ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2014-ENST-0009

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et Réseaux »

présentée et soutenue publiquement par

Davide CANALI

le 12 Février 2014

Plusieurs Axes d'Analyse de sites web compromis et malicieux

Directeur de thèse : **Davide BALZAROTTI**

Jury

M. Levente BUTTYÁN,	Professeur, CrySyS Lab, Budapest University of Technology and Economics	Rapporteur
M. Michael Donald BAILEY,	Professeur, Network and Security Research Group, University of Michigan	Rapporteur
M. Guillaume URVOY-KELLER,	Professeur, Laboratoire I3S, Université de Nice	Examinateur
M. Marc DACIER,	Professeur Associé, Département Réseaux et Sécurité, EURECOM	Examinateur
M. William ROBERTSON,	Maitre de Conferences, Systems Security Lab, Northeastern University	Examinateur
M. Refik MOLVA,	Professeur, Département Réseaux et Sécurité, EURECOM	Examinateur

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

THÈSE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2014-ENST-0009

EDITE - ED 130

ParisTech Ph.D.

Ph.D. Thesis

to obtain the degree of Doctor of Philosophy issued by

TELECOM ParisTech

Specialisation in « Computer Science and Networking »

Publicly presented and discussed by

Davide CANALI

February 12th, 2014

A Multidimensional Analysis of Malicious and Compromised Websites

Advisor : **Davide BALZAROTTI**

Committee in charge

Levente BUTTYÁN,	Associate Professor, CrySyS Lab, Budapest University of Technology and Economics	Reporter
Michael Donald BAILEY,	Associate Professor, Network and Security Research Group, University of Michigan	Reporter
Guillaume URVOY-KELLER,	Professor, Laboratoire I3S, Université de Nice	Examiner
Marc DACIER,	Associate Professor, Département Réseaux et Sécurité, EURECOM	Examiner
William ROBERTSON,	Assistant Professor, Systems Security Lab, Northeastern University	Examiner
Refik MOLVA,	Professor, Département Réseaux et Sécurité, EURECOM	Examiner

THÈSE

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

Acknowledgments

I would like to acknowledge the following people, for their help and support during all the course of my PhD studies.

First, a big thank you goes to my advisor, Davide Balzarotti, for his support and availability at all times during my doctoral studies. He has been much more than an advisor during these three years.

I am grateful to all my present and past colleagues for their inspiration and encouragements, for all the brainstorming sessions and random chats we had during (coffee) breaks, as well as for all the experiences, hacking competitions and projects we completed over these years. I would like to thank Andrea, Andrei, Aurélien, Giancarlo, Jelena, Jonas, Leyla, Luca, Mariano, my papers' co-authors, my fellow Eurecom colleagues, and the good master students I've had the pleasure to work with: Marco, Maurizio, Roberto.

I would also like to thank professors Michael Bailey, Levente Buttyán, Guillaume Urvoy-Keller, Marc Dacier, Will Robertson, and Refik Molva, for agreeing to be reporters and examiners for my Ph.D. dissertation.

Another special thought goes to the members of my family who unfortunately have left us during the last year: zio Bruno, nonno Felice, nonna Angelina, and Crapouille, who has been such a nice and sweet home companion during the last two and a half years.

A very special thank you goes to Elodie, for all her love, support and patience.

Thanks finally to my parents for their constant support and encouragement during my studies: this work is dedicated to them.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 257007.

Abstract

The World Wide Web has become necessary to the lives of hundreds of millions of people, has allowed society to create new jobs, new marketplaces, new leisure activities as well as new ways of sharing information and money. Unfortunately, however, the web is also attracting more and more criminals who see it as a new means of making money and abusing people's property and services for their own benefit.

The World Wide Web is today a very complex ecosystem: for this reason, also attacks that take place on the Internet can be very complex in nature, and different from each other. In general, however, web attacks involve four main actors, namely the attackers, the vulnerable websites hosted on the premises of hosting providers, the web users who end up being victims of attacks, and the security companies and researchers who are involved in monitoring the Internet and in trying to spot and fight malicious or compromised websites.

In this dissertation, we perform a multidimensional analysis of attacks involving malicious or compromised websites. In particular, the focus of our work is to observe the phenomenon of compromised and malicious websites from the point of view of the four actors that are involved in web attacks: attackers, hosting providers, web users and security companies.
Although the study of malicious code on the web is a rather common subject in contemporary computer security literature, our approach based on observing the phenomenon from the points of view of its multiple actors is totally novel, and had never been adopted before.

In particular, we first analyze web attacks from a hosting provider's point of view, showing that current state-of-the-art security measures should allow most providers to detect simple signs of compromise on their customers' websites. However, as we will show in this dissertation, most hosting providers appear to fail in applying even these basic security practices.
Second, we switch our point of view on the attackers, by studying their modus operandi and their goals in a large distributed experiment involving the collection of attacks performed against hundreds of vulnerable web sites.
Third, we observe the behavior of victims of web attacks, based on the analysis of web browsing habits of the customers of a big security company. This allows us to understand if it would be feasible to build risk profiles for web users, somehow similarly to what car insurance companies do for their customers.

Finally, we adopt the point of view of security researchers and focus on finding a solution to the problem of efficiently detecting web attacks that typically spread on compromised websites, and infect thousands of web users every day.

Contents

List of Figures

List of Tables

List of Publications

The results of this dissertation have been published in peer-reviewed conferences. One of these works is currently under review. The list of contributions is the following:

- Davide Canali, Marco Cova, Giovanni Vigna, Christopher Kruegel. *Prophiler: A Fast Filter for the Large-Scale Detection of Malicious Web Pages*. Proceedings of the 20th International World Wide Web Conference (WWW 2011), March 28-April 1, 2011, Hyderabad, India.
- Davide Canali, Davide Balzarotti. *Behind the Scenes of Online Attacks: an Analysis of Exploitation Behaviors on the Web*. Proceedings of the 20th Annual Network & Distributed System Security Symposium (NDSS 2013), February 24-27, 2013, San Diego, CA, USA.
- Davide Canali, Davide Balzarotti, Aurélien Francillon. *The Role of Web Hosting Providers in Detecting Compromised Websites*. Proceedings of the 22nd International World Wide Web Conference (WWW 2013), May 13-17, 2013, Rio de Janeiro, Brazil. *(Best paper nominee)*.
- Davide Canali, Leyla Bilge, Davide Balzarotti. *On The Effectiveness of Risk Prediction Based on Users Browsing Behavior*. To appear in Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2014), June 4-6, 2014, Kyoto, Japan.

Chapter 1

Introduction

In the last few years, the World Wide Web (WWW) has drastically changed society and the way people live and behave on a daily basis. It evolved from a set of static hypertext documents serving a restricted community of scientists, as it was at its birth in 1991, to a worldwide, interconnected network of computing devices providing sophisticated services, generating content dynamically, and allowing people to communicate, share knowledge, data and money from anywhere in the world. A large number of online services are now available, allowing people to make a more efficient and cost-effective use of their time and resources. As a result, many businesses and services are now available online only, and they often shrink and even close their retail points of sale in favor of creating better online shops that cost less and allow for a larger customer base to reach and purchase their services. By shifting to a web-oriented business model, organizations can reduce the costs of services by limiting indirect costs and reducing the number of employees and resources needed, compared to normal shops and agencies. This often has the advantage of reducing companies' ecological footprints too (e.g., by reducing their paper consumption and waste production).

Emails, online banking, online shopping, online tax declarations, and even online voting are only few examples of services that, during the last decade, have revolutionized the way in which people live, do business, interact with each other and with institutions. Even if these changes are contributing to improve the quality of life of the increasing number of people using the web, the Internet is also posing a number of threats to its users. Specifically, the popularity of the web has also attracted miscreants who continuously attempt to abuse its services and users to make illegal profits. This makes any individual, organization, and government a potential target of different kinds of attacks that can originate from the web.

1.1 Malicious Code on the Web

Along with the increasing popularity of online services, criminals and malicious users are looking with interest at the WWW. In fact, an increasing number of

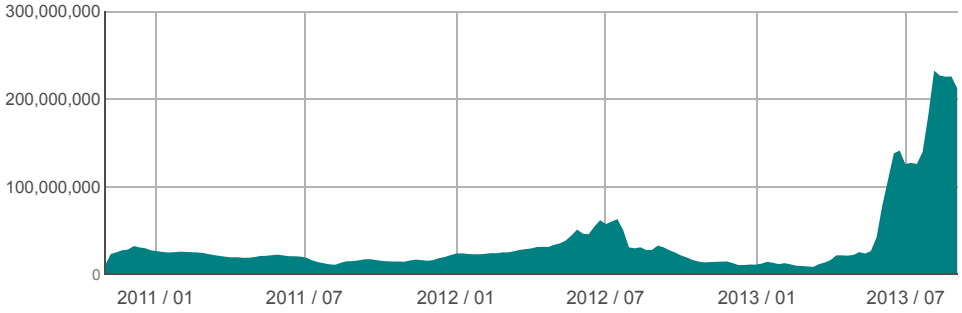

Figure 1.1: Google Safebrowsing warnings shown to web users every week [35]

Internet users means to them an increasing number of potential victims, and thus profits.

Big online services and websites are nowadays storing personal information of several millions of users: gaining access to such data can prove very profitable for any criminal [95] and cause serious financial loss or even bankruptcy for the organization targeted by the attack [106].

In parallel with criminal activities, also governments are starting to exploit the weaknesses of Internet users and services as a means to gain intelligence, for defense purposes [32, 59], but also to attack or disrupt services and facilities of enemy countries [36]. This is often referred to as *Cyber Warfare*.

In this context, we refer to *malicious code* as a broad term to describe any script, system, piece of software or code that can potentially cause undesired effects on a computing device, by attacking any of its "security attributes": confidentiality, integrity and availability (often abbreviated "CIA" in the Information Security community). Examples of such undesired effects can be damaging a system, stealing information, disrupting services, or taking full or partial control over a device.

The web is today the place of choice for *malicious code* to spread, because the interconnected nature of the Internet allows to instantly reach large vulnerable attack surfaces and perform large scale attacks that would not be possible in other scenarios. Also, it often allows skilled attackers to cover their tracks quite effectively.

This trend is well visible in Figure 1.1, which shows that the number of browser warnings raised by the SafeBrowsing [105] service – employed by the most popular browsers on the market (namely Google Chrome, Mozilla Firefox, and Apple Safari) – had almost a 10-fold increase between January 2012 and September 2013. This is supported also by reports from security firms, such as Websense, that in its 2013 Threat Report announced a nearly 600% increase in the number of malicious websites on the Internet [126].

Even more interestingly, according to the Websense report, 85% of websites hosting malicious code were compromised legitimate hosts. Similar increases in the number of malicious domains are reported also by Symantec's 2013 Internet

Security Threat Report [117]. Unfortunately, as the report shows, the increase was not only in the number of malicious domains, but also in the number of vulnerabilities targeting the top five browsers, which raised from 351 in 2011 to a whopping 891 reported total vulnerabilities in 2012.

This explains why, in the last few years, vulnerabilities in web applications and in web clients have become the most common and successful vector for criminals to spread malicious code on the web.

At the same time, security companies and big Internet service providers are spending more and more energies in trying to stop malicious actors from spreading malicious code on the web. Good security practices and resources are often brought to the attention of users through awareness campaigns, by schools, banks and national or international initiatives (e.g., the National Cyber Security Awareness Month [23, 27]). Companies typically employ Intrusion Detection (IDS) and Intrusion Prevention Systems (IPS) [81, 102] in order to detect or block the spreading of *malicious code* on their networks, while service providers and search engines protect their users by employing blacklists or more sophisticated threat detection methods (e.g., [13, 105, 120]). Nonetheless, every security measure comes at a cost, and effectively and efficiently protecting people and organizations from Internet threats is still an open problem. We are facing a cat and mouse game: as security companies and researchers come up with advanced techniques to detect and block *malicious code*, criminals are able to develop increasingly sophisticated malware, capable of bypassing state-of-the-art protection systems.

1.2 Attack Model

As explained in the beginning of this chapter, the World Wide Web has become a very complex ecosystem. Its distributed nature, and the wide variety of services and entities acting in it make it very difficult to enumerate all the possible attack types that can happen on the web. However, there is a very common attack model that can be applied to a wide variety of today's web attacks. We will refer to this as the *web attack model*, and we will have the opportunity of analyzing it in deep detail during the course of this dissertation. Our web attack model takes into consideration four main actors: *attackers*, vulnerable websites hosted on the premises of hosting *providers*, *web users* who end up being victim of attacks, and, finally, *security companies and researchers* who constantly monitor the Internet in order to spot malicious or compromised websites.

The typical scenario representing the web attack model we adopt in this thesis is represented in Figure 1.2. The model involves the four different actors we mentioned; however, it is a general representation we can use to describe a number of specific attack scenarios on the web. In each of these scenarios, all the actors, or only part of them, interact in a specific manner, that may be completely different from case to case.

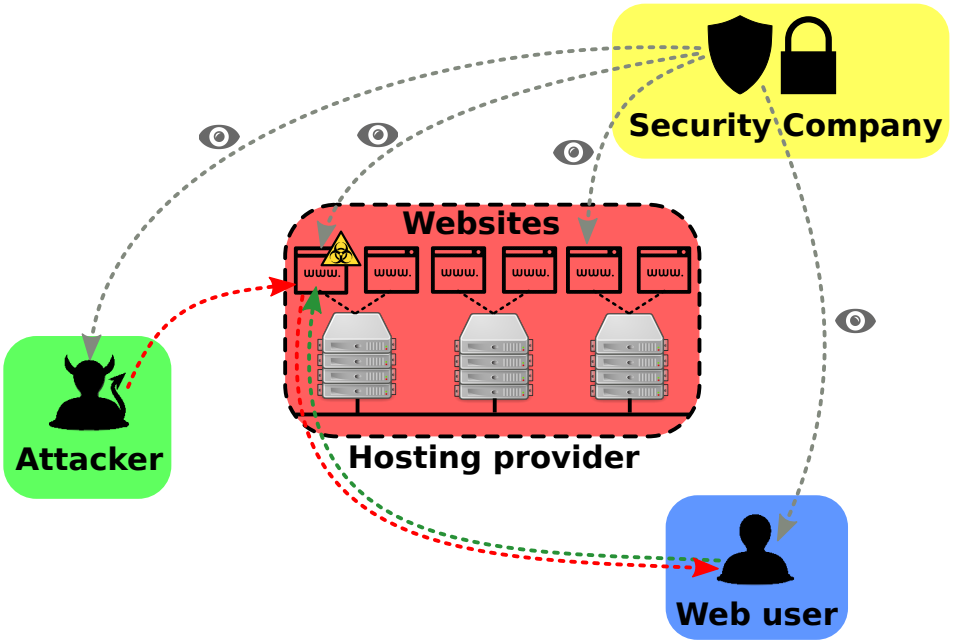

Figure 1.2: A general web attack model

For example, the attack model depicted in Figure 1.2 can be used to describe **drive-by download attacks**. These attacks are today one of the most effective mechanisms for cyber criminals to infect web users on the Internet. In a typical drive-by download scenario, attackers host exploits targeting common browsers on malicious (or legitimate compromised) websites. In particular, drive-by-download attacks install malware on victim machines by exploiting vulnerabilities in the user's browser or in one of the browser's plugins. In order for this to work, the attacker usually injects malicious scripting code (typically JavaScript), into a web page. When a victim visits the malicious page crafted by the attacker, the malicious code is executed, and, if the victim's browser is vulnerable, the browser is compromised, typically installing malware on the victim's machine. This kind of attacks have become pervasive over the last few years [33, 34].

This kind of attacks allow criminals to install malicious software on a multitude of hosts, with relatively little effort, and to make easy money by selling the information stolen by these means to other criminals.

Once a victim machine is infected, the malware program connects to a command and control (C&C) server under the attacker's control and, typically, waits for instructions, while stealing private information from the victim. In this fashion, the infected hosts form a *botnet*, which is a network of machines under the direct control of cyber criminals. As recent studies have shown [110, 114], a botnet can contain hundreds of thousands of compromised hosts, and it can generate significant income for the botmaster who controls it. This kind of attacks will be studied

in detail in Chapter 6.

Another very common kind of attack that can be modeled by the web attack model adopted in this dissertation is **phishing**. In a phishing attack, a criminal sets up a website or a set of pages that resemble in every detail a target website typically holding sensitive private user information, such as an online bank, a social networking website or a web email provider. If a user is successfully tricked into submitting his or her private information, such as login credentials and credit card numbers, to the fake website set up by the attacker, the criminal can easily collect the stolen data and sell it on underground markets.

Both phishing and drive-by download attacks are typically set up by attackers on compromised websites. In general, the website compromise is itself another case of web attack, typically consisting in the attacker gaining access to the machine hosting the website through a vulnerability in a web application (e.g., input validation vulnerability), or through other means such as stolen credentials. Both types of attacks can be modeled by our web attack model, by considering only two actors: the attacker and the provider hosting the target website.

Website compromises are themselves a very general case of web attacks, and as mentioned, can be carried out through different means. There can be different reasons pushing criminals to compromise websites. One of these, as explained before, is to use the compromised infrastructure to mount further attacks against the visitors of the website (as in the case of drive-by downloads and phishing). In other cases, though, the attack may be aimed at directly stealing sensitive information from the compromised host.

Indeed, stealing sensitive data is today one of the most common reasons behind web attacks. Information, and especially confidential documents or private data from users and companies, is a very sought after bounty that can be sold at high prices on black markets or to companies and governments interested in stealing industrial or military secrets.

Big data breaches always hit the news because of their global reach, typically affecting accounts and user information at national or worldwide scales.

As Figure 1.3 shows, an incredibly high number of personal records have been stolen from companies and organizations in the last 9 years. The total number of stolen records already passed the billion, and this only by taking into account the biggest data breaches that hit important news channels. Apart from the high number of data records stolen every year, the last three years have been marked by an increasing number of attacks exfiltrating very highly sensitive information (shown in darker colors in the picture), such as email address and password or full credit card and bank account information. Numbers shown in the graph are, of course, only a lower bound for the real amount of data that has been leaked from organizations in the last years, as it takes into account only incidents that have been publicly reported. Moreover, the graph does not take into account the massive amount of

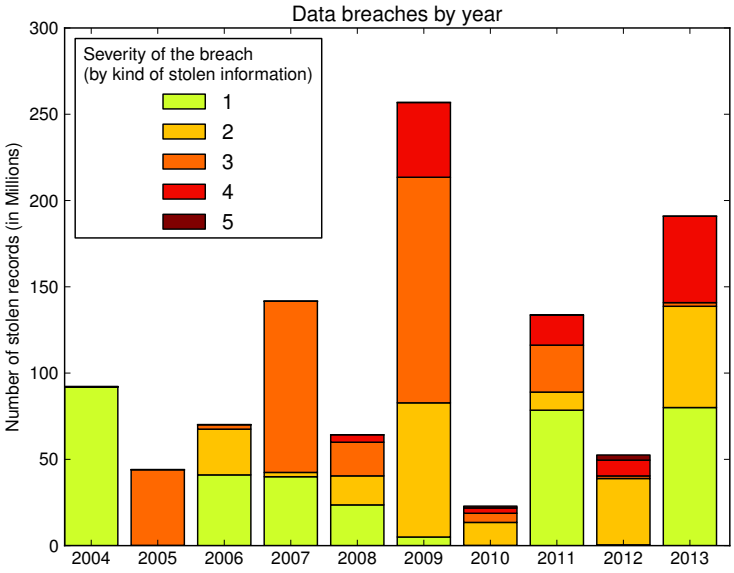

Figure 1.3: Biggest publicly documented data breaches since 2004 [46]

data that criminals steal from individuals every day, by luring them through phishing and social engineering attacks, or by infecting their computers with malware.

