.255.254
Containers (60)
From 10.0.10.16 To 10.0.10.20
Honeypot Management – 10.0.0.0/16

Figure 58. Cybercrime Framework

As shown in Figure 58, the Cybercrime framework has been designed to host several experiments. The framework provides a common Collector host for all the experiments, and a template of OVZ hosts and Gateway. OVZ hosts, OVZ containers (honeypots) templates and Gateway can be customized depending on the experiment design. The experiments share the same physical networks, but different IP address ranges are used to separate them, in particular on the Honeypot Traffic network. As a consequence, the Collector host must have several IP addresses on the Honeypot Network in order to be reachable from the honeypots of each experiment to receive the Syslog messages.

Bibliography

[ABU06] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," SIGCOMM Conference on Internet Measurement, 2006.

[ALA06] E. Alata, V. Nicomette, M. Kaâniche, M. Dacier, and M. Herrb, "Lessons learned from the deployment of a high-interaction honeypot," EDCC, 2006

[ALM08] S. Almotairi, A. Clark, G. Mohay, and J. Zimmermann, "Characterization of attackers' activities in honeypot traffic using principal component analysis." , Network and Parallel Computing, 2008

[ALS07] M. Alsaleh, M. Mannan, P.C. van Oorschot, "Revisiting Defenses against Large-Scale Online Password Guessing Attacks," Dependable and Secure Computing, 2012

[AMU15] Amun: Python Honeypot, http://amunhoney.sourceforge.net, March 2015

[AND05] M. Andreolini, A. Bulgarelli, M. Colajanni, F. Mazzoni, "HoneySpam: honeypots fighting spam at the source," Steps to Reducing Unwanted Traffic on the Internet Workshop, 2005

[ANU15] Anubis: Malware Analysis for Unknown Binaries, https://anubis.iseclab.org, April 2015

[ARC09] J. D'Arcy, J., and A. Hovay. "Does one size fit all? Examining the differential effects of IS security countermeasures", Journal of Business Ethics, 89(1), p. 59-71, 2009

[BAC08] P. Bacher, T. Holz, M. Kotter, and G. Wicherski, "Know Your Enemy: Tracking Botnets (using honeynets to learn more about bots)," Technical report, The Honeynet Project, august 2008.

[BAC15] BackupPC: Open Source Backup to disk, http://backuppc.sourceforge.net, March 2015

[BAI05] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson, "The internet motion sensor: A distributed blackhole monitoring system," Network and Distributed System Security Symposium NDSS 05, 2005

[BAR96] P. Barclay, "Preventing auto theft in suburban Vancouver commuter lots: Effects of a bike patrol. Preventing mass transit crime", Crime prevention studies 6, 1996

[BAY06] U. Bayer, C. Kruegel, and E. Kirda, “TTAnalyze: A Tool for Analyzing Malware,” 15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference, 2006

[BER09] R. Berthier, J. Arjona, and M. Cukier, “Analyzing the Process of Installing Rogue Software,” in Proc. International Conference on Dependable Systems and Networks (DSN-2009), 2009

[BLA01] S. Blank, “Can information warfare be deterred? In Information Age Anthology”, Volume III: The Information Age Military, eds. David S. Alberts and Daniel S. Papp. Washington, DC: Command and Control Research Program, 2001.

[CAS15] Central Authentication Service, https://www.apereo.org/cas, April 2015

[CHA04] G. Chamales, “The honeywall cd-rom,” Security Privacy, 2004.