Publicizing web attacks, and especially data breaches, is another trend of some of today's cyber attacks. Organizations such as Anonymous [128], for example, publicly advertise their attacks, sometimes even before the attack takes place. This is generally for the purpose of hitting the news and diffuse ethical slogans, as the alleged reason for most of the group's attacks is "fighting censorship and promoting transparency and freedom of information". A new word has been defined to describe such events, *hacktivism*, from *hacker* and *activism*. It has to be noted that, in contrast with the early days of "old school" hacking, being an hacktivist is not an *elite* role, reserved to a few "elected" of members. Instead, anyone can join hacktivist chats and operations [6] and use free tools to anonymize his own traces while launching attacks as part of the group.

Finally, as the picture representing our web attack model shows, security companies (or researchers), even if not directly involved in the attacks as the other actors, are typically present behind the scenes of all web attacks. When not directly trying to stop attackers or helping out hosting providers and their users, in fact, these companies continuously monitor the moves of known attackers, scan websites looking for malicious code and collect information from their clients in order to timely detect and analyze new Internet threats.

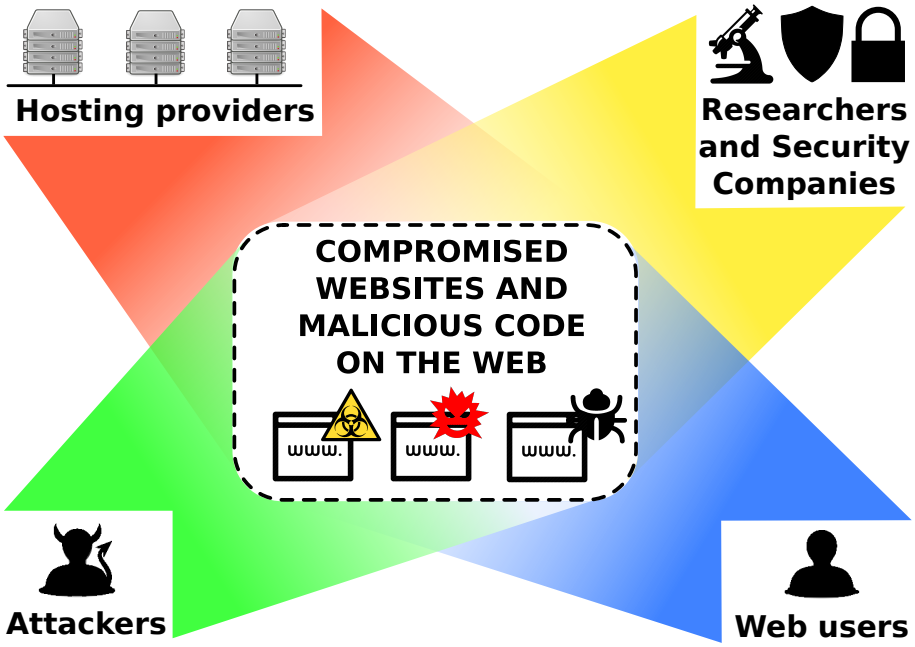

Figure 1.4: Points of view adopted in this dissertation

1.3 Goals

Over the last few years, we have been witnessing a rise in the number of web attacks and compromised websites set up by criminals in order to spread malicious code. Unfortunately, this trend is not likely to stop, and companies and governments, now more than ever, are spending a high amounts of money and resources in trying to combat malicious activities on the Internet.

A successful attack prevention plan starts from a good understanding of the ecosystem that is subject to the attack – the web, in our case.

This is why, in this dissertation, we tackle the problem of compromised websites and malicious code on the web from four distinct perspectives. These perspectives allow us to study the phenomenon of web attacks from the point of view of each actor that is typically involved in web attacks: the attackers, the web hosting providers hosting attacked or infected machines, the researchers or security companies trying to detect and prevent the attacks, and the users, who represent the victims of web attacks. Figure 1.4 graphically shows the four points of view we adopt in this dissertation. Each of them corresponds to a distinct chapter in this dissertation.

Although the study of malicious code on the web is a rather common subject in contemporary computer security literature, very few works have studied this topic by observing it from the points of view of its multiple actors.

The goal of this work is, thus, to provide the reader with a comprehensive view of the phenomenon of compromised websites and malicious code on the web, and

to dive deeper into it from the four different points of view, or "angles" described above. This will have the benefit of giving a full view of the ecosystem, by observing it from its different faces. In particular, we will analyze how attackers find their targets on the Internet, how they exploit them and what are their objectives once they gain access to a compromised system. Second, we will adopt the point of view of web hosting providers, the companies actually hosting the websites that typically end up being infected by criminals. In this case, we will study how they act with regard to preventing malicious uses of their systems, how they detect compromised customer accounts, and how they handle incidents and abuse notifications. Third, we will study how researchers and security companies can proactively scan the Internet and find web pages hosting malicious code, thus, possibly contributing in shutting down malicious sites before they are able to infect victims. Finally, we focus our analysis on the behavior of web users, to understand if certain categories of users are more at risk than others when surfing the web, and, if so, how we can predict it by looking at their behavior.

1.4 Contributions

This dissertation analyzes the spreading of malicious code on the web during the last few years, focusing in particular on compromised web sites. Our research approach allows us to study this phenomenon from the points of view of all the actors that are commonly involved in it: web users, hosting providers, attackers and security companies or researchers (see Figure 1.4). Overall, we make the following contributions to the area of web security:

- We develop a new approach for the collection and analysis of web attacks and malicious code, based on the creation of a network of fully functional honeypot websites. This allows us to study and better understand current web attacks trend, and the reasons behind these actions.
- We present a measurement study on the way shared web hosting companies handle the security of their customers, on a worldwide scale. This includes testing whether any attack prevention mechanism is put in place, whether providers are able to detect obvious signs of compromise on their customers' accounts, and testing if abuse complaints are handled in a timely and appropriate manner.
- We introduce a novel approach that uses static analysis and machine learning techniques for the large-scale detection of web pages launching drive-by-download attacks. We further show this approach can be successfully applied in a real-world deployment with the purpose of efficiently filtering out benign web pages, and forwarding only suspicious ones to dynamic analysis systems.
- We develop a novel approach for building usage profiles for web users, based on their web browsing history. We analyze whether profile characteristics can be correlated to higher or lower chances of visiting malicious websites,

and present a comprehensive study on the effectiveness of risk prediction based on users' browsing behaviors.

The rest of the dissertation is organized as follows. Chapter 2 presents an overview of related work in our area of research, namely web attacks and the spreading of malicious code on the Internet. Chapter 3 describes web attacks from the point of view of an attacked hosting provider. Chapter 4 shows insights on current attack trends and exploitation behaviors on the web, targeting small and medium sized websites, while Chapter 5 describes novel approaches that can be used in order to detect users that have high chances of becoming victims of web infections, by analyzing their browsing history and habits. Chapter 6 discusses novel techniques that can be used to search and detect malicious web pages on the Internet, and presents a system we have developed for this purpose. Finally, Chapter 7 proposes directions for future work and concludes.

Chapter 2

Related Work

As explained in the previous chapter, the vast majority of cyber attacks are nowadays carried out through the web. For this reason, in the last few years, web security has become a very active field of study, both for academia and security companies. A very large number of research papers, projects, tools, and reports have been produced in the last years, covering almost the whole spectrum of computer security issues related to the web. In this chapter, we present some of the most relevant works for this thesis, and in particular projects and publications related to each of the four "points of view" we adopt in the dissertation, to study the phenomenon of compromised websites and malicious code on the Internet. Each of the following sections will present works related to one of the four aforementioned points of view.

2.1 Web Attacks and Hosting Providers

Several works have studied the threats that affect websites all around the world as well as users visiting infected pages [86, 92–94]. Research has been focusing also on the ways in which criminals exploit search engines in order to reach their victims, by poisoning search results for popular queries [50], and on possible solutions to detect them [66]. The work presented by Moore et al. [75] starts from the analysis of logs collected from phishing websites to demonstrate that web attackers typically search for vulnerable websites by employing specific search terms on common search engines. Also, authors find out that websites compromised with the purpose of hosting phishing pages are much more likely to be re-compromised in the future if they can be identified by attackers through specific searches on search engines. John et al. [52], instead, studied how attackers find, compromise and misuse vulnerable servers on the Internet, by employing honeypots that actively attract attackers and dynamically generate honeypot pages related to what they search for, in order to lure them and collect attack logs. This work is closely related to our analysis of the behavior of web attackers, and will be discussed more in detail in Section 2.2.

Researchers have also studied how all malicious activities are combined by criminals in order to be able to conduct attack campaigns infecting tens of thousands of hosts [115].

In a recent paper by Bau et al. [7], the authors evaluate current commercial tools for detecting vulnerabilities in web applications. This is related to what web hosting providers can do in order to detect, or even prevent, attacks on their customer's websites. As this work shows, however, the tested commercial tools mainly rely on black-box approaches, and are not able to find all possible vulnerabilities.

Recently, a web hosting provider [21] announced an improvement of his hosting offer by adding free automated website vulnerability scanning, fixing and recovery. Such a service is presumably running as a white-box approach on the network and server side. This service is related to what, in our work, we refer to as "add-on" security services. Unfortunately, this service was announced when our experiments were already completed, and it was therefore not possible to integrate it into our results.

The study most closely related to our analysis of compromised websites from the hosting providers' point of view, has been presented by Commtouch in 2012 [16]. In this report, the authors surveyed 600 compromised websites owners and, among other things, reported on the process by which the websites owners became aware of the compromise. However, this was done with a publicly advertised pool on *detected* compromised websites and may therefore be biased. Also, compromised website owners often answered questions with "I do not know/I have no idea", as they often had little knowledge of website management and system administration. Another problem was that, in some cases, the owners had no way to access the website's logs, or did not know how to do that. One of the interesting findings was however that the compromise, and the way in which the hosting provider handled it, often affected the site owner's willingness to keep hosting his or her website with the same company. In particular, 28% of the surveyed users said they were considering moving to another hosting company after the incident, while only 12% of them said they had become more appreciative of their current provider.

Finally, some past work has been focusing on studying the take-down process employed in the case of phishing websites [72, 73]. This is related to some of the findings we report in Section 3.3.3, but is aimed at studying the phenomenon at a ISP and hosting provider level, rather than analyzing the providers' responses one by one and provide details on how they react to abuse notifications. The work presented by Moore and Clayton [72], in particular, shows that website removal is an effective way to block phishing websites, but that it is not fast enough to completely mitigate the problem. As the authors report, and as we confirm in this dissertation (cf. 3.3.3), some providers are fast at taking down websites, but others react when it is already too late.

As of today, previous works in this area of research have been mostly focusing on the analysis of how web hosting accounts and websites get exploited, but very few works have studied the phenomenon of website compromises from the pro-

viders' point of view. The work presented by Commtouch [16] has been the first attempt to analyze also web hosting providers' behaviors when it comes do deal with such incidents. The study is however based on a survey, it is somehow limited in its findings and its approach cannot be easily automated. To our knowledge, our analysis of web hosting providers and their handling of compromised websites is the first attempt to systematically study, on a worldwide scale, how these providers act with regard to the security of their customers and of their own infrastructure - focusing in particular on the detection of compromised accounts, rather than vulnerabilities.

2.2 Behavior of Web Attackers

Honeypots are nowadays the tool of choice to detect attacks and suspicious behaviors on the Internet. Most of the research done with the purpose of observing or analyzing attackers' actions has been carried out by using honeypot systems. The purpose of these systems is, typically, to collect information related to all actions performed by an attacker on a vulnerable system, or seemingly vulnerable one, in form of event logs, network traces, and files that have been created and modified during the attack. Honeypots can be classified in two categories: client honeypots, which detect exploits by actively visiting websites or executing files, and server honeypots, which attract the attackers by exposing one or more vulnerable (or apparently vulnerable) services. The behavior of web attackers can be studied by employing honeypots of the second type, as in this case we are interested in observing what attackers do on the server, after a website has been compromised. Several server-side honeypots have been proposed in the past years, allowing for the deployment of honeypots for virtually any possible service. In particular, we can distinguish two main classes of server honeypots: low-interaction and high-interaction ones. The first class of honeypots only *simulates* services, and thus can observe incoming attacks but cannot be really exploited. These honeypots usually have limited capabilities, but are very useful to gather information about network probes and automated attack activities. Examples of these are honeyd [91], a framework for virtual honeypots that is able to simulate the networking stack of different operating systems, Leurre.com [89], a project for the development and deployment of a distributed honeypot network spanning several countries and collecting network-level attack information, and SGNET [60], a scalable framework for the deployment of low interaction honeypot networks, where each honeypot is able to collect almost the same amount of information as a real high interaction honeypot if dealing with server based code injection attacks.

High-interaction honeypots, on the other hand, present to the attacker a fully functional environment that can be exploited. This kind of honeypots is much more useful to get insights into the modus operandi of attackers, but usually comes with high setup and maintenance costs. Due to the fact that they can be exploited, high-interaction honeypots are usually deployed as virtual machines, allowing their

original state to be restored after a compromise. Another issue with high interaction systems is the liability of the honeypot manager for any illicit action or damage that attackers can cause while abusing the honeypot's resources. In the study conducted by Nicomette et al. [80], the authors ran a high interaction SSH honeypot for more than one year and collected logs of all activities performed by the attackers while trying to gain access to the machine, and once inside the honeypot. This allowed them to draw a basic picture of the behaviors of attackers targeting servers running SSH.

The study of attacks against web applications is often done through the deployment of *web* honeypots. In recent years, the scientific and industrial community has developed and made available several low-interaction web honeypot solutions. The Google Hack Honeypot [31] has been designed especially to attract attackers that use search engines to find vulnerable web applications. Glastopf [100] and the DShield Web Honeypot project [24] are both based on the idea of using templates or patterns in order to mimic several vulnerable web applications. They emulate a vulnerable web server hosting many web pages and web applications, thus simulating an attack surface exposing hundreds of vulnerabilities. These tools are also able to respond to attackers by presenting templates associated with certain request patterns (e.g. by replying with a fake passwd file if the attacker seems to be trying to do a local file inclusion looking for the system's passwd file).

Another interesting approach for creating low interaction web honeypots has been proposed by John et al. [53]: with the aid of search engines' logs, the system proposed by the authors is able to identify malicious queries from attackers and automatically generate and deploy honeypot pages responding to the observed search criteria. Unfortunately, the results that can be collected by low-interaction solutions are limited to visits from crawlers and automated scripts. Any manual interaction with the system will be missed, because humans can quickly realize the system is a trap and not a real functional application. Apart from this, the study by John et al. [53] collected some interesting insights about automated attacks. For example, the authors found that the median time for honeypot pages to be attacked after they have been crawled by a search engine spider is 12 days, and that local file disclosure vulnerabilities seem to be the most sought after by attackers, accounting to more than 40% of the malicious requests received by their heat-seeking honeypots. Other very common attack patterns were trying to access specific files (e.g., web application installation scripts), and looking for remote file inclusion vulnerabilities. A common characteristic of all these patterns is that they are very suitable for an automatic attack, as they only require to access some fixed paths or trying to inject precomputed data in URL query strings. The authors also proposed a setup that is similar to the one adopted in our study, but they decided not to implement it due to the their concerns about the possibility for attackers to use infected honeypot machines as a stepping stone for other attacks. We explain how we deal with this aspect in Section 4.2.1.

If one is interested in studying the real behavior of attackers, one has to take a different approach based on high interaction honeypots. A first attempt in this

direction was done by the HIHAT toolkit [78]. This solution allows to transform arbitrary PHP applications into high-interaction web honeypots by instrumenting the application's code with special logging functions, and provides a graphical user interface to manage and monitor the honeypot. Unfortunately, the evaluation of the tool did not contain any interesting finding, as it was run for few days only and the honeypot received only 8000 hits, mostly from benign crawlers.

However, some similar work has been done on categorizing the attackers' behavior on interactive shells of high-interaction honeypots running SSH [80, 97]. Some interesting findings of these studies are that attackers seem to specialize their machines for some specific tasks (i.e., scans and SSH bruteforce attacks are run from machines that are different from the ones used for intrusion), and that many of them do not act as knowledgeable users, using very similar attack methods and sequences of commands, suggesting that most attackers are actually following cookbooks that can be found on the Internet. Also, the commands issued on these SSH honeypots highlight that the main activities performed on the systems were checking the software configuration, and trying to install malicious software, such as botnet scripts. As we will describe in Section 4.5, we also observed similar behaviors in our study.

Part of our study, finally, concerns the categorization of files collected by honeypots. Several papers have been published on how to detect similarities between source code files, especially for plagiarism detection [12, 104]. These systems propose new ad-hoc metrics for measuring the amount of shared information between compiled binaries, and to compactly model their structural information. Other similarity frameworks have been proposed for the detection of similarities between images and other multimedia formats, mostly for the same purpose.

Unfortunately, when dealing with files collected from web attacks, as explained in Chapter 4, a good portion of the collected data consists in obfuscated source code (that renders most plagiarism detection methods useless), binary data or archives. Also, many of the proposed plagiarism detection tools and algorithms are very resource-demanding, and difficult to apply to large datasets. These reasons make the plagiarism detection approaches unsuitable for our needs.

Other approaches have been proposed in order to recover lineage for a given binary program [49], or to detect similar compiled programs even in case they employ packing and obfuscation [48]. However, these solutions are specific to the world of compiled binaries, and mostly suitable for the analysis of malware samples. Even if they are effective in detecting similarities between binaries, they have high execution times and cannot be applied in a scenario as generic as ours, requiring the analysis of many different types of files.

The problem of classifying and fingerprinting files of any type has, however, been studied in the area of forensics. In particular, some studies based on the idea of similarity digest have been published in the last few years [57, 103]. These approaches have been proven to be reliable and fast with regard to the detection of similarities between files of any kind, being based on the byte-stream representation of data. We chose to follow this approach, and use the two tools, namely

ssdeep [57] and *sdhash* [103], for our work.

As this section showed, there has been a significant amount of research in the area of web security that focused on how attackers exploit websites and how they spread malicious code on the Internet. Previous research also covered the development of tools, such as honeypots, whose goal is to better understand and analyze attacker's malicious actions. However, the limitation of most of the studies proposed so far is that they focused on only one aspect of the phenomenon, e.g., observing *how* attackers find and exploit vulnerable websites. This is because all existing projects used non functional honeypots, providing attackers with fake applications that cannot be really exploited. This means that all the steps that would commonly be performed by an attacker after the exploitation will be missed. In Chapter 4, we instead focus our analysis on two separate aspects: the exploitation phase, in which we investigate *how* attacks are performed, and the post-exploitation phase, in which we analyze *what* attackers do after they take control of the web application. For this reason, the study we present in Chapter 4 is the first large scale evaluation of the post-exploitation behavior of attackers on the web. This is made possible by employing a new architecture we developed for this task: a network of hundreds of fully functional honeypot websites, hosting real vulnerable web applications, whose aim is to attract attackers and collect information on what they do during and after their attacks. By collecting the files created and modified during each attack, moreover, we were able to identify the behavior behind each action performed both during and after the exploitation of a web application. This allows us, for the first time, to draw a general picture of the landscape of web attacks targeting small and medium-size websites.

2.3 The User Point of View

Somehow similarly to the point of view of hosting providers on web attacks and compromised websites (cf. Chapter 2.1), also the study of these phenomena from a user's perspective has not been thoroughly conducted yet, especially in academia. The main cause of this is the unavailability of data about people's web browsing habits and experiences. Thus, the number of studies that tried to understand if there is any relation between users' behaviors or characteristics and their probability of visiting malicious web pages is still very limited. Moreover, most of the existing studies have been built upon the observation of very limited customer bases, or on clinical-style case studies based on data collection and surveys on tens or hundreds of users in a monitored environment [61].

2.3.1 User-based Risk Analysis

One of the first studies that sought to understand the risk factors behind user infections was carried out by researchers at École Polytechnique de Montréal and

Carleton University [61]. This study, which has a similar nature to works that evaluate medical interventions, examined the interactions among three important players: the users, the AV software and the malicious software detected on the system. The authors provided 50 users that accepted to join the experiment with a laptop that was configured to constantly monitor possible malware infections and collect information about how the users behaved in such cases. The results of the experiment show that user behavior is significantly related to infections, but that demographic factors such as age, sex, and education cannot be significantly related to risk. Furthermore, innocuous categories of websites such as sports and Internet infrastructure are associated with a higher rate of infection when compared to other categories, such as porn and illegal content sites, that common sense traditionally associates with higher risk. Similarly, and surprisingly, computer expertise seems to be one of the factors positively related to higher risk of infection.

Onarlioglu et al. [84] studied the behavior of users when they are faced with concrete Internet attack scenarios. The authors built an online experimental platform that was used to evaluate the behavior of 164 users with different backgrounds. Their findings confirm that non-technical users tend to fail in spotting sophisticated attacks, and that they are easily deceived by tricky advertising banners. On the other hand, they are able to protect themselves as effectively as technical users when dealing with simple threats.

Maier et al. [68] conducted a study of the security hygiene of approximately 50K users from four diverse environments: a large US research institute, a European ISP, a community network in rural India, and a set of dormitory users of a large US university. The paper analyzes anonymized network traces, which were collected from each observed environment for a period that ranges between 4 and 14 days, containing only the first bytes of each connection. The results of the analysis indicate that having a good security hygiene (i.e., following antivirus and OS software updates) has little correlation with being at risk. However, on the other hand, risky behaviors such as accessing blacklisted URLs double the likelihood of becoming infected with malware. Unlike our work, this paper has the advantage of being able to monitor all the Internet activity of the users, and thus is not limited to the analysis of web browsing traffic. However, it considers only malicious behaviors that overtly manifest themselves at a network level, e.g., sending spam emails, performing address scans or communications with botnet C&C servers. For this reason, attacks that produce little traffic, or install themselves on victim's machines and remain latent for long periods of time, may have been missed.

Finally, a recent report by TrendMicro and the Deakin University provided an analysis on the Australian web threat landscape [56]. The report states that, in average, 0.14% of the web browsing hits collected by the AV company are malicious in nature. One interesting finding is that Australians seem to incur in a higher percentage of daily malicious hits during holidays and weekends, when compared to weekdays. Moreover, the percentage of malicious hits rises during night time, with a peak around 4 am. As explained in Section 5.3.1, these findings are also confirmed as a worldwide trend by our experiments. Finally, the reported statistics

show that one out of eight Australian IP addresses are exposed to web threats, on a typical day. In our study, instead, we find a higher risk of exposure to web threats for users in our dataset (19% of *at risk* users, overall).

Our work is fairly different from these studies in many respects. Compared to the majority of previous works, we perform our analysis on a much larger dataset (i.e. three months of data generated by 160K distinct users). Moreover, our analysis does not rely on personal information about the users such as their educational background, sex and age. We significantly extend the study of the Australian threat landscape by conducting similar analysis on a *worldwide* basis. In addition, we performed a more precise and deeper analysis by building user profiles based on over 70 features, and we tried to understand if different risk categories have different probabilities to end up in malicious web sites.

2.3.2 User Profiling

Olejnik et al. [83] presented a study in which they evaluated the possibility of fingerprinting users given their past web browsing history. The methodology the authors adopt to fingerprint users was able to profile 42% out of approximately 380,000 users involved in the study. From the experiment they performed with only 50 web pages, they conclude that categorization information of visited websites could be a useful parameter to build more accurate user profiles. The results of our study confirm that categorization information could be used for user profiling.

The problem of user profiling has been largely studied within recommender systems [22, 70, 71], to help users find topics that are in their interest, while hiding those topics that are unattractive to them. Therefore, the goals of user profiling in recommender systems' research are completely different from ours.

2.4 Detection of Drive-by-Download Attacks

In the last few years, the detection of web pages used to launch drive-by-download attacks has become an active area of research and several new approaches have been proposed. In Chapter 6, we will describe the design and implementation of a fast filter for the large scale detection of malicious web pages. The purpose of this system is to proactively crawl the web and filter the collected web pages in search for potentially malicious ones. As such, it can be very useful to researchers and antivirus companies willing to collect and discover new malicious pages on the Internet. This section briefly goes through the papers and projects that have been published in this field of research in recent years.

2.4.1 Dynamic approaches

Dynamic approaches use honeyclient systems to visit web pages and determine if they are malicious or not. In high-interaction honeyclients, the analysis is performed by using traditional browsers running in a monitored environment and

detecting signs of a successful drive-by-download attack (e.g., changes in the file system, the registry, or the set of running processes) [76, 92, 107, 124]. In low-interaction honeyclients, the analysis relies on emulated browsers whose execution during the visit of a web page is monitored to detect the manifestation of an attack (e.g., the invocation of a vulnerable method in a plugin) [5, 17, 79].

Both high- and low-interaction systems require to fully execute the content of a web page. This includes fetching the page itself, all the resources that are linked from it, and, most importantly, interpreting the associated dynamic content, such as JavaScript code. These approaches usually yield good detection rates with low false positives, since, by performing dynamic analysis, they have complete "visibility" into the actions performed by an attack. The down-side is that this analysis can be relatively slow, because of the time required by the browser (either simulated or real) to retrieve and execute all the contents of a web page, taking from a few seconds to several minutes depending on the complexity of the analyzed page.

Scalability issues with today's honeyclient systems (relatively slow processing speed combined with relatively high hardware requirements) motivated our work on a filtering system that could be used to quickly scan a large number of web pages and forward to the dynamic analysis component (e.g., a high interaction honeyclient) only the suspicious ones. The filter we developed, described in Chapter 6, achieves higher performance by forgoing dynamic analysis (e.g., the interpretation of JavaScript code), and relying instead on static analysis only.

Other approaches have been proposed, such as the NOZZLE [98] and ZOZZLE [19] systems by Microsoft Research, that tackle the problem of detecting malicious JavaScript code by instrumenting the browser's JavaScript engine. These solutions achieve good accuracy and have the advantage of running inside a normal browser, but are mostly aimed at detecting one class of attacks (heap spraying) and still suffer from a non negligible performance overhead, when it comes to analyzing a large number of pages. The paper presenting ZOZZLE [19], however, claims the system can be employed also as an offline filter for web pages (like Prophiler, the solution we present in Chapter 6). In this case, the system would be able to detect heap spraying JavaScript attacks in a fast and lightweight way. Still, as the authors claim, the emphasis of ZOZZLE is on very low false positive rates, which can sometimes imply higher false negative rates. Also, being focused on heap spraying, ZOZZLE would probably miss some malicious pages using other types of attack, that Prophiler would be capable of detecting.

2.4.2 Static approaches

Static approaches to the detection of drive-by-download attacks rely on the analysis of the static aspects of a web page, such as its textual content, features of its HTML and JavaScript code, and characteristics of the associated URL.

String signatures (i.e., string patterns that are common in malicious code) are used by traditional antivirus tools, such as ClamAV [14], to identify malicious

pages. Unfortunately, signatures can be easily evaded using obfuscation. Therefore, these tools suffer from high false negatives rates (earlier studies report between 65% and 80% missed detections [17, 99]), which make them unsuitable for filtering likely malicious pages. Our filter is also based on static techniques, but achieves better detection rates by relying on a combination of several characteristics of a web page based on its HTML content, JavaScript code, and other URL and host features, rather than simple static string patterns. Moreover, our filter can be more aggressive in labeling a page as malicious. The reason is that incorrect detections are discarded by the subsequent (dynamic) back-end analysis, and hence, false positives only incur a performance penalty.

Several systems have focused on statically analyzing JavaScript code to identify malicious web pages [28, 63, 109]. The most common features extracted from scripts are the presence of redirects (e.g., assignments to the *location.href* property), the presence of functions commonly used for obfuscation/deobfuscation (such as *fromCharCode()*), calls to the *eval()* function, large numbers of string manipulation instructions, abnormally long lines, and the presence of shellcode-like strings. In our filter, we considerably extend the set of JavaScript features used for detection by previous works: for example, we detect the presence of sections of code resembling deobfuscation routines, we take into consideration the entropy of both the entire script and of the strings declared in it, we identify the number of event attachments, and we analyze both Document Object Model (DOM) manipulation functions and fingerprinting functions (such as *navigator.userAgent()*).

Seifert et al. [109] also use the characteristics of the HTML structure of a web page as indicators of maliciousness. For example, they consider the visibility and size of `iframe` tags and the number of script tags referencing external resources. We extend this analysis by adding more than ten new features, such as the number of out-of-place elements (e.g., scripts outside `<html>` tags), as well as the percentage of the page occupied by JavaScript code.

Characteristics of URLs and host information have been used in the past to identify sites involved in malicious activity, such as phishing and scams. Garera et al. use statistical techniques to classify phishing URLs [30]. Ma et al. [67] use lexical properties of URLs and registration, hosting, and geographical information of the corresponding hosts to classify malicious web pages at a larger scale. In Chapter 6, we discuss the issues involved in applying this approach to detecting pages involved in drive-by-downloads (opposed to threats such as phishing and scam pages), and propose a number of new features that are effective in this context.

It is important to observe that we did not simply introduce new detection features for the sake of being able to point to a longer feature list. As our experiments demonstrate, adding these additional features significantly contributes to the improved accuracy of our system.

Several of the detection tools described here have been used as components of crawler-based infrastructures designed to effectively find malicious web pages,

e.g., [44,76,92]. In Chapter 6, we describe a similar setup, where *Prophiler* is used as a fast filtering component.

After the publication of our work, and starting from the study of the system we designed (see Chapter 6), Eshete et al. proposed BINSPECT [25]. BINSPECT is a detector for malicious web pages based on most of the features introduced by our work, plus some new features, such as "social reputation" features extracted by social networking sites. BINSPECT uses multi-model training with confidence-weighted majority voting for the classification of each web page, and spawns a fresh instance of an emulated browser in order to extract features from web pages. The system has very good performances overall, reaching a detection rate of about 97%, with negligible false negatives. The main difference with Prophiler is, however, the fact that its run time ranges between 3 and 5 seconds per page, which renders it unsuitable for the large scale filtering of malicious web pages.

2.4.3 Alternative approaches

Finally, a few other works using completely different approaches for finding and detecting malicious web pages have been proposed in recent times. Zhang et al. have developed ARROW [133], a system that leverages aggregate information on URL redirect chains employed by malware distribution networks in order to generate signatures to block drive-by download attacks. Invernizzi et al. have proposed EvilSeed [47], a system whose aim is to build URL datasets that are likely to contain malicious web pages. EvilSeed starts from an initial seed of known, malicious web pages, and generates search engine queries to identify pages that are related to the original ones, and that are likely to be malicious as well. As the authors claim, such a system can be considered orthogonal to pre-filters like Prophiler, as it tries to create a "highly toxic" URL set instead of starting from a set of normal pages and filtering out the benign ones.

Delta [10], finally, presents a novel static analysis approach for the detection of web infection campaigns. The proposed system extracts change-related features from two version of the same website, visited at different points in time; it derives a model of website changes, and detects if the change is malicious or not by means of machine learning and clustering. Running the Delta system, the authors were able to spot previously unknown web-based infection campaigns. A downside of this approach, compared to Prophiler, is that it would not be able to detect every single drive-by download web page, as it is based on web page changes, and thus if no website infection/content injection is in place (e.g., the malicious page has been setup by the attacker, or the site has not been visited before), no changes can be detected and no comparison with known models can be performed.

These systems all adopt different approaches from the work we present in Chapter 6, but are related to the subject of malicious web page detection, and can be considered complementary to what systems like Prophiler do.

As it can be noticed, detecting and blocking drive-by download attacks has become a very active field of research in the last few years, and many works have proposed solutions to this problem. The solution we propose has been developed specifically for the needs of companies or researchers who need to process large amounts of URLs, and thus need an efficient filter for detecting possibly malicious web pages. In fact, when it is necessary to assess the maliciousness of a web page, the tools of choice are high-interaction honeyclients, as we have described previously in this section (see 2.4.3). High-interaction honeyclients, however, suffer from limited scalability. Our study proposes a fast pre-filtering solution to quickly analyze and discard benign web pages from a given set of URLs. This allows researchers to save a large amount of resources, since the costly back-end analysis is performed only on pages that are deemed as likely to contain malicious content. Second, previous static drive-by download detection systems were limited to using only HTML and JavaScript features to assess the maliciousness of a page [28, 63, 109], or its URL-based features only [30, 67]. We improve the state of the art of static drive-by download detection systems by combining HTML, JavaScript and URL-based features into one single detector, and by significantly extending the set of features used for detection. As we demonstrate in Section 6.4, these new features significantly help improving the accuracy of our system. Third, we test our system on real large-scale datasets and demonstrate it reaches very low false negative rates compared to previous systems, while producing an acceptable amount of false positives, allowing it to be used efficiently as a pre-filtering component for a dynamic analysis system. Finally, our large-scale tests allow us to assess the real-world performances of the system, and demonstrate it can process an average of about 320,000 pages per day on a single machine. This differentiates our work from systems like BINSPECT [25], that further extend the set of features proposed by our system, but have limited scalability compared to Prophiler, even though being more lightweight than traditional high-interaction honeyclients.

Chapter 3

Web Attacks From a Provider's Point of View

This chapter starts our in-depth analysis of each of the four points of view outlined in the beginning of this dissertation. We start our analysis by studying web attacks from the point of view of web hosting providers, for two reasons. First, the majority of websites on the Internet are hosted on shared web hosting plans, and basically every web attack starts with the compromise of a website or hosting account. Second, being the starting point of each attack, if hosting providers were able to detect compromises on their customers' accounts in a precise and timely way, most of today's web attacks would fail or have a very short lifetime. Hence, the importance of studying compromise detection from a provider's point of view.

In this chapter, we test the ability of web hosting providers to detect compromised websites and react to user complaints. Our study covers also six specialized services that provide security monitoring of web pages for a small fee.

During a period of 30 days, we hosted our own vulnerable websites on 22 shared hosting providers, including 12 of the most popular ones. We repeatedly ran five different attacks against each of them. Our tests included a bot-like infection, a drive-by download, the upload of malicious files, an SQL injection stealing credit card numbers, and a phishing kit for a famous American bank. In addition, we also generated traffic from seemingly valid victims of phishing and drive-by download sites. We show that most of these attacks could have been detected by free network or file analysis tools. After 25 days, if no malicious activity was detected, we started to file abuse complaints to the providers. This allowed us to study the reaction of the web hosting providers to both real and bogus complaints.

The general picture we drew from our study is quite alarming. The vast majority of the providers, or "add-on" security monitoring services, are unable to detect the most simple signs of malicious activity on hosted websites.

3.1 Introduction

Owning and operating a website has become a quite common activity in many parts of the world, and millions of websites are operated, every day, for both personal and professional use. People do not need anymore to be computer "gurus" in order to be able to install and run a website: a web browser, a credit card with a few dollars' balance, and some basic computer skills are usually enough to start such an activity.

Of all the possible ways to host a website, shared hosting is usually the most economical option. It consists in having a website hosted on a web server where other websites may reside and share the machine's resources. Thanks to its low price, shared hosting has become the solution of choice for hosting the majority of personal and small business websites all over the world.

Being so common, however, shared hosting websites have also high chances of being targets of web attacks, and become means for criminals to spread malware or host phishing scams. In addition, such websites are often operated by users with little or no security background, who are unlikely to be able to detect attacks or to afford professional security monitoring services.

The analysis presented in this chapter focuses on shared web hosting services, and presents a study on what shared hosting providers do in order to help their customers in detecting when their websites have been compromised. We believe this is an important commitment, given the fact that shared hosting customers are the most vulnerable to web attacks [58]. Furthermore, even a security-aware shared hosting customer would never be able to fully protect and monitor his or her account without the provider's cooperation. In fact, in a shared hosting configuration, the user has few privileges on the machine and she is not allowed to to run or install any monitoring or IDS application, nor to customize the machine's web server, its firewall, or security settings. Thus, in order to protect his or her website, a user has to fully rely on the security measures employed by the hosting provider.

We also tested the providers' reactions to abuse complaints, and the attack detection capabilities of six specialized services providing security monitoring of websites for a small fee.

In a recent survey [16], Commtouch and the StopBadware organization reported the results of a questionnaire in which 600 owners of compromised websites have been asked some questions about the attacks that targeted their websites. From this study, it emerged that, among the surveyed users, 49% of them were made aware of the compromise by a browser warning, while in fewer cases they were notified by their hosting provider (7%) or by a security organization (10%). Also, 14% of the users who took the survey said their hosting provider removed the malicious content from their website after the infection. At the end, only 12% of the customers were satisfied from the way their hosting provider handled the situation, while 28% of users who took the survey were considering to move to a new provider because of this experience.

Inspired by the StopBadware report, we decided to systematically analyze, on a wider scale and in an automated way, how web hosting companies behave with regard to the detection of compromised websites, what their reactions are in case of abuse complaints, and how they proceed to inform a customer about his website being compromised.

This chapter presents the first world-scale analysis of the quality and reliability of security monitoring activities performed by web hosting providers to detect compromised customer websites. Unfortunately, the general picture we drew from our results is quite alarming: the vast majority of providers and "add-on" security monitoring services are unable to detect the most simple signs of malicious activity on hosted websites. It is important to note that we do not want to blame such providers for not protecting their customers, since this service is often not part of the contract for which users are paying for. However, we believe it would be in the interest of the providers and of the general public to implement simple detection mechanisms to promptly identify when a website has been compromised and it is used to perform malicious activities.

3.2 Setup and Deployment

For our study, we selected a total of 22 hosting providers, chosen among the world's top providers in 2011 and 2012 (we will refer to these as *global-1* to *global-12*), and among other regional providers operating in different countries (referred to as *regional-1* to *regional-10*). We selected the *global* providers by picking the ones appearing most frequently on lists of top shared hosting providers published on web hosting-related websites, e.g., `tophosts.com`, `webhosting.info`, and `webhostingreviews.com`. The *regional* providers were instead chosen from the "Country-wise Top hosts" list published by the `webhosting.info` website [125], with the aim of having an approximately uniform geographical distribution over every area of the world. Our final list included providers in the US, Europe, India, Russia, Algeria, Hong Kong, Argentina and Indonesia.

For our study, we limited our choice to providers that allowed international registrations, as our hosting accounts were registered using real personal data of people belonging to our research group. In fact, we noticed that some providers, probably because of regulations in their country, limit the possibility of registering a web hosting service only to national customers. This is especially true for countries such as China, Brazil, and Vietnam, whose providers often require a national ID card number upon registration.

Also, our choice was limited to providers offering shared hosting services as part of their products, allowing to host at least one domain name per account, supporting the PHP programming language, and the FTP transfer protocol.

3.2.1 Test Cases

We conducted our study by registering five shared hosting accounts for each of the 22 web hosting providers. Each one of the five accounts was targeting a particular class of threat, chosen among the most common types of web attacks that could be easily detected by hosting providers.

Four out of the five test cases we deployed are based on a static snapshot of a website running OsCommerce v.2.2. The application was modified so that the PHP pages always returned a static version of the site, without the need of installing a back-end database. Each snapshot was modified by hand in order to include the ad-hoc code required for our experiments, and to diversify the content, the appearance, and the images shown in each page.

Our test files were deployed in the `/osco` sub directory of every hosting account we registered, while the home page of each domain showed only an empty page with the message "Coming soon...". We did not create any link to the `/osco` sub directory, and we excluded the possibility for well behaving web spiders to visit our test case websites by denying any robot access using the `robots.txt` file. This was done in order to avoid external visits to our test case websites, which could have interfered with our tests.

Intentionally installing and exploiting vulnerable web applications on shared hosting accounts may raise some ethical and legal concerns. For this reason, we carefully designed our tests to resemble real compromised websites - being at the same time completely harmless for both the provider and other Internet users. For example, we modified the application code to mimic an existing vulnerability but, compared to their real counterparts, our code was executed only when an additional POST parameter contained a password that we hard-coded in the application, thus allowing only us to exploit the bug.

SQL Injection and Data Exfiltration (SQLi)

The first test case aimed at detecting whether web hosting providers detect or block SQL injection and data exfiltration attacks against their customers' websites. The test consisted in deploying the static snapshot of OsCommerce including a page that mimics the SQL injection vulnerability presented in CVE-2005-4677.