[CHA11] Chang, Ee-Chien, et al. "Enhancing host security using external environment sensors." International Journal of Information Security 10.5, 2011

[CHA15] Chaosreader, http://chaosreader.sourceforge.net, March 2015

[CHE95] Y.-W. Cheung, L. Kon, “Lag Order and Critical Values of the Augmented Dickey-Fuller Test,” Journal of Business & Economic Statistics, American Statistical Association, 1995

[CLA97] R. Clarke, “Situational crime prevention”, Monsey, NY: Criminal Justice Press, 1997

[COL03] R. Colin, "Operating systems incorporating UNIX and Windows”, 1997

[CON15] Conpot ICS/SCADA Honeypot, http://conpot.org, March 2015

[CUR04] K. Curran, C. Morrissey, C. Fagan, C. Murphy, B. O’Donnell, G. Fitzpatrick, and S. Condit, “A year in the life of the irish honeynet: attacked, probed and bruised but still fighting,” Information Knowledge System Management, 2004.

[CUS93] M. Cusson, "Situational deterrence: Fear during the criminal event”, Crime prevention studies 1, 1993

[CWS14] Malware Analysis CWSandbox: http://www.mwanalysis.org, March 2014

[DIO15] Dionaea – catches bugs, http://dionaea.carnivore.it, March 2015

[DHA06] R. Dhamija, J. D. Tygar, and M. Hearst, “Why phishing works,” In Proceedings of the SIGCHI conference on Human Factors in computing systems, CHI ’06, 2006.

[DHI07] C. Dhinakaran, Lee Jae Kwang, D. Nagamalai, “An Empirical Study of Spam and Spam Vulnerable email Accounts,” Future Generation Communication and Networking (FGCN 2007), 2007

[DIC79] D. Dickey, W. Fuller, “Distribution of the Estimators for Autoregressive Time Series With a Unit Root,” Journal of the American Statistical Association, American Statistical Association, 1979

[ELD01] J. Elder, and P. E. Kennedy, “Testing for unit roots: what should students be taught?.”, The Journal of Economic Education 32.2 (2001)

[FIL15] The Fine Free File Command, http://www.darwinsys.com/file/, March 2015.

[FPR15] NetFlow Probes: fprobe and fprobe-ulog, http://fprobe.sourceforge.net, March 2015.

[FRA07] J. Franklin, A. Perrig, V. Paxson, S. Savage, “An inquiry into the nature and causes of the wealth of internet miscreants”, Computer and communications security, 2007

[FUR02] S. Furnell, “Cybercrime: Vandalizing the Information Society,” Boston, MA: Addison-Wesley, 2002

[GAD08] J. Gadge, A.A. Patil, “Port scan detection,” ICON, 2008

[GAS13a] M. Cukier, I. Gashi, B. Sobesto and V. Stankovic. "Does Malware Detection Improve with Diverse Antivirus Products? An Empirical Study". in 32nd International Conference on Computer Safety, Reliability and Security (SAFECOMP), 2013.

[GAS13b] I. Gashi, B. Sobesto, S. Mason, V. Stankovic and M. Cukier, “A Study of the Relationship between Antivirus Regressions and Label Changes”, International Symposium on Software Reliability Engineering (ISSRE), 2013.

[GEE75] M. R. Geerken,W. R. Gove, “Deterrence: Some theoretical considerations”, Law and Society Review, 1975

[GHA14] R. Dhamija, J. D. Tygar, and M. Hearst, “Why phishing works”, SIGCHI Conference on Human Factors in Computing Systems, 2006.

[GIB75] J. P. Gibbs, Crime, punishment, and deterrence. New York: Elsevier, 1975

[GIT15] Git, http://git-scm.com, March 2015

[GLA15] Glastopf Honeypot Project, http://glastopf.org, March 2015.

[GOE07] J. Goebel, T. Holz, “Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation”, Hot Topics in Understanding Botnets. 2007.

[GOO10] W. Goodman, "Cyber Deterrence." STRATEGIC STUDIES 102, 2010

[GRE85] G. S. Green, “General Deterrence and Televison Cable Crime: A Field Experiment in Social Control”, Criminology 23.4, 1985

[HAR96] R. Harknett, “The Information Technology Network and the Ability to Deter: the Impact of Organizational Change on 21 st Century Conflict”, Joint Center for International and Security Studies, 1996.