Setup - The `product_info.php` page was modified to recognize our SQL injection attempts and respond by returning a list of randomly generated credit card numbers along with personal details of fictious people (name, address, email, and MD5 password hash). In order to pass the Luhn test, fake credit card numbers were generated using an online credit card test number generator [29].

Attack - The attack for this test case was run every hour, and consisted of a script mimicking a real SQL injection attack: first, the fake vulnerable page (`product_info.php`) was visited, then a sequence of GET requests were sent to the same page adding different payloads to the `products_id` GET parameter. The first request simulated somebody testing for the presence of SQL injection vul-

nerabilities by setting `products_id=99'`; then, five attack requests were issued to the same page by setting the following payloads for the vulnerable parameter:

```
1 : 99' UNION SELECT null,CONCAT(first_name, ... customers_password)
    ,1,CONCAT(cc_type, ... cc_expiration) FROM customers LIMIT
    1,1/*
2 : 99' UNION ALL SELECT null,CONCAT(first_name, ...
    customers_password),1,CONCAT(cc_type, ... cc_expiration) FROM
    customers LIMIT 2,1/*
3 : 99' UNION S/**/ELECT null,CONCAT(first_name, ...
    customers_password),1,CONCAT(cc_type, ... cc_expiration) FROM
    customers LIMIT 3,1/*
4 : 99' UNION S/**/ELECT null,CONCAT(first_name, ...
    customers_password),1,CONCAT(cc_type, ... cc_expiration) FR
    /**/OM customers LIMIT 4,1/*
5 : 99' UNION S/**/ELECT null,CO/**/NCAT(first_name, ...
    customers_password),1,CO/**/NCAT(cc_type, ... cc_expiration)
    FR/**/OM customers LI/**/MIT 5,1/*
```

Listing 3.1: Payloads of fake SQL Injection requests

The purpose of these payloads was to detect whether hosting providers employ any blacklist-based approach to detect SQL injection attempts on their customers' websites. Requests in lines 1 and 2 would fail in case the providers employ simple blacklisting rules (blocking any UNION SELECT and UNION ALL SELECT) in URLs. The last three requests would fail only if providers deploy more complex rules that are able to blacklist typical SQL words even in case they are stuffed with comments, or if words like FROM, CONCAT and LIMIT are blacklisted as well.

Remote File Upload (Web Shell) and Code Injection Using Web Shell (SH)

The goal of this test is to understand whether providers detect the upload and usage of a standard PHP shell, automatic file modifications on the customer's account, or the presence of malicious code on the home page of the website. In the test, a fake web shell is uploaded to the hosting account, and fake commands are issued to it, resulting in some drive-by-download code being added to the home page of the e-commerce web application.

Setup - This test uses the base static snapshot of the OsCommerce v.2.2 web application, and simulates a Remote File Upload vulnerability in the file *`admin/categories.php/login.php`*, as the one described in [82]. Our fake attack was designed to upload a modified version of the popular c99 PHP shell (one of the most common web shells on the web), that has no harmful effects other than the ability to inject custom code in the home page of the e-commerce web application. Also in this case, the custom code injection is enabled only when certain hidden parameters are specified along with the request of the c99 shell, thus allowing only us to trigger the injection. The content to be injected in OsCommerce's index page is a snippet of a real malicious code launching a drive-by download attack, that has been disabled by wrapping it into an `if` statement with a complex condition

that is always False. We submitted the index page with the injected content to the VirusTotal online virus scanning service [123], and it was detected as malicious by 13 anti virus engines.

Attack - The test case for this attack was run every hour, and consisted in a script performing the upload of the web shell, followed by a number of commands issued on the shell. The shell file, called *c99.php* as the original shell, in order to be easily identifiable from the web server logs, was uploaded to the vulnerable URL by specifying the secret parameter enabling the upload. If the upload was successful, five commands were issued to the *c99.php*, picked randomly from a list of GET and POST requests containing both Unix commands and file names, so to make the requests seem like the result of someone trying to manually explore the contents of the server. The requests simulated actions such as trying to read files (e.g., `/etc/passwd`) and execute unix commands (`who`, `uptime`, `uname`, `ls`, `ps`). Our intuition was that hosting providers would probably be alerted by requests containing some of these file names or commands. Finally, the test used the PHP shell to inject a plain-text version of the malicious code into the home page of OsCommerce.

Remote File Upload of a Phishing Kit (Phish)

Similarly to the previous test, this test uses a file upload vulnerability in the OsCommerce application to upload a phishing kit to the web server. The phishing kit consists of an archive containing a static snapshot of a real Bank of America scam. The test aims at detecting whether hosting providers are able to detect the presence of a phishing kit on the customer's account. The phishing kit was installed inside a directory named `/bankofamerica.com`, thus allowing to detect any visit to the scam pages by simply looking at the requested URLs.

Setup - This copy of the application is configured with the same Remote File Upload vulnerability explained for the previous test. However, the vulnerable path for this test is *`admin/banner_manager.php/login.php`*. Whenever this script is issued an upload request for a file with `tar` extension, it uploads the archive and automatically unpacks its contents to the upload directory, thus allowing for an automatic installation of the phishing kit. The phishing kit we deployed is an exact copy of a real Bank of America phishing kit found in the wild, modified to remove the back end code (thus making it unable to store and send any user information).

Attack - This attack was split in two phases, which we refer to as *attacker* and *victim*. The *attacker* phase, run every 6 hours, consisted in triggering the remote file upload vulnerability and uploading the phishing kit. The *victim* phase of the attack was run four times per hour, and consisted in a script that simulated a victim falling prey of the scam. In order to look realistic, the victim requests were disguised as coming from a range of different valid User-Agent strings used by web browsers on Windows operating systems. Every simulated victim visit comprised a sequence of GET and POST requests containing the form parameters required by

the phishing pages. At each victim visit, the data sent in the requests was randomly picked among a set of fake personal details we created by hand, containing names, addresses, passwords and credit card numbers of fictitious people.

Suspicious Network Activity: IRC Bot (Bot)

This test aims at understanding whether providers employ any network rules to detect suspicious connection attempts to possibly malicious services. For this study, we opted to deploy to our accounts a script simulating an IRC bot. The reason for this choice is that IRC bots are probably one of the most common and easily detectable bots, because IRC connections are very often made to the standard IRC port (6667) using clear-text communication.

Setup - This test uses our basic OsCommerce installation with no modifications. The executable bot client was deployed to the hosting account via FTP, thus simulating an attack in which the attacker has stolen the customer's web hosting credentials. The files to be uploaded are two IRC client binaries written in C (one compiled for 32-bit architectures, and one for 64-bit ones), and a PHP script that executes the right binary depending on the underlying OS type, and outputs its results. The IRC client, once launched, disguises itself as "syslogd" and tries to connect to a machine hosted on our premises that runs a fake IRC server on the standard IRC port. If the connection succeeds, the client and server exchange a few messages resembling real IRC commands (such as `NICK xxx`, `USER xxx`, `JOIN #channel`) and the client reports some information about the infected machine (host name, OS type, kernel version); at last, the client closes the connection.

Attack - The test case for Bot was run every hour, and started with opening a FTP connection and uploading the two binaries and the PHP file in a new directory created in the web site's root folder. If the upload succeeded, an HTTP request was issued to the PHP file launching the IRC client. The output of this request allowed us to determine whether the hosting provider was blocking the use of possibly dangerous PHP functions (IRC client execution denied - *system()* function disabled), blocking outgoing connections to certain ports (binary executed, but connection attempt failed), or allowing everything (successful connection to the server). In order to make the upload of the IRC botnet files appear even more suspicious, the FTP upload was executed using IP addresses from several different countries.

Known Malicious Files (AV)

This test aimed at understanding whether providers perform any scans of their disks with off-the-shelf anti virus software. The test simply consisted in deploying, via FTP, two common known malicious files to the customer's hosting account.

Setup - Websites hosting this test used a simpler structure than the previous tests, and consisted in a single static HTML page containing random sentences in English and a few images. As in test Bot, we chose to use FTP to upload the ma-

licious files to the account, to simulate a case in which the attacker has knowledge of the customer's account credentials. The two malicious files were *c99.php*, a real c99 PHP web shell, detected on VirusTotal with a score of 25/43 (25 antivirus engines detecting it, out of 43 it was tested against), and *sb.exe*, a copy of the 2011 Ramnit worm, detected by 36 out of 42 antivirus products according to VirusTotal. In order to make sure the malicious files were not reachable by any web visitor, but only available to people having internal access to the server, they were uploaded to a directory protected by means of *.htaccess* (denying the listing of its files) and *.htpasswd* (requiring a password to access its files from the web).

Attack - The attack itself consisted simply in connecting to the hosting account's web space via FTP and uploading every time (deleting and re-uploading if already present) the protected directory and the two malicious files. Also in this case, FTP connections were issued from IP addresses in different countries.

3.2.2 Attack Detection Using State-of-the-Art Tools

Before deploying the tests to the shared hosting accounts, we made sure they could be detected using common state-of-the art tools, that can be easily employed by any hosting provider. In order to do so, we executed our tests against an installation of the SecurityOnion Linux distribution, which includes a pre-configured set of open source tools for monitoring suspicious network and system activity (Bro IDS, Snort, Sguil). The installation of this distribution was then equipped with the Apache2 web server and the ModSecurity plugin, along with its base recommended rule set.

We also installed the OWASP ModSecurity "Core Rule Set", a set of common security rules for Apache ModSecurity that is maintained by the OWASP foundation [87]. These are free certified rule sets providing generic protection from unknown vulnerabilities often found in web applications. We installed version 2.2.5 of the rule set on our test machine, and disabled some rule sets (base rules number 21, 23, 30) for being too generic and generating too many false alarms. We finally ran each of the five test cases toggling on and off the OWASP ModSecurity rules.

Table 3.1 summarizes what we were able to detect or block using this setup, during the execution of each test. Four out of the five attacks would have been blocked or detected by employing free network and host monitoring solutions like the ones mentioned above, and the remaining attack could have been easily detected by setting up a simple connection filtering rule in the firewall.

SQLi

The attacks of test SQLi, when run using the basic installation of ModSecurity, succeed, but generate a series of five different high severity alerts about possible

Test #	SQLi	SH	Phish	Bot	AV
Blocked by ModSecurity base rule set	○	○	○	○	-
Blocked by ModSecurity OWASP rule set	●	◐	○	○	-
High severity IDS alerts	5	2	2	0	0
Detectable by antiviruses	no	yes	no	no	yes

Table 3.1: Attacks detection using freely available state-of-the-art security scanning tools. Legend:
○ no; ◐ in part; ● yes (full); - not applicable

web server SQL injection attempts. When the OWASP rule set is enabled, however, all the five SQL injection attempts on which the attack is built fail.

SH

The SH test, executed against a webserver with the basic ModSecurity rules, successfully uploads the c99 shell and injects the drive-by code in *index.php*. However, two high severity events are raised by the IDS, one of which notifying a remote code execution on OsCommerce v.2.2 (triggered by our attack to upload of the web shell). If the OWASP rules are enabled, the remote file upload succeeds but most of the commands issued to the web shell fail and raise critical alert messages, notifying the possibility of a web file injection attack. The index file modification, finally, fails and raises a message notifying the detection of multiple URL encodings in the request, as a possible sign of protocol evasion. Finally, it has to be noted that, although we removed all the existing functionalities from the original web shell, our *c99.php* contains some original PHP code to display images and UI elements, plus our custom drive-by injection code. As such, it would still be detected during a virus scan by approximately 17% of the antivirus engines on the market (its VirusTotal score is 7/42). The *index.php* containing the injected content would instead be detected by almost 30% of the antiviruses, having a VirusTotal detection score of 13/44.

Phish

This attack succeeded but raised two high severity events: potential remote code execution in OsCommerce v.2.2, and presence of PHP tags in the HTTP post (detected on the tar file containing the phishing kit). On the victim's side, no HTTP request is blocked when uploading personal information to the scam pages. A possible solution to stop, or at least raise alerts on the victim's requests, however, could be deploying a simple IDS/IPS rule that detects the submission of parameters containing clear-text personal details, such as credit card numbers and cvv2 codes.

Bot

The Bot test case was undetected by the basic and OWASP ModSecurity rule sets, as it was run via FTP. In our tests, the connection succeeded and the bot and fake IRC server completed their message exchange. A normal firewall rule blocking outgoing connections to port 6667 (IRC) would have, however, blocked the attack.

AV

The malware upload test (AV) was undetected by our test deployment, because no HTTP traffic was generated and no network antivirus was used. However, as explained in Section 3.2.1, we recall that the uploaded *c99.php* and *sb.exe* are common malicious files detected by VirusTotal with a detection scores of 25/42 and 36/42, respectively. Therefore, the vast majority of off-the-shelf antiviruses would have detected them during a scan of the website's root directory.

3.2.3 Test Scheduling and Provider Solicitation

All attacks were run without interruption on every hosting account for the first 25 days of testing. As explained in the previous section, each attack was repeated multiple times per day in order to generate more alerts and increase the probability of being detected.

If the hosting provider did not detect any suspicious activity during this time frame, the tests entered a second phase, during which we solicited the provider to detect our attacks and take action against them. This solicitation took place as an abuse notification email for the Phish and AV tests, in which we reported the presence of malicious files on the web application. We also generated "fake" abuse notifications to study the reaction of the providers to bogus complains.

This allowed us to understand: 1) how quickly providers respond to abuse notifications, if they ever do, 2) if they actually verify the presence of malicious content or activity on the account before taking any action, and 3) what kind of actions they take in order to stop the abuse. Abuse notifications were sent to providers by email, using real (authenticated) email addresses registered on 3rd party domains, to make them look as realistic notifications from random web users.

Real Abuse Notifications

Starting the 25th day of testing, we started sending one abuse complaint per day to each provider on which tests Phish and AV had not been previously detected. We stopped the notification process and the real attacks on the account either when the 30 day testing period elapsed, or after the provider responded to the notification. The notification email explained that an email had been received, with a link pointing to content hosted on the provider's premises. The link pointed to the phishing kit's index page for Phish, and to the *sb.exe* file for AV test. In addition, the email

mentioned that the user's antivirus raised an alert when trying to visit the URL, and suggested the web provider to check the contents of the account.

Fake Abuse Notifications

Apart from real abuse notifications, we also sent emails in which we complained for perfectly clean websites. To perform this test, we cleaned and re-used the account used for the SQLi and Bot tests. The website contents were replaced by a single static HTML page containing one JPG picture and a long list of news extracted from the RSS feeds of popular international news websites. Starting on the 25th day, we sent to every provider an email per day, where the user complained about the presence of offending or malicious content on these accounts. Since at the time these emails were sent the websites were absolutely clean, these fake notifications allowed us to understand whether providers actually check the veracity of the complaints they receive before taking any action. The first complaint email was from a user pretending that the website's content was offending his religious views, and kindly asking to stop the website owner from spreading such disrespectful messages. In the second scenario, the notification email was from a user claiming to have received an email with a link to the website in question. The user explained that his browser denied access to the URL, and that at a closer look the website looked like hosting a phishing scam. Also in this case, the account hosting the reported webpage was absolutely clean, hosting only the benign static HTML home page.

One may argue that, in case of these fake notifications, the provider could react by suspending or shutting down the user account by having a look at the logs of the machine on which the account was setup, and noticing past malicious activity, even though, at notification time, the website was clean. We did our best in order to avoid this from happening, by deploying our tests for fake notifications on accounts that hosted the SQLi and Bot tests. These tests could not be considered malicious (no malware nor phishing files were ever uploaded) but the mere evidence that the website was under attack. Moreover, attacks for these tests could only have been detected at a network level, since no trace was left on the disk.

3.3 Evaluation

During our experiments, we evaluated the security measures put in place by web hosting providers to detect malicious activities, compromised websites, and prevent abuse of their services. We group our findings in three categories: account verification upon sign-up (3.3.1), compromise prevention and detection capabilities on live websites (3.3.2), and responses to abuse notifications (3.3.3).

3.3.1 Sign-up Restrictions and Security Measures

Even though our work was not meant to test the anti-abuse sign-up policies of web hosting providers, we report here some results that may contribute in understanding how much effort providers put in preventing services subscription by malicious users.

Several providers try to discourage abusers by asking to verify the information entered during the sign-up phase, either by calling the customers on the phone, or by requiring a scanned copy of their documents (such as government issued ID, credit card used for the purchase). Some providers also use 3rd party fraud protection services, that block purchases based on a set of heuristics. For example, we observed several cases in which the providers correlated the geographic location of the customer, the billing information, and the IP address used for the purchase.

The shared hosting accounts we used for our study were all registered using real personal information of people working in our group, and the billing information of our research institute. The sign-up process was carried out from several IP addresses, using either credit card or PayPal payments.

Anti-abuse sign-up policies vary widely between hosting providers. Top *global* hosting providers are more cautious with regard to sign-up, often blocking attempts - e.g., blocking multiple registrations from the same billing address and credit card number, verifying the customer's personal information by verification phone calls or ID and credit card checks. *Regional* providers seem to be more permissive, probably because they have less incentives in making their sign-up process more difficult, which could make them lose potential customers.

Among the twelve *global* providers, seven of them required us to verify our account information for at least one of the accounts we registered with them. In order to verify our account information, all these companies required a scanned version or photocopy of a government issued photo identification card (such as passport or driver's license) and the front and back of the credit card used at sign-up (without showing the first 12 numbers and the cvv2 code). Only one out of these seven companies claimed, on its website, to manually verify every customer's sign-up before allowing the purchase of its web hosting services. Indeed, this was the only provider that verified every account we registered with them.

Regional providers, instead, do not seem to be as cautious during the account sign-up phase. Only one out of ten blocked an account creation because of a mismatch between our billing address and the geolocation of the IP address used for registration.

Finally, three of the *regional* providers we tested had a very simple sign-up process, where users could register an account in one click, by filling all the required personal and payment information in one page. These providers never asked us to verify our information upon registration, and thus could possibly be a good choice for criminals wanting to perform abusive subscriptions.

Provider	Verification time		
	Before payment	**Before activation**	**After activation**
global-2	25%	-	50%
global-3	25%	-	25%
global-4	33%	-	-
global-5	40%	-	-
global-6	-	33%	-
global-7	100%	-	-
global-8	50%	25%	-
regional-2	33%	-	-

Table 3.2: Account verification times. Values represent the percentage of verification requests on the number of accounts we registered for each provider. "Before payment" means during the registration process. "Before activation" means once the client's billing account is created, but the hosting service is not yet active. "After activation" indicates when the hosting account is active and a website has possibly already been installed.

Sign-up verification requests are either sent during registration or after a successful account registration and activation. While requiring an account verification upon sign-up can be effective in preventing malicious registrations, it can also make the hosting provider lose potential good customers that may not have time or patience to provide all the required information. On the other hand, requiring an account verification once the service has been purchased and set up has the drawback of temporarily suspending an account on which a website has already possibly been deployed, thus causing a service outage for a benign customer. During our experiments we encountered both situations. Table 3.2 shows the percentage of verification requests on the number of accounts we registered for each provider, grouped by the time at which the request was issued. Only providers that requested at least one account verification are listed.

The table shows that, in general, most of the anti-abuse systems send alerts and block a registration attempt during the customer's sign-up phase. This typically happens when the user enters his or her credit card details and tries to complete the hosting purchase. Others, instead, let the client sign up for the service and receive its management panel credentials, but lock the web hosting service activation until a copy of the customer's document is received by the support department. Two web hosting providers (*global-2,3*) sent verification requests when the web hosting account was already active and the customer's website deployed. This caused a temporary service disruption for the affected accounts, making their websites unavailable for several hours. Certain providers, finally, issued verification requests at different times, probably depending on the kind of alert they received from their abuse prevention system (*global-2,3,8*).

3.3.2 Attack and Compromise Detection

During the first phase of our experiments, we deployed our five test suites on every hosting provider and recorded whether the hosting provider took some action or contacted us to notify that malicious activity was observed on our account. As explained in Section 3.2.3, if no malicious activity was detected on the account during the first 25 days, we started sending abuse notifications to the hosting provider, in order to stimulate a response. The results of this second phase are summarized in Section 3.3.3.

To make our fake attacks look realistic, our test cases were run automatically at certain time intervals (as explained in Section 3.2.1), and the attacks were executed from different IP addresses belonging to several different countries. Also, in order to avoid having only "artificial" malicious requests in the web server logs of our accounts, we generated some background traffic simulating real visits to our websites. This was accomplished by developing a simple traffic generator tool, that visited every account we deployed every 10 minutes, and randomly followed links on every website up to a depth of 30. In the general case, this meant following an average of 13 links on every website, thus generating a bit less than two thousand hits per day on every active account. The machine used for traffic generation was not used for other experiments and used a different set of public IP addresses than the ones we used to run the attacks.

Attack Prevention

Even though our study focuses on the ability of the providers to detect compromised websites, during our experiments some of our attacks were blocked and were therefore ineffective. In some of these cases, we proceeded by manually compromising the account. For example, whenever a provider denied the possibility of running test SH, we manually uploaded the drive-by download code to the account to continue the experiment. This allowed us to test whether the provider was able to detect the menace by scanning the customer's account. For the phishing attack (Phish), since it had to be detected on a network level, we did not take such measure and thus no manual upload was performed on accounts of providers blocking the remote file upload.

Table 3.3 reports, for each test and provider, whether the web hosting company took any measure to prevent the attack. Such measures depend on the test case, and ranged from employing URL blacklists to blocking outgoing connections or process executions.

URL blacklisting Some providers employ URL blacklists in order to prevent SQL injection attempts (test SQLi) and remote file uploads (SH, Phish).

However, as shown in column *SQLi* of Table 3.3, none of the providers we tested were able to fully prevent our SQL injection attacks. This is probably due to the adoption of simple keyword-based blacklisting rules, that can be easily by-

passed by introducing SQL comments in the middle of blacklisted keywords (such as using "SE/**/LECT" instead of "SELECT", as explained in Section 3.2.1). Two providers (*global-1, regional-2*) blocked the first four requests of our attacks, and other five providers were able to block only the first two. The remaining did not adopt any SQL-injection protection mechanism at all.

Regarding tests SH and Phish, some providers were able to prevent the attack by employing URL blacklists probably containing specific rules for the detection of common vulnerabilities on web applications, such as the ones we employed for the tests presented in Section 3.2.2, provided by the OWASP foundation. Regarding SH, Table 3.3 shows that some providers were able to only partially prevent the attack. These providers did not block the file upload itself, but employed blacklisting rules to block some requests to the web shell (these requests contained common file names, e.g., */etc/passwd*, or common parameter sequences such as *.php?act=cmd*).

Connection and OS-level filtering The attack files for test Bot were first uploaded to the customer's account via FTP, then the fake IRC client was executed issuing a HTTP request to a PHP file launching an executable file using the *system()* PHP function. A total of 18 providers were able to fully stop the attack: of these, 50% did so by disabling the *system()* function in PHP, while the remaining half firewalled outgoing connections to the IRC port.

When the attack was prevented, we were expecting some form of notification regarding the suspicious activity. After all, it is not normal that a shared hosting user has a disguised process that tries to connect to an IRC server every hour for one month.

Two hosting providers allowed the attack only at certain periods in time (*global-2* and *global-6*). This may due to temporary misconfigurations on their networks or to automatic account migrations over different machines with different configurations (for example, the account running test Bot on provider *global-6* connected to our fake IRC server from eight different hosts during the 25 days testing period).

No prevention results are shown for test AV, as this test did not run any attack and no filtering was done on the upload of malicious files via FTP.

As a final remark, we noticed that, for some tests, some providers had exactly the same behavior. This is the case, for example, of *global-1* and *regional-2* and *global-8* and *regional-3*. We thus believe that these providers employ the same protection mechanisms and web server security configurations for their shared web hosting solutions. These services are probably provided by third party companies as part of common off-the-shelf security solutions.

Compromise Detection

Sadly, all but one of the providers we tested did not notify their clients when their websites were compromised and were used to perpetrate obvious malicious activities.

Provider	Account verification	Attack Prevention/Detection (days)					Solicitation Reaction		
		SQLi	SH	Phish	Bot	AV	Abuse complaint	Fake abuse complaint	Avg. reply delay (days)
global-1	○	◐/ ○	●/ ○	●/ -	●/ ○	- / ○	○ N	● N	-
global-2	◐	○/ ○	○/ ○	○/ ○	◐/ ○	- / ○	○ T	- -	1
global-3	◐	- / -	○/ ○	○/ ○	●/ ○	- / ○	○ N/T	- -	-
global-4	◐	○/ ○	○/ ○	○/ ○	●/ ○	- / ●(17)	● S	◐ U	0
global-5	◐	- / -	○/ ○	○/ ○	●/ ○	- / ○	○ T	- -	0
global-6	◐	○/ ○	○/ ○	○/ ○	◐/ ○	- / ○	○ U	● O	2
global-7	●	◐/ ○	○/ ○	○/ ○	●/ ○	- / ○	○ N	● N	-
global-8	◐	◐/ ○	○/ ○	●/ -	●/ ○	- / ○	○ N	● N	-
global-9	○	○/ ○	●/ ○	●/ -	●/ ○	- / ○	○ N	● N	-
global-10	○	○/ ○	●/ ○	●/ -	●/ ○	- / ○	◐ S	● N	4
global-11	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	○ N	● N	-
global-12	○	○/ ○	○/ ○	○/ ○	○/ ○	- / ○	◐ T,C	● O	0
regional-1	○	◐/ ○	◐/ ○	○/ ○	●/ ○	- / ○	● S,C	○ S	0
regional-2	◐	◐/ ○	●/ ○	●/ -	●/ ○	- / ○	○ N	● N	-
regional-3	○	◐/ ○	○/ ○	●/ -	●/ ○	- / ○	◐ O,C	● O	0
regional-4	○	○/ ○	○/ ○	○/ ○	○/ ○	- / ○	○ N	● N	-
regional-5	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	◐ S	● O	16
regional-6	○	◐/ ○	◐/ ○	○/ ○	●/ ○	- / ○	◐ C	○ C	1
regional-7	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	○ N	◐ U	5
regional-8	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	● S,F	● O	1
regional-9	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	○ N	● N	-
regional-10	○	○/ ○	○/ ○	○/ ○	●/ ○	- / ○	○ N	○ P	0

Table 3.3: The results of our study. Legend:

-	not applicable	N	no reply	P	forced password reset
○	no / not satisfying	S	account suspension	C	cleanup or file removal
◐	in part / partly satisfying	T	account termination	U	ultimatum to the user
●	yes (full) / satisfying	F	complaint email forwarded	O	reply but no action

The only hosting provider that reacted to one of the attacks was *global-4*, but that reaction happened 17 days after the beginning of test AV. The provider properly notified the presence of a malicious file (the `c99` shell) on the user's web hosting account. In addition, the provider warned the user that a service suspension would occur if no reply to the alert was received by the customer support service within 24 hours. However, the message was not mentioning the presence of the other malicious file on the account, namely, `sb.exe`. This suggests that the alert was an automated message resulting from a virus scan of the account, and that no human operator actually checked the contents of the directory in which the two malicious files were stored.

We were quite surprised by our findings, as we were expecting to have at least a few of our scenarios detected by the vast majority of web hosting providers. It emerges that, on shared hosting servers, even the most basic virus scan is not as common as one could expect. From our measurements, we are not able to tell if the hosting providers run antivirus systems on their shared hosting servers. However, if they do, they are either using outdated signature definitions, or the frequency at which they perform the scans is less than once a month.

3.3.3 Solicitation Reactions

As explained in detail in Section 3.2.3, whenever one of our test suites was not detected by the hosting provider for 25 consecutive days, we started sending daily abuse notification emails to the provider's abuse contact. The purpose of sending these messages was to understand whether web hosting providers respond and react to abuse notifications (e.g., by suspending a compromised account or notifying the customer of his or her website being compromised). To complete our test, we also sent fake abuse notifications for perfectly clean webpages, with the aim of understanding whether any providers take action without first verifying the claims. This would pose a serious menace, as it would be a very easy and effective way to conduct a Denial of Service attack against websites of other users. The following paragraphs are meant to give some insights and details on what is presented in the "Solicitation Reaction" section of Table 3.3.

Abuse Notifications

Unfortunately, 50% of both the *global* and *regional* web hosting providers never replied to any of the real abuse notifications we sent. This percentage is quite alarming, and means that if a website is hosting malicious content (such as phishing or malware), no action will be taken to stop it from spreading and reaching its victims. Moreover, phishing attacks and malware files used in dropzones usually have a short lifetime, and, as such, even a late response to a malware or phishing abuse notification would have little or no effect on the general outcome of the attack.

Seven out of the eleven providers that replied to our complaints replied either the same day or the day after the notification was sent. This is a good indicator, meaning that these companies probably care about web abuses and are able to handle these issues in a timely manner. The only provider that replied later than 5 days after the notification was *regional-5*, with an average response time of 16 days. After such a long delay any action would be basically useless, as the website may have completely changed in the meantime.

There were a variety of reactions to our abuse complaints. The most common approach was to temporary suspend the customer's account, with five companies performing at least one suspension as result of a malware or phishing abuse complaint. We consider this action a reasonable response to the abuse, causing a temporary disruption of the services the client is paying for, but blocking the immediate threat. Other providers responded to the notifications by cleaning up the account, removing the suspicious files (4 providers - note that this action seems to be more common among *regional* providers), or by forwarding the abuse notification to the customer (1 case). We considered such responses, in general, to be appropriate to stop the menaces from spreading, and at the same time avoiding to impact too much the user's services.

Provider *global-12* reacted without notifying the website's owner: in the case of AV, the account was terminated, while in the case of phishing (Phish), the directory containing the fake phishing kit was removed. Also in the case of provider *regional-6*, actions were taken without notifying the user, with the exception that, in this case, the reactions to the abuse notifications consisted in deleting all the files (including the clean ones) of the user's websites!

Controversial responses to our abuse notifications were those from providers that sent ultimatums to the user (marked with U, in the table), warning him that offending content had been found on his website, and that if no cleanup was performed within a few hours, the account would have been suspended. This was controversial because, as in the case of provider *global-6*, even though we did not take any action to respond to the provider ultimatum, the fake phishing pages were still present on our account after several days. This means that the provider did not keep to its commitment.

Finally, a few responses were partially or fully unsatisfying. The *regional-3* provider replied to the malware abuse complaint probably after scanning the customer's account using an antivirus. The reply stated that a c99 PHP shell had been found on the account, and asked the notifier if he wanted them to remove it. The malicious executable was not mentioned at all and no further action was taken, thus leaving both malicious files on the account. The case of providers *global-2, global-3* and *global-5* is quite particular. While experiments were in progress on most of the providers, and once our tests Phish and AV reached their 25th day on *global-2* and *global-5*, notifications were sent to the two providers. First, provider *global-5* replied by terminating the account (disabling both the billing and the hosting account) and giving the customer 15 days to reply and to recover his files. We replied, asking to re-enable the account for recovering our files, but in the meanwhile another abuse response was received from provider *global-2*, terminating our account. Starting that moment, within a few hours, all the accounts we had registered on providers *global-2, global-3* and *global-5* were terminated without any explanation, even when we tried to contact the companies to ask details about the reasons of our accounts' termination. The only response we were able to get was: *"Due to certain items contained in the account information, this account was flagged for fraud. For security reasons, this flag caused the system to delete your account. At this time we ask you to seek out a new hosting company."*

Either the three companies used the same support service, provided by a third party, or they shared information between them. Indeed, the termination notifications for all the accounts on the three providers were sent by the same support representatives, and contained exactly the same text (only the email signature changed, containing the email and postal address of the appropriate company). For this reason, we expect the support center for these companies was able to link our accounts' personal information and understand they were all registered by the same group of individuals. Thus, having received complaints for two of the accounts, all the other accounts that could have been reasonably linked to them were terminated as well.

When this happened, some test cases had not been deployed yet on these providers (SQLi on *global-3, global-5*) and others had not yet reached their 25th day of execution (Phish on all, and SQLi on *global-2*), thus no fake abuse notifications were sent for them. This explains why Table 3.3 has missing data for such providers in columns "SQLi" and "Fake abuse complaint". This is also why in the "Abuse complaint" cell for provider *global-3*, we listed N/T: no abuse notification response was received (N), but a termination occurred anyway (T) for other reasons.

Finally, for provider *global-9*, we were not able to properly contact its abuse department: out of the four different abuse notifications we sent to its abuse email address, only the last two received an automated reply, saying that in order to report an abuse, it is necessary to click on the help link on the web hosting provider's home page and follow a series of steps (at the time we received these responses, the five-days testing period was already expired). We flagged this case as "no reply" because, although we tried to submit the complaints following the company's advice, the user interface adopted by the provider makes it very difficult, even for an experienced user, to find the right way to report a website abuse. Moreover, once a visitor is able to reach the right page for submitting a website abuse notification, he or she is required to register an account before being able to file a complaint.

Fake Abuse Notifications

We expected most web hosting providers to ignore our abuse notifications regarding "offending content" (see 3.2.3) and to check the website's contents but take no action in case of the fake phishing complaints. In Table 3.3, we thus marked as "satisfying" also the providers that never replied to our complaints. However, this is not always a good sign, especially when the same provider never responded to the real complaints.

Sadly, some of the reactions we observed were clearly in contrast with our expectations. Both providers marked with "U" believed either our religious complaint (*global-4*) or our phishing one (*regional-7*), warning the website owner about the possibility for his account to be suspended if the offending content was not removed within a few days. However, contrarily to what was promised, the content of the websites was left untouched and none of these providers took any action to block the user's account after the ultimatum expired.

One provider, *regional-1*, suspended one of our clean accounts on the same day it was notified as hosting a phishing website. *regional-6*, instead, acted as in the case of real abuse complaints: all the pages on the account's web hosting directory were deleted, and the website's home page was replaced by an "under construction" page. This was already bad when associated to a real malicious content, but in case of a bogus complaint it is really an unacceptable behavior. One last provider, then, responded to the fake phishing abuse notification by sending the website owner an email stating that his website has been attacked, and as such a password reset had been forced on the account. Furthermore, the malicious files were disabled (by means of changing their access permissions) and their list was sent to the user: the

list contained the benign website home page and the `jpeg` picture included in it. We were not able to figure out how the web hosting provider assumed the static HTML home page and the picture could contain malicious code.

Only four web hosting providers replied to our fake abuse notifications with messages that completely satisfied our expectations. In these cases, marked with "O" in the table, the support representative informed the notifier that upon manual inspection, the website seemed to be clean, and, in case some content seems to be offending somebody's cultural views, the issue has to be resolved in person by contacting the owner of the website. From this analysis it seems that *regional* providers are slightly more likely to perform a manual content inspection on the websites they host (at least 30% of the ones we tested), compared to *global* providers (only two out of twelve).

3.3.4 Re-Activation Policies

Whenever an hosting account was suspended, providers often provide the customer with the steps to follow in order to have the account re-activated. These steps usually imply changing every password of the account (billing, FTP, database passwords, etc.), writing a letter or an email stating the agreement to the provider's Terms of Service, and removing the malicious files or re-installing a clean copy of the website. Among the companies that suspended our accounts, *global* hosting providers seem to stick to strict legal requirements before allowing customers to have their accounts re-activated after a violation of the terms of service. The two hosting providers that suspended at least one of our accounts required us to send an email (*global-4*) or a scanned letter or fax (*global-10*) to their support department, stating that we have followed all the necessary steps to clean up our account and reset our login credentials, and that in future we will abide by the terms of service of the company. Regional providers appear to be more "informal" with regard to this, as often a simple email replying to the incident notification, explaining that we were running a vulnerable web application or using a weak FTP password, was sufficient to have our account re-activated. Also *regional* providers, however, in their incident notifications, advised the user to follow basic steps to secure his account (password change, website cleanup) before requesting a service re-activation. During our tests on *regional-1*, in one case, a scanned version of the customer's identification card was required in order to re-activate a suspended account.

Finally, in the case of service terminations, the providers just wanted the user to leave their company, replying to service re-activation requests with emails stating in that, given the kind of activity encountered on the account, the company was not willing anymore to provide their service to such customers.

3.3.5 Security Add-on Services

In our study, we also evaluated the ability of third party "add-on security providers" to detect attacks or abuses on a website. These services can be purchased separately from web hosting accounts *, and associated with a domain or website to monitor. In some cases, the subscriber has even the option to give his FTP/SFTP access credentials to the security service, to allow an in-depth scan of all the files on his or her account (also those that may not be reachable from the web). For our study, we selected four companies offering such security services, chosen among the most common and advertised on the web. We limited our choice to services that are affordable for a personal or small business use ($30/month max subscription price). We did so in order to test services that are in line with the level of web hosting we were testing. Indeed, it would not be reasonable to pay hundreds of dollars per month, or more, to protect a $10/month hosting plan.

Some of the add-on companies we evaluated are proposing several level of service, at different pricing. We thus registered every protection level available, up to the $30/month threshold we had fixed, ending up registering a total of six security add-on services (two each from the companies offering multiple levels of protection). Six additional hosting accounts were purchased, from different companies, in order to accommodate our tests for these security services. In the following, we refer to them as *sec-1* through *sec-4*. The two variants for companies offering different levels of protection are labeled with a *-basic* or *-pro* suffix, to distinguish, respectively, the cheapest version of the service from the more expensive one. Services in the *-pro* version, for both providers *sec-1* and *sec-2*, allow to scan, daily, all the files on the customer's FTP hosting account, if they are provided with his or her access credentials. We configured both services to enable this kind of scans. The other four security services, contrarily, perform only scans on publicly accessible pages of the websites they are configured to monitor. Such scans include, in most of the cases, checking for malware, malicious links, blacklisted pages, and performing reputation checks on both the website and the provider hosting its contents.

Evaluation of the Security Services

The security services' evaluation schedule was tighter than the normal test evaluation schedule, as we expected security add-on services to react faster to attacks and suspicious account activities, being specially designed for detecting security issues. Thus, the tests on accounts hosting the security add-on services were run for a total duration of 50 days, 10 days for each test, from SH to AV. The SQL injection test was not run on such web hosting accounts, because its attack does

∗. Although these services can be purchased separately, several web hosting providers offer security services from third party companies at a discounted price, if purchased in conjunction with a web hosting plan.

Provider	Attack Detection				
	SH	Phish	Bot	AV	SH-BL
sec-1-basic	○	○	○	○	○
sec-1-pro	○	○	○	◐	●
sec-2-basic	○	○	○	○	○
sec-2-pro	○	○	○	○	○
sec-3	○	○	○	○	○
sec-4	○	○	○	○	○

Table 3.4: Results of our evaluation of third party security services. Symbols and their meanings are the same as in Table 3.3.

not generate any side effect on the hosting account and thus could not be detected by third party external security services.

We noticed that two of the companies providing the add-on security services are listed among the partners of known URL blacklisting services. We therefore used the last 10 days of testing to study reactions to the notification of suspicious URLs to such blacklists. For this, we scheduled a last test consisting in a new deployment of SH, along with the submission of its drive-by download page to a few malicious URL reporting and blacklisting services. The URL blacklisting requests were sent on the same day the tests were deployed. We refer to this test as "SH-BL".

Results are shown in Table 3.4. One can see that detection capabilities for add-on services are comparable to those of providers. However, in this case, customers pay for a service whose only commitment should be monitoring a website in search of potential vulnerabilities or malicious content. Almost all the services we tested in this part of our study seem to completely fail this objective.

All the services were configured to send notifications to the user whenever a security issue was detected on the monitored website. None of the add-on security services detected anything anomalous during our tests SH, Phish, Bot (attacks were all successful and never blocked by the hosting provider). Test AV was not detected either, but the *sec-1-pro* service raised a warning for having detected the c99 web shell on our hosting account. However, this alert was visible only when logged on the security service's web management panel, where the c99.php file was listed as suspicious. No critical alerts were issued, nor any email was sent to the user as notification for this event. Finally, the only successful detection was performed by the *sec-1-pro* service, detecting our drive-by download page the day following our blacklisting request for its URL. As the *sec-1* security company was listed as one of the partners of the blacklisting service, we expect that our URL blacklisting request was forwarded to the security service right after our submission, thus allowing a timely detection.