[HES95] R. Hesseling, "Theft from cars: reduced or displaced?" European Journal on Criminal Policy and Research 3.3, 1995

[HOD15] Honeydrive honeypot bundle distro, http://bruteforce.gr/honeydrive, March 2015.

[HON15] Honeymole, http://www.honeynet.org/project/Honeymole, March 2015.

[HOS15] Honeysnap, https://projects.honeynet.org/honeysnap/, March 2015.

[HUNT09] P. Hunton, "The growing phenomenon of crime and the Internet: A cybercrime execution and analysis model," Computer Law & Security Review, Volume 25, Issue 6, p. 528-535, November 2009

[ILO03] J. Ilonen, "Keystroke dynamics," Advanced Topics in Information Processing, Lecture, 2003.

[IRA08] D. Irani, S. Webb, J. Giffin, C. Pu, "Evolutionary study of phishing," eCrime Researchers Summit, October 2008

[JOY89] R. Joyce, G.K. Gupta, and J. Cook, "User authorization based on keystroke latencies," Dept. of Computer Science, James Cook University of North Queensland, 1989.

[JIA06] X. Jiang, D. Xu, and Y. M. Wang, “Collapsar: A VM-based honeyfarm and reverse honeyfarm architecture for network attack capture and detention”, Journal of Parallel and Distributed Computing, 2006

[KAA06] M. Kaâniche, E. Alata, V. Nicomette, Y. Deswarte and M. Dacier, “Empirical analysis and statistical modeling of attack processes based on honeypots”, Workshop on Empirical Evaluation of Dependability and Security (WEEDS-DSN06), Philadelphia, PA, June 28, 2006

[KER09] O. S. Kerr, "Vagueness Challenges to the Computer Fraud and Abuse Act”, Minn. L. Rev. 94, 2009

[KIP15] desaster/kippo GitHub, https://github.com/desaster/kippo, March 2015

[KIT03] K. Burden and C. Palmer, “Internet crime: Cyber Crime — A new breed of criminal?,” Computer Law & Security Review, Volume 19, Issue 3, p. 222-227, May 2003

[KOH09] J. Kohlrausch, “Experiences with the noah Honeynet testbed to detect new internet worms”, IT Security Incident Management and IT Forensics, 2009.

[LAY92] G. Laycock, C. Austin, "Crime prevention in parking facilities", Security Journal 3.3, 1992

[LEI08a] C. Leita and M. Dacier, "Sgnet: A worldwide deployable framework to support the analysis of malware threat models", European dependable Computing Conference, Washington, DC,USA, 2008

[LEI08b] C. Leita and M. Dacier, "Sgnet: Implementation Insights". Network Operations and Management Symposium (NOMS), 2008

[LEI08c] C. Leita, K. Mermoud and M. Dacier, "ScriptGen: an automated script generation tool for Honeyd", Computer Security Applications Conference, 2005

[LEV03] J. Levine, R. Labella, H. Owen, D. Contis, and B. Culver, "The Use of Honeynets to Detect Exploited Systems Across Large Enterprise Networks", IEEE Workshop on Information Assurance, IEEE Systems, Man and Cybernetics Society, 2003

[LIB15] Libssh – The SSH Library!, https://www.libssh.org, March 2015.

[LOU12] T. A. Loughran, A. Piquero, A. R. Fagan, and E. P. Mulvey, "Differential deterrence: Studying heterogeneity and changes in perceptual deterrence among serious youthful offenders", Crime & Delinquency, 2012

[MAI13] D. Maimon, A. Kamerdze, M. Cukier and B. Sobesto, "Daily Trends and Origin of Computer-Focused Crimes Against a Large University Computer Network An Application of the Routine-Activities and Lifestyle Perspective", British Journal of Criminology, 2013

[MAI14] D. Maimon, M. Alper, B. Sobesto and M. Cukier, "Restrictive Deterrent Effects of a Warning Banner in an Attacker Computer System", Criminology, 2014.