3.4 Lessons Learned, Conclusions

This section concludes our analysis of web attacks from the point of view of hosting providers. We can summarize the main findings of our experiments around the following five points:

Registration - Top providers invest a considerable effort to collect information about the users who register with them. This procedure can be an effective technique to prevent criminals from hosting their malicious pages on those providers.

Prevention - About 40% of the providers deployed some kind of security mechanism to block simple attacks, ranging from SQL injections to exploitation of common web application vulnerabilities.

Detection - Once the customer is registered, most of the providers do nothing to detect malicious activities or compromised websites - therefore providing very little help to their customers. We were surprised to discover that 21 out of the 22 tested providers did not even run an antivirus once per month (or they run them with old or insufficient signature sets) on the hosted websites. Moreover, none of them considered suspicious having multiple outgoing connection attempts towards an IRC server.

Abuse Notification - Only 36% of the providers reacted to our abuse notifications. When they promptly replied, most of the time their reaction was inappropriate or excessive. None of the *global* providers and only one of the *regional* ones were able to properly manage both the real and the fake complaints in a timely manner.

Security Services - The use of inexpensive security add-on services did not provide any additional layer of security in our experiments. Also the services that were configured to scan the content of our sites via FTP failed to discover the malicious files.

The main differences between *global* and *regional* providers appeared to be in terms of registration verification (in favor of *global* providers) and reaction to real complaints (in favor of *regional* ones).

As we already mentioned in the beginning of this chapter, web hosting providers are in the position to play a key role in the security of the Web. In fact, they host millions of websites that are often poorly managed by inexperienced users, and that are likely to be compromised to spread malware and host phishing kits. Unfortunately, all the shared web hosting providers we tested in our study missed this opportunity.

Chapter 4

Web Attacks From the Attacker's Point of View

This chapter analyzes the details of how websites are attacked from the point of view of attackers and criminals exploiting them. As we already mentioned in Chapters 1 and 2, web attacks are nowadays one of the major threats on the Internet, and several studies have analyzed them, providing details on how they are performed and how they spread. However, no study seems to have sufficiently analyzed the typical behavior of an attacker *after* a website has been compromised.

In this chapter, we present the design, implementation, and deployment of a network of 500 fully functional honeypot websites, hosting a range of different services, whose aim is to attract attackers and collect information on what they do during and after their attacks. In 100 days of experiments, our system automatically collected, normalized, and clustered over 85,000 files that were created during approximately 6,000 attacks. Labeling the clusters allowed us to draw a general picture of the attack landscape, identifying the behavior behind each action performed both during and after the exploitation of a web application (such as installing a phishing web page, a botnet script, or a local exploit to escalate privileges on the compromised machine.)

4.1 Introduction

Web attacks are one of the most important sources of loss of financial and intellectual property. In the last years, such attacks have been evolving in number and sophistication, targeting governments and high profile companies, stealing valuable personal user information and causing financial losses of millions of euros. Moreover, the number of people browsing the web through computers, tablets and smartphones is constantly increasing, making web-related attacks a very appealing target for criminals.

This trend is also reflected in the topic of academic research. In fact, a quick look at the papers published in the last few years shows how a large number of

them cover web-related attacks and defenses. Some of these studies focus on common vulnerabilities related to web applications, web servers, or web browsers, and on the way these components get compromised. Others dissect and analyze the internals of specific attack campaigns [11, 51, 74], or propose new protection mechanisms to mitigate existing attacks.

The result is that almost all the web infections panorama has been studied in detail: how attackers scan the web or use google dorks to find vulnerable applications, how they run automated attacks, and how they deliver malicious content to the final users. However, there is still a missing piece in the puzzle. In fact, before our analysis, no work seems to have sufficiently detailed the behavior of an average attacker *during* and *after* a website is compromised. Sometimes the attackers are only after the information stored in the service itself, for instance when the goal is to steal user credentials through a SQL injection. But in the majority of the cases, the attacker wants to maintain access to the compromised machine and include it as part of a larger malicious infrastructure (e.g., to act as a C&C server for a botnet or to deliver malicious documents to the users who visit the page).

While the recent literature often focuses on catchy topics, such as drive-by-downloads and black-hat SEO, this is just the tip of the iceberg. In fact, there is a wide variety of malicious activities performed on the Internet on a daily basis, with goals that are often different from those of the high-profile cyber criminals who attract the media and the security firms' attention.

The main reason for which no previous work was done in this direction of research is that almost all of the existing projects based on web honeypots use fake, or 'mock' applications. This means that no real attacks can be performed and thus, in the general case, that all the steps that would commonly be performed by the attacker after the exploitation will be missed.

As a result, to better understand the motivation of the various classes of attackers, antivirus companies have often relied on the information reported by their clients. For example, in a recent survey conducted by Commtouch and the StopBadware organization [16], 600 owners of compromised websites have been asked to fill a questionnaire to report what the attacker did after exploiting the website. The results are interesting, but the approach cannot be automated, it is difficult to repeat, and there is no guarantee that the users (most of the time not experts in security) were able to successfully distinguish one class of attack from the other.

In this chapter we provide, for the first time, a comprehensive and aggregate study of the behavior of attackers on the web. We focus our analysis on two separate aspects: i) the exploitation phase, in which we investigate how attacks are performed until the point where the application is compromised, and ii) the post-exploitation phase, in which we examine what attackers do after they take control of the application. The first part deals with methods and techniques (i.e., the *"how"*) used to attack web applications, while the second part tries to infer the reasons and goals (i.e., the *"why"*) behind such attacks.

For this reason, this study does not analyze common SQL injections or cross-site scripting vulnerabilities. Instead, our honeypots are tailored to attract and mon-

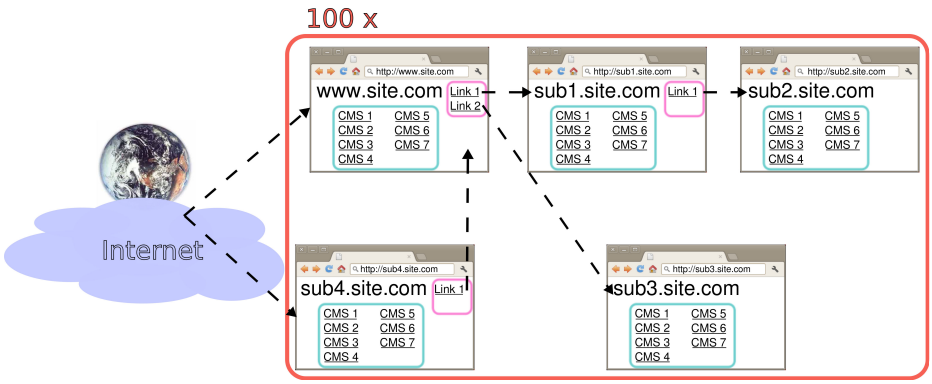

Figure 4.1: Architecture of the system - high level.

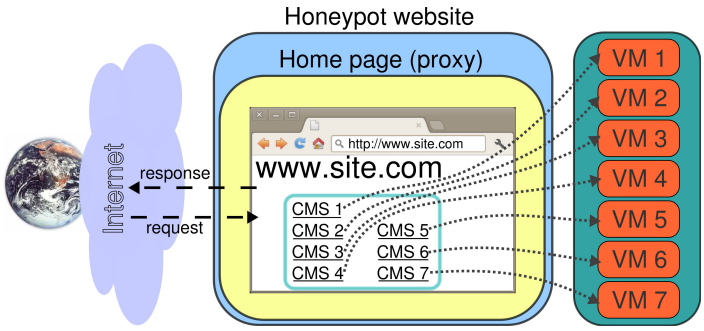

Figure 4.2: Architecture of the system - detail.

itor criminals that are interested in gaining (and maintaining) control of web applications. Our results show interesting trends on the way in which the majority of such attacks are performed in the wild. For example, we identify 4 separate phases and 13 different goals that are commonly pursued by the attackers. We also provide some insights into a few interesting attack scenarios that we identified during the operation of our honeypots.

4.2 HoneyProxy

Our honeypot system is composed of a number of websites (500 in our experiments), each containing the installation of five among the most common - and notoriously vulnerable - content management systems, 17 pre-installed PHP web shells, and a static web site.

We mitigated the problem of managing a large number of independent installations by hosting all the web applications in our facilities, in seven isolated virtual machines running on a VMWare Server. On the hosting provider side we installed only an ad-hoc proxy script (*HoneyProxy*) in charge of forwarding all the received traffic to the right VM on our server. This allowed us to centralize the data col-

lection while still being able to distinguish the requests from distinct hosts. A high-level overview of the system is shown in Figure 4.1.

The PHP proxy adds two custom headers to each request it receives from a visitor:

- *X-Forwarded-For*: this standard header, which is used in general by proxies, is set to the real IP address of the client. In case the client arrives with this header already set, the final X-Forwarded-For will list all the previous IPs seen, keeping thus track of all the proxies traversed by the client.
- *X-Server-Path*: this custom header is set by the PHP proxy in order to make it possible, for us, to understand the domain of provenance of the request when analyzing the request logs on the virtual machines. An example of such an entry is: `X-Server-Path: http://sub1.site.com/`

These two headers are transmitted for tracking purposes only between the hosting provider's webserver and the honeypot VM's webserver, and thus are not visible to the users of the HoneyProxy.

4.2.1 Containment

Each virtual machine was properly set up to contain the attackers and prevent them from causing any harm outside our honeypot. In particular, we blocked outgoing connections (which could otherwise result in attacks to external hosts), patched the source code of the vulnerable blog and forum applications to hide messages posted by spammers (that could result in advertising malicious links), and tuned the file system privileges to allow attackers to perpetrate their attacks, but not to take control of the machine or to modify the main source files of each application. Still, the danger of hosting malicious files uploaded by attackers exists, and we tackle this problem by restoring every virtual machine to its pristine state at regular time intervals.

In the following lines, we briefly explain the possible abuses that can be perpetrated on a honeypot machine and present our way to prevent or mitigate them.

- *Gaining high privileges on the machine*. We tackle this problem by using virtual machines with up-to-date software and security patches. In each virtual machine, the web server and all exposed services run as non privileged user. Of course, this solution does not guarantee a protection against new 0-day attacks, but we did our best to limit the attack surface, having only 3 services running on the machine (apache,sshd,mysqld), among which only the web server is exposed to the Internet. We considered the possibility of a 0-day attack against apache fairly remote, and, may it happen, a vast majority of the Internet will be exposed to it as well.
- *Using the honeypot machine as a stepping stone to launch attacks or email campaigns*. This is probably the most important concern that has to be addressed before deploying a fully functional honeypot machine. In our case, we used regular `iptables` rules to block (and log) all outgoing traffic from the virtual machines, except for already established connections. One excep-

tion to this rule is the IRC port (6667). We will explain this in more detail in sections 4.3 and 4.5.

- *Hosting and distributing illegal content(e.g., phishing pages).* It is difficult to prevent this threat when applications have remote file upload vulnerabilities. However, it is possible to mitigate the risk of distributing illegal content by limiting the privileges of directories in which files can be uploaded and preventing the modification of all the existing HTML and PHP files. In addition, we also monitor every change on the VM file systems, and whenever a file change is detected, the system takes a snapshot of it. The virtual machine is then restored, at regular intervals, to its original snapshot, thus preventing potentially harmful content from being delivered to victims or indexed by search engines.
- *Illegally promoting goods or services (e.g., spam links).* Another issue is raised by applications that, as part of their basic way of working, allow users to write and publish comments or posts. This is the case for any blog or forum CMS. These applications are often an easy target for spammers, as we will show in section 4.4.3, and when hosting an honeypot it is important to make sure that links and posts that are posted by bots do not reach any end user or do not get indexed by search engines. We solved this problem by modifying the source code of the blog and forum applications (namely, Wordpress and Simple Machines Forum), commenting out the snippets of code responsible of showing the content of posts. With this modification, it was still possible for attackers to post messages (and for us to collect them), but navigating the posts or comments will only show blank messages.

These countermeasures are limiting the information we can collect with our honeypot (e.g., in the case in which an attacker uploads a back-connect script that is blocked by our firewall), but we believe they are necessary to prevent our infrastructure to be misused for malicious purposes.

4.2.2 Data Collection and Analysis

Our analysis of the attackers' behavior is based on two sources of information: the logs of the incoming HTTP requests, and the files that are modified or generated by the attackers after they obtain access to the compromised machines.

We built some tools for the analysis of HTTP request logs, allowing us to identify known benign crawlers, known attacks on our web applications, as well as obtaining detailed statistics (number and type of requests received, User-Agent, IP address and geolocalization of every visitor, analysis of the 'Referer' header, and analysis of the inter-arrival time between requests). Our analysis tools also allow us to normalize the time of attack relatively to the timezone of the attacker, and to detect possible correlations between attacks (e.g., an automated script infecting a web application uploading a file, followed by another IP visiting the uploaded file from another IP address). We also developed a parser for the HTTP request logs

of the most commonly used PHP web shells, allowing us to extract the requested commands and understand what the attacker was doing on our systems.

We employed two sources of uploaded or modified files: webserver logs and file snapshots from monitored directories. Webserver logs are the primary source of uploaded files, as every file upload processed by our honeypots is fully logged on the apache mod_security logs. File snapshots from monitored directories on the virtual machines, instead, are the primary source for files that are modified or generated on the machine, or about archives or encrypted files that are decompressed on the system. The total number of files we were able to extract from these sources was 85,567, of which 34,259 unique.

Given the high number of unique files we collected, a manual file analysis was practically infeasible. Therefore, in order to ease the analysis of the collected data, we first separate files according to their types, and then apply similarity clustering to see how many of them actually differ from each other in a substantial way. This allows us to identify common practices in the underground communities, such as redistributing the same attack or phishing scripts after changing the owner's name, the login credentials, or after inserting a backdoor.

First of all we employed the *file* Linux utility to categorize files and group them in 10 macro-categories: source code, picture, executable, data, archive, text, HTML document, link, multimedia, and other.

We then observed that many files in the same category only differ by a few bytes (often white spaces due to cut&paste) or to different text included in source code comments. Therefore, to improve the results of our comparison, we first pre-processed each file and transformed it to a normalized form. As part of the normalization process, we removed all double spaces, tabs and new line characters, we removed all comments (both C-style and bash-style), and we normalized new lines and stripped out email addresses appearing in the code. For HTML files, we used the *html2text* utility to strip out all HTML tags as well.

PHP files underwent an additional pre-processing step. We noticed that a large amount of PHP files that were uploaded to our honeypots as result of an exploitation were obfuscated. For files in this form it is very difficult, even with automated tools, to detect similarities among similar files encoded in different ways. In order to overcome this issue, we built an automatic PHP deobfuscation tool based on the *evalhook* PHP extension [26], a module that hooks every call to dynamic code evaluation functions, allowing for step-by-step deobfuscation of PHP code. We deployed our tool on a virtual machine with no network access (to avoid launching attacks or scans against remote machines, as some obfuscated scripts could start remote connections or attacks upon execution) and, for each file with at least one level of deobfuscation (i.e., nested call to *eval()*), we saved its deobfuscated code.

Our approach allowed us to deobfuscate almost all the PHP files that were obfuscated using regular built-in features of the language (e.g., gzip and base64 encoding and decoding, dynamic code evaluation using the *eval()* function). The only obfuscated PHP files we were not able to decode were those terminating with an error (often because of syntax errors) and those encoded with specialized com-

mercial tools, such as Zend Optimizer or ionCube PHP Encoder. However, we observed only three samples encoded with these tools.

In total, we successfully deobfuscated 1,217 distinct files, accounting for 24% of the source code we collected. Interestingly, each file was normally encoded multiple times and required an average of 9 rounds of de-obfuscation to retrieve the original PHP code (with few samples that required a stunning 101 rounds).

Similarity Clustering

Once the normalization step was completed, we computed two similarity measures between any given couple of files in the same category, using two state-of-the-art tools for (binary data) similarity detection: *ssdeep* [57] and *sdhash* [103]. We then applied a simple agglomerative clustering algorithm to cluster all files whose similarity score was greater than 0.5 into the same group.

We discarded files for which our analysis was not able to find any similar element. For the remaining part, we performed a manual analysis to categorize each cluster according to its purpose. Since files had already been grouped by similarity, only the analysis (i.e., opening and inspecting the content) of one file per group was necessary. During this phase, we were able to define several file categories, allowing us to better understand the intentions of the attackers. Moreover, this step allowed us to gain some insights on a number of interesting attack cases, some of which are reported in the following sections as short in-depth examples.

4.3 System Deployment

The 500 honeyproxy have been deployed on shared hosting plans * chosen from eight of the most popular international web hosting providers on the Internet (from USA, France, Germany, and the Netherlands). In order for our HoneyProxy to work properly, each provider had to support the use of the cURL libraries through PHP, and allow outgoing connections to ports other than 80 and 443.

To make our honeypots reachable from web users, we purchased 100 bulk domain names on GoDaddy.com with privacy protection. The domains were equally distributed among the `.com`, `.org`, and `.net` TLDs, and assigned evenly across the hosting providers. On each hosting provider, we configured 4 additional subdomains for every domain, thus having 5 distinct websites (to preserve the anonymity of our honeypot, hereinafter we will simply call them www.site.com, sub1.site.com, sub2.site.com, sub3.site.com, sub4.site.com) Finally, we advertised the 500 domains on the home page of the authors and on the research group's website by means of transparent links, as already proposed by Müter et al. [78] for a similar purpose.

∗. This is usually the most economical hosting option, and consists in having a website hosted on a web server where many other websites reside and share the machine's resources.

We used a modified version of the *ftp-deploy* script [42] to upload, in batch, a customized PHP proxy to each of the 500 websites in our possession. This simplified the deployment and update of the PHP proxy, and uniformed the way in which we upload files to each hosting service †, Thanks to a combination of `.htaccess`, ModRewrite, and cURL, we were able to transparently forward the user requests to the appropriate URL on the corresponding virtual machine. Any attempt to read a non-existing resource, or to access the proxy page itself would result in a blank error page shown to the user. Not taking into account possible timing attacks or intrusions on the web hosting provider's servers, there was no way for a visitor to understand that he was talking to a proxy.

The HoneyProxy system installed on every website is composed of an index file, the PHP proxy script itself and a configuration file. The index file is the home page of the website, and it links to the vulnerable web applications and to other honeypot websites, based on the contents of the configuration file.

The linking structure is not the same for every subdomain, as can be noticed taking a closer look at Figure 4.1. Indeed, each subdomain links to at most 2 different subdomains under its same domain. We put in place this small linking graph with the aim of detecting possible malicious traffic from systems that automatically follow links and perform automated attacks or scans.

4.3.1 Installed Web Applications

We installed a total of 5 vulnerable CMSs on 7 distinct Virtual Machines. The Content Management Systems were chosen among the most known and vulnerable ones at the time we started our deployment. For each CMS, we chose a version with a high number of reported vulnerabilities, or at least with a critical one that would allow the attacker to take full control of the application. We also limited our choice to version no more than 5 years old in order to ensure our websites are still of interest to attackers.

Our choice was guided by the belief that attackers are always looking for low-hanging fruits. On the other hand, our honeypots will probably miss sophisticated and unconventional attacks, mostly targeted to high profile organizations or well known websites. However, these attacks are not easy to study with simple honeypot infrastructures and are therefore outside the scope of our study.

Table 4.1 describes the vulnerable applications installed on the 7 virtual machines, along with their publication date and the list of their known and exploitable vulnerabilities. We have installed two instances of WordPress 2.8, one with CAPTCHA protection on comments, and one without CAPTCHA protection, in order to see if there are attackers that register fake accounts by hand, or systems that are capable of automatically solve CAPTCHAs. This does not seem to be the case, since we

†. Shared web hosting services from different providers usually come with their own custom administrative web interface and directory structure, and very few of them offer ssh access or other 'advanced' management options. Thus, the only possible way to automate the deployment of the websites was to use FTP, the only protocol supported by every provider.

VM #	CMS, version	Plugins	Description	Vulnerabilities
1	phpMyAdmin, 3.0.1.1	-	MySQL database manager	PHP code injection
2	osCommerce, 2.2-RC2a	-	Online shop	2 remote file upload, arbitrary admin password modification
3	Joomla, 1.5.0	com_graphics, tinymce	Generic/multipurpose portal	XSS, arbitrary admin password modification, remote file upload, local file inclusion
4	Wordpress, 2.8	kino, amphion lite theme	Blog (non moderated comments)	Remote file include, admin password reset
5	Simple Machines Forum (SMF), 1.1.3	-	Forum (non moderated posts)	HTML injection in posts, stored XSS, blind SQL injection, local file include (partially working)
6	PHP web shells, static site	-	Static site and 17 PHP shells (reachable through hidden links)	PHP shells allow to run any kind of commands on the host
7	Wordpress, 2.8	kino, amphion lite theme	Blog (captcha-protected comments)	Remote file include, admin password reset

Table 4.1: Applications installed on the honeypot virtual machines, together with a brief description and a list of their known and exploitable vulnerabilities.

did not receive any post on the CAPTCHA-protected blog. Therefore, we will not discuss it any further in the rest of this study.

4.3.2 Data Collection

We collected 100 days of logs on our virtual machines, starting December 23rd, 2011. All the results presented in our work derive from the analysis of the logs of these 7 machines.

Overall, we collected 9.5 GB of raw HTTP requests, consisting in approximately 11.0M GET and 1.9M POST. Our honeypots were visited by more than 73,000 different IP addresses, spanning 178 countries and presenting themselves with more than 11,000 distinct User-Agents. This is over one order of magnitude larger than what has been observed in the previous study by John et al. on low interaction web-application honeypots [53]. Moreover, we also extracted over 85,000 files that were uploaded or modified during attacks against our web sites.

There are two different ways to look at the data we collected: one is to identify and study the attacks looking at the web server logs, and the other one is to try to associate a *goal* to each of them by analyzing the uploaded and modified files. These two views are described in more detail in the next two Sections.

4.4 Exploitation and Post-Exploitation Behaviors

In order to better analyze the behavior of attackers lured by our honeypots, we decided to divide each attack in four different phases: discovery, reconnaissance, exploitation, and post-exploitation. The *Discovery* phase describes how attackers find their targets, e.g. by querying a search engine or by simply scanning IP addresses. The *Reconnaissance* phase contains information related to the way in

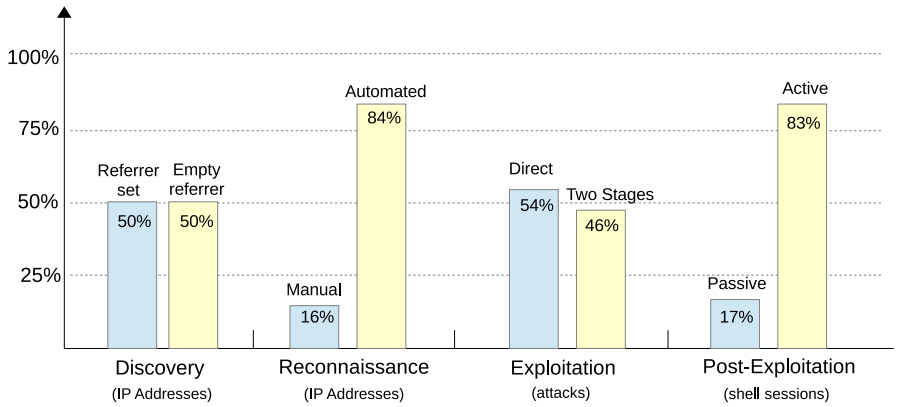

Figure 4.3: Overview of the four phases of an attack

which the pages were visited, for instance by using automated crawlers or by manual access through an anonymization proxy. In the *Exploitation* phase we describe the number and types of actual attacks performed against our web applications. Some of the attacks reach their final goal themselves (for instance by changing a page to redirect to a malicious website), while others are only uploading a second stage. In this case, the uploaded file is often a web shell that is later used by the attacker to manually log in to the compromised system and continue the attack. We refer to this later stage as the *Post-Exploitation* phase.

It is hard to present all possible combinations of behaviors. Not all phases are always present in each attack (e.g., reconnaissance and exploitation can be performed in a single monolithic step), some of the visits never lead to any actual attack, and sometimes it is just impossible to link together different actions performed by the same attacker with different IP addresses. However, by extracting the most common patterns from the data collected at each stage, we can identify the "typical attack profile" observed in our experiment. Such profile can be summarized as follows:

1. 69.8% of the attacks start with a *scout bot* visiting the page. The scout often tries to hide its User Agent or disguise as a legitimate browser or search engine crawler.

2. Few seconds after the scout has identified the page as an interesting target, a second automated system (hereinafter *exploitation bot*) visits the page and executes the real exploit. This is often a separate script that does not fake the user agent, therefore often appearing with strings such as `libwww/perl`.

3. If the vulnerability allows the attacker to upload a file, in 46% of the cases the exploitation bot uploads a web shell. Moreover, the majority of the attacks upload the same file multiple times (in average 9, and sometimes up to 30), probably to be sure that the attack was successful.

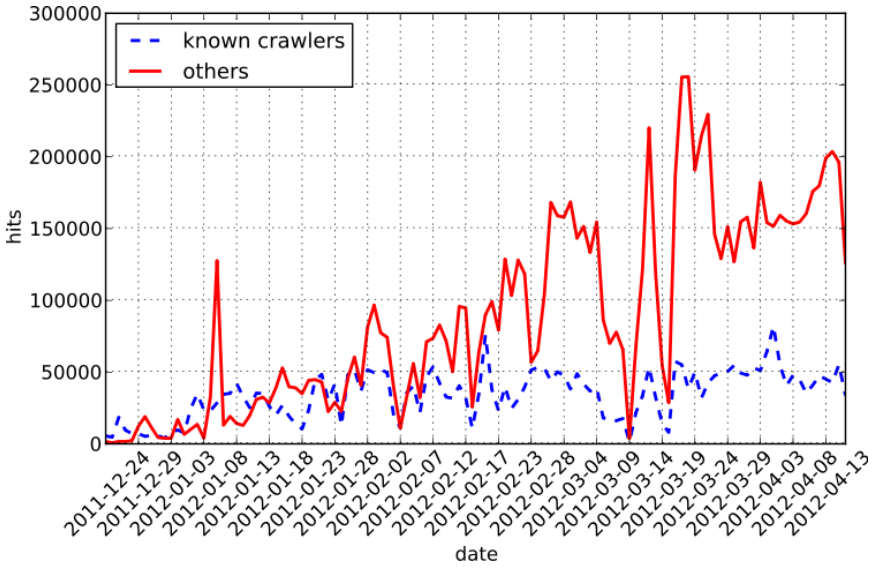

Figure 4.4: Volume of HTTP requests received by our honeypots during the study.

4. After an average of 3 hours and 26 minutes, the attacker logs into the machine using the previously uploaded shell. The average login time for an attacker interactive session is 5 minutes and 37 seconds.

While this represents the most common behavior extracted from our dataset, many other combinations were observed as well - some of which are described in the rest of the section. Finally, it is important to mention that the attack behavior may change depending on the application and on the vulnerability that is exploited. Therefore, we should say that the previous description summarizes the most common behavior of attacks against osCommerce 2.2 (the web application that received by far the largest number of attacks among our honeypots).

Figure 4.3 shows a quick summary of some of the characteristics of each phase.[‡] More information and statistics are reported in the rest of the section. Then, based on the analysis of the files uploaded or modified during the exploitation and post-exploitation phases, in Section 4.5 we will try to summarize the different goals and motivations behind the attacks we observed in our experiments.

4.4.1 Discovery

The very first HTTP request hit our honeypot proxies only 10 minutes after the deployment, from Googlebot. The first direct request on one IP address of our virtual machines (running on port 8002) came after 1 hour and 50 minutes.

During the first few days, most of the traffic was caused by benign web crawlers. Therefore, we designed a simple solution to filter out benign crawler-generated

‡. The picture does not count the traffic towards the open forum, because its extremely large number of connections compared with other attacks would have completely dominated the statistics.

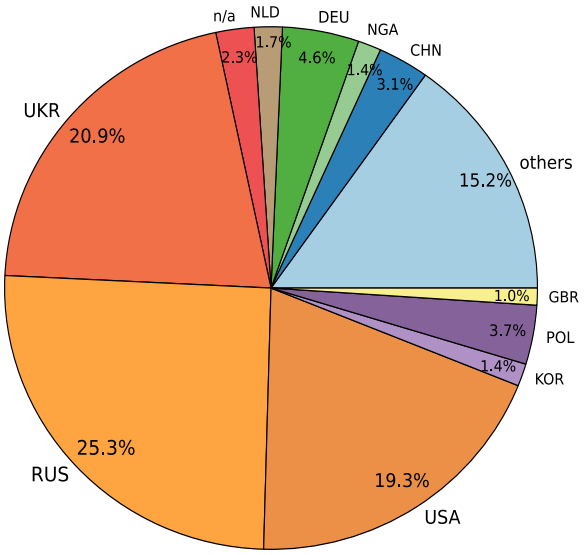

Figure 4.5: Amount of requests, by issuing country.

traffic from the remaining traffic. Since HTTP headers alone are not trustable (e.g., attackers often use User Agents such as 'Googlebot' in their scripts) we collected public information available on bots [101, 112] and we combined them with information extracted from our logs and validated with WHOIS results in order to identify crawlers from known companies. By combining UserAgent strings and the IP address ranges associated to known companies, we were able to identify with certainty 14 different crawlers, originating from 1965 different IPs. Even though this is not a complete list (e.g, John et al. [53] used a more complex technique to identify 16 web crawlers), it was able to successfully filter out most of the traffic generated by benign crawlers.

Some statistics about the origin of the requests is shown in Figure 4.4. The amount of legitimate crawler requests is more or less stable in time, while, as time goes by and the honeypot websites get indexed by search engines and linked on hacking forums or on link farming networks, the number of requests by malicious bots or non-crawlers has an almost linear increase.

When plotting these general statistics we also identified a number of suspicious spikes in the access patterns. In several cases, one of our web applications was visited, in few hours, by several thousands of unique IP addresses (compared with an average of 192 per day), a clear indication that a botnet was used to scan our sites.

Interestingly, we observed the first suspicious activity only 2 hours and 10 minutes after the deployment of our system, when our forum web application started receiving few automated registrations. However, the first posts on the forum appeared only four days later, on December 27th. Even more surprising was the fact that the first visit from a non-crawler coincided with the first attack: 4 hours 30

minutes after the deployment of the honeypots, a browser with Polish locale visited our osCommerce web application [§] and exploited a file upload vulnerability to upload a malicious PHP script to the honeypot. Figure 4.5 summarizes the visits received by our honeypot (benign crawlers excluded), grouped by their geolocalization.

Referer Analysis

The analysis of the Referer HTTP header (whenever available) helped us identify how visitors were able to find our honeypots on the web. Based on the results, we can distinguish two main categories of users: criminals using search engines to find vulnerable applications, and victims of phishing attacks following links posted in emails and public forums (an example of this phenomenon is discussed in Section 4.5.8).

A total of 66,449 visitors reached our honeypot pages with the Referer header set. The domains that appear most frequently as referrers are search engines, followed by web mails and public forums. Google is leading with 17,156 entries. Other important search engines used by the attackers to locate our websites, were Yandex (1,016), Bing (263), and Yahoo (98). A total of 7,325 visitors arrived from web mail services (4,776 from SFR, 972 from Facebook, 944 were from Yahoo!Mail, 493 from Live.com, 407 from AOL Mail, and 108 from comcast.net). Finally, 15,746 requests originated from several public web forums, partially belonging to hacking communities, and partially just targeted by spam bots.

Finally, we extracted search queries (also known as 'dorks', when used for malicious purposes) from Referer headers set by the most common web search engines. Our analysis shows that the search terms used by attackers highly depend on the application deployed on the honeypot. For example, the most common dork that was used to reach our Joomla web application contained the words *'joomla allows you'*, while the Simple Machines Forum was often reached by searching *'powered by smf'*. Our machine containing public web shells was often reached via dorks like *'inurl:c99.php'*, *'[cyber anarchy shell]'* or even *'[ftp buteforcer] [security info] [processes] [mysql] [php-code] [encoder] [backdoor] [back-connection] [home] [enumerate] [md5-lookup] [word-lists] [milw0rm it!] [search] [self-kill] [about]'*. The latter query, even though very long, was used more than 150 times to reach our machine with web shells. It was probably preferred to searching via `'intitle:'` or `'inurl:'` because script names and titles are often customized by attackers and as such searching for their textual content may return more results than searching for fixed URL patterns or page titles. Some specialized search engines appear to be used as well, such as devilfinder.com, which was adopted in 141 cases to reach some of the shells on our machines. This search engine claims to show more low-ranking results than common search engines, not to store any search data, and to return up to 300 results on the same web page, making it very

§. Since UserAgent information can be easily spoofed, we cannot prove that our assumptions about the browser and tools run by the attacker, and his or her locale, are correct.

suitable for attackers willing to search for dorks and collect long lists of vulnerable websites.

4.4.2 Reconnaissance

After removing the legitimate crawlers, the largest part of the traffic received by our honeypots was from unidentified sources, many of which were responsible of sending automated HTTP requests. We found these sources to be responsible for the majority of attacks and spam messages targeting our honeypots during the study.

However, distinguishing attackers that manually visited our applications from the ones that employed automated scout bots is not easy. We applied the following three rules to flag the automated requests:

- *Inter-arrival time*. If requests from the same IP address arrive at a frequency higher than a certain threshold, we consider the traffic as originated from a possible malicious bot.
- *Request of images*. Automated systems, and especially those having to optimize their speed, almost never request images or other presentation-related content from websites. Scanning web logs for visitors that never request images or CSS content is thus an easy way of spotting possible automated scanners.
- *Subdomain visit pattern*. As described in Section 4.3, each web site we deployed consisted in a number of sub-domains linked together according to a predetermined pattern. If the same IP accesses them in a short time frame, following our patterns, then it is likely to be an automated crawler.

For example, after removing the benign crawlers, a total of 9.5M hits were received by systems who did not request any image, against 1.8M from system that also requested images and presentation content. On the contrary, only 641 IP addresses (responsible for 13.4K hits) visited our websites by following our links in a precise access pattern. Among them, 60% followed a breadth first approach.

85% of the automated requests were directed to our forum web application, and were responsible for registering fake user profiles and posting spam messages. Of the remaining 1.4M requests directed to the six remaining honeypot applications, 95K were mimicking the User-Agent of known search engines, and 264K switched between multiple User-Agents over time. The remaining requests did not contain any suspicious User-Agent string, did not follow paths between domains, neither requested images. As such, we classified them as unknown (possibly benign) bots.

4.4.3 Exploitation

The first important activity to do in order to detect exploitation attempts was parsing the log files in search of attack traces. Luckily, knowing already the vulnerabilities affecting our web applications allowed us to quickly and reliably scan for attacks in our logs using a set of regular expressions.

Overall, we logged 444 distinct exploitation sessions. An interesting finding is that 310 of them adopted two or more different User-Agent strings, appearing in short sequence from the same IP address. As explained in the beginning of Section 4.4, this often happens when attackers employ a combination of scout bots and automatic attack scripts in order to speed up attacks and quickly find new targets. In particular, in two thirds (294) of the total exploitation sessions we observed, the User-Agent used for the exploitation was the one associated to the LibWWW Perl library (`libwww/perl`).

In some of these exploitation sessions, the attacker tried to disguise her tools and browser as known benign bots. Some crawler User-Agent strings that were often used during exploitation sessions were: *FreeWebMonitoring*, *Gigabot/3.0*, *gsa-crawler*, *IlTrovatore-Setaccio/1.2*, *bingbot/2.0;*, and *Googlebot/2.1*.

The most remarkable side effect of every exploitation session is the upload or modification of files on the victim machine. Quite surprisingly, we noticed that when an exploitation session uploads a file, the file is uploaded in average 9.75 times. This strange behavior can be explained by the fact that most of the exploitation tools are automated, and since the attacker does not check in real-time whether each exploit succeeded or not, uploading the same file multiple times can increase the chance for the file to be successfully uploaded at least once.

Using the approach presented in Section 4.2.2, we automatically categorized the files uploaded to our honeypots as a result of exploiting vulnerable services. We then correlated information about each attack session with the categorization results for the collected files. Results of this phase show that the files uploaded during attack sessions consist, in 45.75% of the cases, in web shells, in 17.25% of the cases in phishing files (single HTML pages or complete phishing kits), in 1.75% of the cases in scripts that automatically try to download and execute files from remote URLs, and in 1.5% of the cases in scripts for local information gathering. Finally, 32.75% of the uploaded files were not categorized by our system, either because they were not similar to anything else that we observed, or because they were multimedia files and pictures (e.g., images or soundtracks for defacement pages) that were not relevant for our study.

Figure 4.6 shows the normalized times of the attacks received by our honeypots. The values were computed by adjusting the actual time of the attack with the timezone extracted from the IP geolocalization. As such, our normalization does not reflect the correct value in case the attacker is proxying its connection through an IP in a different part of the world. However, the graph shows a clear daylight trend for both the exploitation and post-exploitation phases. In particular, for the interactive sessions we observed fewer attacks performed between 4am and 10am, when probably also the criminals need to get some sleep. Interestingly, also the exploitation phase, that is mostly automated, shows a similar trend (even though not as clear). This could be the consequence of scans performed through botnet infected machines, some of which are probably turned off by their users during the night.

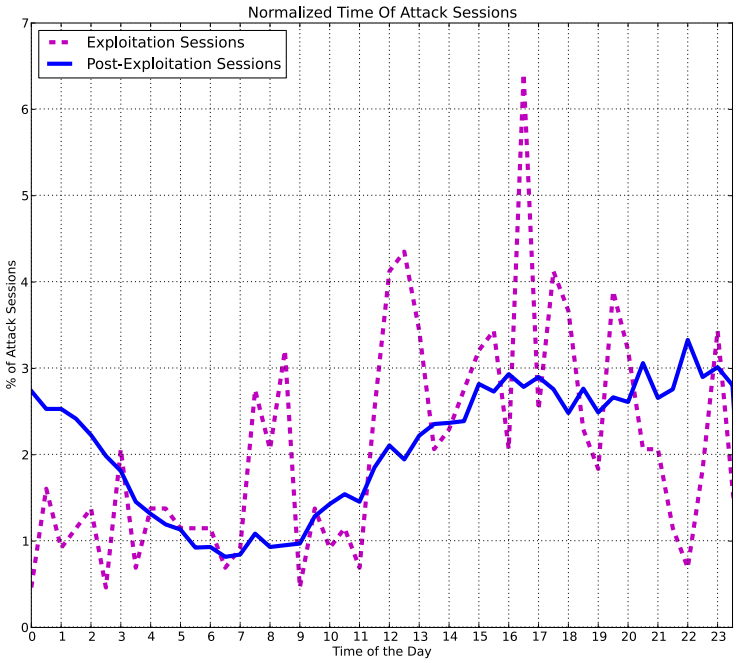

Figure 4.6: Normalized time distribution for attack sessions

Searching our attack logs for information about attackers reaching directly our virtual machines, without passing through the honeypot proxies, we found that a small, but still significant number of attacks were carried out directly against the ip:port of our honeypots. In particular, we found 25 of such attack sessions against our e-commerce web honeypot and 19 against our machine hosting the web shells and the static website. In both cases, the attacker may have used a previous exploit to extract the IP of our machines (stored in a osCommerce configuration file that was often downloaded by many attackers, or by inspecting the machine through an interactive shell) and use this information in the following attacks.