[MAI15] D. Maimon, T. Wilson, W. Ren and T. Berenblum, "On the Relevance of Spatial and Temporal Dimensions in Assessing Computer Susceptibility to System Trespassing Incidents", British Journal of Criminology, 2015.

[MON97] F. Monrose and A. Rubin, "Authentication via keystroke dynamics", Computer and Communications Security, 1997.

[MUN15] Munin, http://munin-monitoring.orgm, March 2015.

[NFD15] NFDump, http://nfdump.sourceforge.net , March 2015.

[NFS15] NfSen – Netflow Sensor, http://nfsen.sourceforge.net, March 2015.

[OPE15] OpenProject, https://www.openproject.org, March 2015.

[OVZ15] OpenVZ Linux Containers Wiki, http://openvz.org/, March 2015.

[PAT87] R. Paternoster, "The deterrent effect of the perceived certainty and severity of punishment: A review of the evidence and issues”, Justice Quarterly 4.2, 1987

[PAU14] A. Pauna and I. Bica "RASSH - Reinforced adaptive SSH honeypot," Communications (COMM), 2014

[PEF15] Inside Windows: Win32 Portable Executable File, https://msdn.microsoft.com/en-us/magazine/cc301805.aspx, March 2015.

[PFS15] PFSense Project – Open Source Firewall and Router Software Distribution, https://www.pfsense.org, March 2015.

[PHO15] J. Nazario, PhoneyC: A Virtual Client Honeypot, USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More, 2009.

[PNG09] I. PL Png, Q. Wang, "Information security: Facilitating user precautions vis-à-vis enforcement against attackers", Journal of Management Information Systems 26.2, 2009

[POL10] M. Polychronakis, K.G. Anagnostakis and E.P. Markatos, "Comprehensive Shellcode Detection Using Runtime Heuristics", in Proceedings of the 26th Annual Computer Security Applications Conference, 2010

[POU05] F. Pouget, M. Dacier, and V. H. Pham, "Leurre.com: on the advantages of deploying a large scale distributed honeypot platform," In ECCE'05, E-Crime and Computer Conference, 2005

[PRO03] N. Provos, "Honeyd: A Virtual Honeypot Daemon," Technical report, Center for Information Technology Integration, University of Michigan, 2003

[PRO07] N. Provos and T. Holz, "Virtual honeypots: from botnet tracking to intrusion detection," Addison-Wesley Professional, first edition, 2007

[RAM00] P. Rämä, R. Kulmala, "Effects of variable message signs for slippery road conditions on driving speed and headways", Transportation research part F: traffic psychology and behaviour 3.2, 2000

[RAM07] D. Ramsbrock, R. Berthier, M. Cukier, “Profiling Attacker Behavior Following SSH Compromises,” Dependable Systems and Networks (DSN'07), 2007

[RAT13] R. P. Rathod, P. U. Bhalchandra, S. D. Khamitkar S.D, S. N. Lokhande, “A Critical Investigation of Botnet,” GJCST-E: Network, Web & Security 13.9, 2013

[ROE99] M. Roesch, “Snort - lightweight intrusion detection for networks,” In Proceedings of the 13th international conference on Large Installation System Administration, 1999

[SAL11] G. Salles-Loustau, R. Berthier, E. Collange, B. Sobesto, and M. Cukier, “Characterizing Attackers and Attacks: An Empirical Study”, Pacific Rim International Symposium on Dependable Computing, 2011

[SBK15] Sebek – The Honeynet Project, https://projects.honeynet.org/sebek/, March 2015.

[SCM15] SCM-Manager, https://www.scm-manager.org, March 2015.

[SHE93] L. W. Sherman, “Defiance, deterrence, and irrelevance: A theory of the criminal sanction,” Journal of research in Crime and Delinquency, 30(4), p. 445-473, 1993

[SHE95] L. W. Sherma and D. L. Weisburd, “General deterrence effects of police patrol in crime hot spots: A randomized, controlled trial”, Justice Quarterly 12:625–48, 1995.

[SOB11] B. Sobesto, M. Cukier, M. Hiltunen, D. Kormann, G. Vesonder, and R. Berthier, “DarkNOC: Dashboard for Honeypot Management”, USENIX conference on Large Installation System Administration, Berkeley, CA, USA: USENIX Association, 2011.

[SOB12] B. Sobesto, M. Cukier, and D. Maimon, “Are Computer Focused Crimes Impacted by System Configurations? An Empirical Study”, Software Reliability Engineering (ISSRE), 2012.

[SON01] D.X. Song, D. Wagner, and X. Tian, “Timing analysis of keystrokes and timing attacks on SSH”, USENIX Security Symposium, 2001.

[SPI02] L. Spitzner. Honeypots, “Tracking Hackers,” Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[SSH15] SSH Analysis, http://www.brendangregg.com/sshanalysis.html, March 2015.

[STR90] Jr, Straub, W. Detmar, “Effective IS security: An empirical study”, Information Systems Research, 1990.

[STU12] I. Studnia, V. Nicomette, M. Kaaniche and E. Alata, "A distributed platform of high interaction honeypots and experimental results," Privacy, Security and Trust (PST), 2012.

[SVN15] Apache Subversion, https://subversion.apache.org, March 2015.

[SYS15] Syslog – Log Management, http://www.syslog.org, March 2015.

[TCP15] TCPDUMP & Libpcap, http://www.tcpdump.org, March 2015.

[THO13] K. Thomas, T. A. Loughran, and A. R. Piquero, “Do individual characteristics explain variation in sanction risk updating among serious juvenile offenders? Advancing the logic of differential deterrence,” Law and Human Behavior, 37(1), p.10, 2013

[THU15] Thug - Python low-interaction honeyclient, http://buffer.github.io/thug/, March 2015.

[VIR15] VirusTotal - Free Online Virus, Malware and URL Scanner, https://www.virustotal.com, March 2015.

[VIS11] V. Visoottiviseth, U. Jaralrungroj, E. Phoomrungraungsuk, and P. Kultanon, “Distributed honeypot log management and visualization of attacker geographical distribution,” In Computer Science and Software Engineering (JCSSE), 2011.

[VMW15] vSphere, http://www.vmware.com/products/vsphere, March 2015.

[VMS15] VMware Server, http://en.wikipedia.org/wiki/VMware_Server, March 2015

[VRA05] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, and S. Savage, “Scalability, fidelity, and containment in the potemkin virtual honeyfarm,” SIGOPS Oper. Syst. Rev., 2005.

[WEI02] N. Weiler, “Honeypots for distributed denial of service attacks,”, Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002.

[WEL08] B. Welsh, D. Farrington, "Effects of Closed Circuit Television Surveillance on Crime: A Systematic Review", Campbell Systematic Reviews 4.17, 2008

[WILL07] C. Willems, T. Holz and F. Freiling, “Toward Automated Dynamic Malware Analysis Using CWSandbox,” IEEE Security and Privacy, 2007

[WIR15] Wireshark – Go deep, https://www.wireshark.org, March 2015.

[XU09] X. Rui; M. Wen-li; Z. Wen-ling, “Defending against UDP Flooding by Negative Selection Algorithm Based on Eigenvalue Sets,” Information Assurance and Security, 2009.

[ZAB15] Zabbix :: The Enterprise-Class Open Source Network Monitoring Solution, http://www.zabbix.com, March 2015.

[ZHU07] J. Zhuge, T. Holz, X. Han, J. Guo, and W. Zou, “Characterizing the irc-based botnet phenomenon,” Peking University & University of Mannheim Technical Report, 2007