Posts

Since the 1st day of operation, our forum application received a very large amount of traffic. Most of it was from automated spamming bots that kept flooding the forum with fake registrations and spam messages. We analyzed every snapshot of the machine's database in order to extract information about the forum's posts and the URLs that were embedded in each of them. This allowed us to identify and categorize several spam and link farming campaigns, as well as finding some rogue practices such as selling forum accounts.

A total of 68,201 unique messages were posted on the forum during our study, by 15,753 users using 3,144 unique IP addresses. Daily statistics on the forum show trends that are typical of medium to high traffic message boards: an average

of 604 posts per day (with a max of 3085), with an average of 232 online users during peak hours (max 403).

Even more surprising than the number of posts is the number of new users registered to the forum: 1907 per day in average, and reaching a peak of 14,400 on March 23, 2012. This phenomenon was so common that 33.8% of the IP addresses that performed actions on our forum were responsible of creating at least one fake account, but never posted any message. This finding suggests there are some incentives for criminals to perform automatic user registrations, perhaps making this task even more profitable than the spamming activity itself. Our hypothesis is that, in some cases, forum accounts can be sold in bulk to other actors in the black market. We indeed found 1,260 fake accounts that were created from an IP address and then used few days later by other, different IPs, to post messages. This does not necessarily validate our hypothesis, but shows at least that forum spamming has become a complex ecosystem and it is difficult, nowadays, to find only a single actor behind a spam or link farming campaign.

A closer look at the geolocation of IP addresses responsible for registering users and posting to the forum shows that most of them are from the United States or Eastern Europe countries (mostly Russia, Ukraine, Poland, Latvia, Romania). A total of 6687 distinct IP addresses were active on our forum (that is, posted at least one message or registered one or more accounts). Among these, 36.8% were associated to locations in the US, while 24.6% came from Eastern European countries. The country coverage drastically changes if we consider only IP addresses that posted at least one message to the forum. In this case, IPs from the United States represent, alone, 62.3% of all the IP addresses responsible for posting messages (Eastern Europe IPs in this case represent 21.2% of the total).

Finally, we performed a simple categorization on all the messages posted on the forum, based on the presence of certain keywords. This allowed us to quickly identify common spam topics and campaigns. Thanks to this method, we were able to automatically categorize 63,763 messages (93.5% of the total).

The trends we extracted from message topics show clearly that the most common category is drugs (55% of the categorized messages, and showing peaks of 2000 messages per day), followed by search engine optimization (SEO) and electronics (11%), adult content (8%), health care and home safety (6%).

All the links inserted in the forum posts underwent an in-depth analysis using two automated, state-of-the-art tools for the detection of malicious web pages, namely Google Safe Browsing [105] and Wepawet [17]. The detection results of these two tools show that, on the 221,423 URLs we extracted from the forum posts, a small but not insignificant fraction (2248, roughly 1 out of 100) consisted in malicious or possibly harmful links.

4.4.4 Post-Exploitation

The post-exploitation phase includes the analysis of the interaction between the attackers and the compromised machines. In our case, this is done through the

web shells installed during the exploitation phase or, to increase the collected data, through the access to the public shells that we already pre-installed in our virtual machines.

The analysis of the post-exploitation phase deserves special attention since it is made of interactive sessions in which the attackers can issue arbitrary commands. However, these web shells do not have any notion of session: they just receive commands via HTTP requests and provide the responses in a state-less fashion.

During our experiments we received a total of 74,497 shell commands. These varied from simple file system navigation commands, to file inspection and editing, up to complex tasks as uploading new files or performing network scans.

To better understand what this number represents, we decided to group together individual commands in virtual "interactive sessions" every time they are issued from the same IP, and the idle time between consecutive commands is less than 5 minutes.

According to this definition, we registered 232 interactive sessions as a consequence of one of the exploited services, and 8268 in our pre-installed shells ¶. The average session duration was of 5 minutes and 37 seconds, however, we registered 9 sessions lasting more than one hour each. The longest, in terms of commands issued to the system, was from a user in Saudi Arabia that sent 663 commands to the shell, including the manual editing of several files.

Interestingly, one of the most common actions performed by users during an attack is the upload of a custom shell, even if the attacker broke into the system using a shell that was already available on the website. The reason for this is that attackers know that, with a high probability, shells installed by others will contain backdoors and most likely leak information to their owner. In addition to the 17 web shells supported by our tools, we also identified the HTTP patterns associated to the most common custom shells uploaded by the attackers, so that we could parse the majority of commands issued to them.

In 83% of the cases, attackers tried to use at least one active command (uploading or editing a file, changing file permissions, creating files or directories, scanning hosts, killing a process, connecting to a database, sending emails, etc.). The remaining sessions were purely passive, with the attackers only browsing our system and downloading source and configuration files.

Finally, in 61% of the sessions the attackers uploaded a new file, and in 50% of them they tried to modify a file already on the machine (in 13% of the cases to perform a defacement). Regarding individual commands, the most commonly executed were the ones related to listing and reading files and directories, followed by editing files, uploading files, running commands on the system, listing the processes running on the system, and downloading files.

¶. For the pre-installed shells, we also removed sessions that contained very fast sequences of commands or that did not fetch images on the pages, because they could have been the result of crawlers visiting our public pages. Since shells uploaded by attackers were not linked from any page, we did not apply this filtering to them.

File Type	Clustered	Not Clustered	Clusters
Archive	335 (82.6%)	71 (17.4%)	159
Data	221 (62.5%)	133 (37.5%)	87
Executable	102 (82.3%)	22 (17.7%)	41
HTML doc	4341 (100.0%)	0 (0%)	822
Image	1703 (81.9%)	374 (18.1%)	811
Source code	3791 (100.0%)	0 (0%)	482
Text	886 (43.8%)	1138 (56.2%)	219
Various	118 (65.9%)	61 (34.1%)	42
Total	11,497 (86.5%)	1799 (13.5%)	2663

Table 4.2: Results of clustering

4.5 Attackers Goals

In this section we shift the focus from the way the attacks are performed to the motivation behind them. In other words, we try to understand what criminals do after they compromise a web application. Do they install a botnet? Do they try to gain administrator privileges on the host? Do they modify the code of the application and insert backdoors or malicious iFrames?

To answer these questions, we analyzed the files uploaded during the exploitation phase, and the ones created or modified during the post-exploitation phase. We normalized each file content as explained in Section 4.2, and we clustered them together according to their similarity. Finally, we manually labeled each cluster, to identify the "purpose" of the files. The results of the clustering are summarized in table 4.2 and cover, in total, 86.4% of the unique files collected by our honeypots. For these, Figure 4.7 shows the distribution of the file categories‖. For example, 1.7% of the unique files we observed in our experiments were used to try to escalate the privileges on the compromised machine. This is different from saying that 1.7% of the attackers tried to escalate the privileges of the machine. Unfortunately, linking the files to the attacks in which they were used is not always possible. Therefore, we computed an estimation of the attackers that performed a certain action by identifying each unique IP that uploaded a certain file during an attack. Identifying an attacker only based on his or her IP address is not always correct, but still provides a reasonable approximation. Thus, if we say that a certain

‖. We removed from the graph the irrelevant and damaged documents, that accounted in total for 10% of the files.

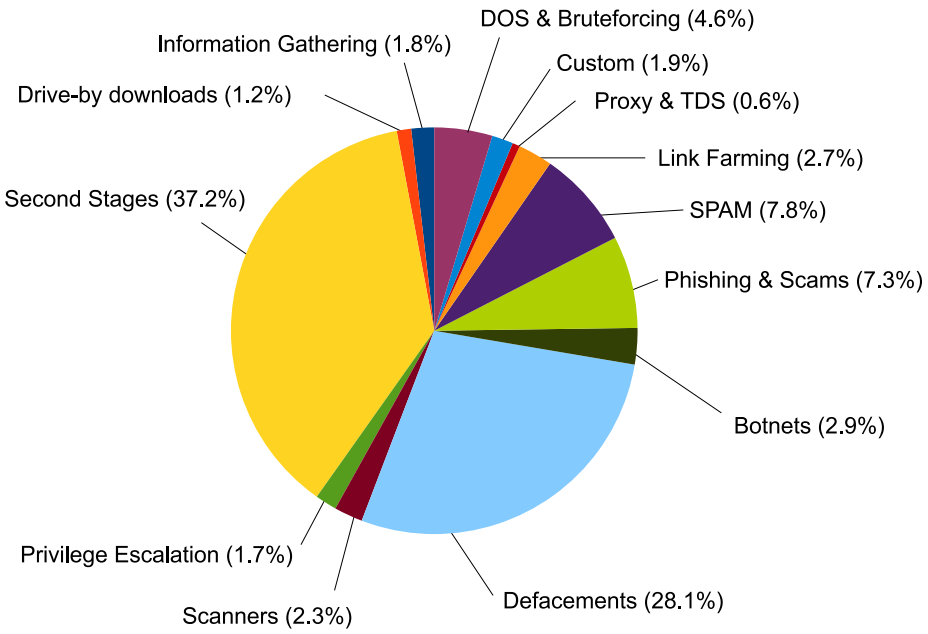

Figure 4.7: Attack behavior, based on unique files uploaded

category has an *estimated attackers ratio* of 20%, it means that 1 attacker out of 5 uploaded at least one file of that category during his or her operation.

Only 14% of the attackers uploaded multiple files belonging at least to two separate categories. This means that most of the attacks have a precise goal, or that attackers often change their IP addresses, making it very hard for us to track them.

In the rest of the section, we briefly introduce each of the 13 categories.

4.5.1 Information gathering

Unique files ratio	1.8%
Estimated attackers ratio	2.2%

These files consist mainly in automated scripts for the analysis of the compromised system, and are often used as a first stage of a manual attack, in which the attacker tries to gather information on the attacked system before proceeding with other malicious actions. In general, we observed a number of attackers using scripts to search, archive, and download several system configuration files.

For example, an attack using such tools hit our honeypots on April 7, 2012. The attacker, using a normal browser and coming from a Malaysian IP address, uploaded a script called `allsoft.pl`. Once executed, the script scans the system for a list of directories containing configuration files of known CMSs (e.g., Wordpress, Joomla, WHM, phpBB, vBulletin, ...), creates a tar archive containing all the files it was able to find, and returns to the attacker a link to the created archive, that can thus be easily downloaded. The script iterates on both the users and the

possible multiple home directories in the system trying to gather information from as many accounts as possible on the attacked machine.

4.5.2 Drive-by Downloads

Unique files ratio	1.2%
Estimated attackers ratio	1.1%

We have witnessed few attacks that aimed at creating drive-by download webpages, by inserting custom exploit code in the HTML source of the web pages of our honeypots, or by uploading documents that contain exploits for known browser vulnerabilities. This kind of activity is aimed at exploiting users visiting the website, typically to convert their machines in bots that can be later used for a large spectrum of illicit activity.

An example of such attacks was the *intu.html* web page uploaded to one of our honeypots on February 28th, 2012. When opened, the page shows *'Intuit Market. Loading your order, please wait...'*. Behind the scenes, a malicious JavaScript loads an iframe pointing to a document hosted at `twistedtarts.net`. This document is malicious and contains two exploits, for CVE-2010-0188 and CVE-2010-1885. Wepawet [17] reported the document as malicious on the same day this webpage was uploaded to our honeypots.

4.5.3 Second Stages

Unique files ratio	37.2%
Estimated attackers ratio	49.4%

This category includes downloaders (programs designed to download and execute another file), uploaders (web pages that can be used to remotely upload other files), web shells, and backdoors included in already existing documents. These are the tools of choice for attackers to perform manual web-based attacks. The reason is that such tools allow either to upload any file to the victim machine, or to issue arbitrary commands as if the attacker was logged in to one of the server's terminals. The majority of the attacks logged by our honeypot adopted a mix of web shells and custom scripts to try to hack the machine and install malicious software on it.

An example of this behavior is the attack that started at 6:50 am (GMT) on January 1st, 2012. An IP address from Englewood, Colorado, with an User-Agent set to *'blackberry8520_ver1_subvodafone'* connected directly to our honeypot virtual machine running osCommerce and exploited a file upload vulnerability, uploading several different PHP scripts, all of them launching IRC bots connecting to different IRC servers. The same person also uploaded a PHP shell, and used it to download the configuration file of the CMS installed on the machine.

The fact that the attacker was not connecting through our HoneyProxy infrastructure but directly to our IP address was unusual, and attracted our attention. Searching backwards in our logs starting the date of the attack, we found out that

less than 24 hours before, an automated system with an User-Agent set to *'bingbot/2.0'* connected to one of our websites from another IP address from Englewood, Colorado, exploited a vulnerability and downloaded the osCommerce configuration file, which contains the real IP of our virtual machine hosting the e-commerce web application.

4.5.4 Privilege Escalation

Unique files ratio	1.7%
Estimated attackers ratio	2.2%

Privilege escalation exploits are among the oldest types of exploits in the computer security history, but are still among the most sought after, as they allow an attacker to gain administrator privileges and thus full control of vulnerable machines. Successfully executing a privilege escalation exploit on server machines used in a shared web hosting environment would make the attacker in the position to modify the files of every website hosted on the server, possibly allowing for mass exploitations of hundreds or even thousands of websites at the same time.

An example of such kind of attack hit our honeypots on February 9, 2012. An attacker with an Hungarian IP address uploaded a file called `mempodipper.c` to our machine hosting the web shells, and used one of the shells to try to compile its source code with *gcc*. The machine had no available compiler, thus, less than 5 minutes later, the attacker uploaded a pre-compiled ELF binary named `mempodipper`, and tried to execute it through one of the shells. We found this exploit to be for a very recent vulnerability, the CVE-2012-0056, published less than 20 days before this attack. At the time of the attack, the exploit for this vulnerability, titled *Linux Local Privilege Escalation via SUID /proc/pid/mem Write* was already publicly available [134]. However, the kernel of our virtual machines was not vulnerable to it.

4.5.5 Scanners

Unique files ratio	2.3%
Estimated attackers ratio	2.8%

This kind of activity is performed to find other local or remote vulnerable target websites that could possibly be exploited by the attacker. For example, FTP scanning, querying search engines using 'dorks', or trying to list all the domain names being hosted on the machine belong to this category.

A concrete example is the *trdomain.php* page, uploaded to one of our honeypots on December 26th, from a Turkish IP address. It contains a local domain name scanner, that pulls the domain names configured on the machine from the local configuration files (such as named.conf), gets their PageRank from Google, as well as their document root and their owner's username, and returns a web page with a list containing all this information. The title of the page is *'Domain ve User*

ListeLiyici | by W£ßRooT '; as of today, searching such title on the web still yields many results, showing that this kind of attack is very common and wide spread.

4.5.6 Defacements

Unique files ratio	28.1%
Estimated attackers ratio	27.7%

Attacks of this kind are among the most frequent ones on our honeypots. In this kind of attack, the attackers modify existing web pages on the honeypot, or upload new pages with the purpose of claiming their responsibility for hacking the website. Usually, but not always, the claims are accompanied by religious or politic propaganda, or by funny or shocking images. Many of the attackers performing such attacks even insert links to their personal website or Facebook page, where one can see they are mainly teenagers looking for fame and bragging in front of their friends.

One of the many defacements attacks that hit our honeypots happened around 8 pm GMT on the 6th of March. Somebody connecting from a German IP address found one of the hidden shells in our machine hosting the static website, and used it to edit one of the static html pages hosted on the machine. The code of the page was thus uploaded using copy-and-paste in a textarea provided by the web shell. The defacement page contained a short slogan from the author, an animated JavaScript text slowly unveiling a Portuguese quote, and a set of links to the personal Twitter pages of each member of the hacking crew, some of which had more than 1000 tweets and several hundred followers. Quickly looking at these Twitter profiles, we found out that all the members are actively posting their defacements on their profile pages. Apparently, they do so in order to build some sort of reputation. This is confirmed by the URL they posted as a personal webpage on Twitter, a web page from the `zone-h.org` website, reporting statistics about previous defacements of the crew. The statistics are quite impressive: at the time of writing the whole crew has claimed more than 41,600 defacements starting July 20, 2011, of which almost 500 are on important websites with high reputation (government websites, universities, multinational corporations, etc.).

Thanks to attacks like this we found out that it is common practice among attackers to advertise their defacements on publicly accessible 'defacement' showcases, such as the one on the `zone-h.org` website. It seems that some of these people are really in a sort of competition in order to show off their presumed skills at hacking websites, and our honeypot domains were often reported as trophies by several groups.

4.5.7 Botnets

Unique files ratio	28.1%
Estimated attackers ratio	27.7%

Several attackers, after exploiting our honeypots, tried to make our servers join an IRC botnet by uploading dedicated PHP or Perl scripts.

Two of the honeypot virtual machines, and specifically those with the most severe vulnerabilities, allowing attackers to upload and run arbitrary files on the server, have been set up to allow outgoing connections to port 6667 (IRC). We did so in order to monitor IRC botnet activity launched by an eventual attacker on our machines. We allowed connections only to port 6667, allowing thus only botnets running on the standard IRC port to connect to their management chat rooms. To avoid being tracked down by bot masters, every connection to the IRC port was tunneled through a privacy-protected VPN that anonymized our real IP address. No other outgoing connections were allowed from the machines, in order to avoid the possibility for our machines to launch attacks or scans against other hosts.

Our expectations proved to be correct, and we indeed logged several connections from our two machines to IRC command and control servers. The analysis of the packet traces showed some interesting information.

First of all, we were expecting IRC botnets to be quite rare nowadays, given the relatively high number of web-based exploit packs circulating on the black market. However, the analysis of the files that were uploaded on our honeypots showed an opposite trend, with about 200 distinct scripts launching IRC bots.

Another interesting observation is that, apparently, most of these IRC botnets are operated by young teenagers, as some IRC logs show. Some of the bot masters even put links to their Facebook or Twitter profiles in order to show off with their friends. Despite being run by youngsters, however, most of our connection logs show IRC rooms with hundreds to thousands of bots (the biggest IRC botnet we observed was comprised of 11900 bots).

While some logs showed us some of the bot masters attacking rivals on other IRC servers (which we considered a typical script-kiddie behavior), we were interested to see that these young people already deal with money and are able to use (and probably develop themselves) automated tools for searching on search engines and exploiting web vulnerabilities. We received a number of commands to perform DoS attacks, search engines scans using dorks, automatic mass exploitations, and instructions to report back usernames and passwords, as well as credit card credentials, stolen from exploited websites.

A final interesting finding, supported by the language used in the IRC logs and by an analysis of the IP addresses used for the upload of the IRC script, was that the majority of these IRC botnets were installed by users from South-Eastern Asian countries (mostly Malaysia and Indonesia).

4.5.8 Phishing

Unique files ratio	7.3%
Estimated attackers ratio	6.3%

Phishing is one of the most dangerous activities that online criminals perform nowadays. We found proof of many attempts to install phishing pages or phishing kits on our honeypots. This kind of activity is always profit-driven; the vast majority of phishing websites are replicas of online banking websites, but we also collected few examples of online email portal phishing and even a handful of web pages mimicking ISPs and airline companies' websites.

During the 100 days of operation, our honeypots collected a total of 470 phishing-related files, 129 of which were complete phishing packages (archives often containing a full phishing website installation, including images, CSS files, and the phishing scripts themselves). Surprisingly, Nigeria seems to be a very active country for this kind of attacks, with Nigerian IP addresses responsible for approximately 45% of the phishing attacks logged by our honeypots.

An interesting case was logged by our honeypots starting on March 27th. Analyzing the Referer header of the requests received by our websites, we found 4776 requests, from 1762 different IP addresses, reaching our pages with the referer set to the mail servers of `sfr.fr`, one of the major French ISPs. Inspecting the webserver logs, we found out that all the HTTP requests having a Referer from sfr.fr requested only two png images. Both files had been uploaded to our honeypots on the 24th of March; when the first hit from SFR arrived, the virtual machines had already been cleaned up several times, but we found the original version of the pictures in our snapshots of uploaded files. Surprisingly, the pictures showed a message resembling a regular communication from SFR's customer service. All the users that hit our honeypots with a Referer from sfr.fr had thus received a phishing email containing links to the two png files, and their web client was only trying to download and show them the contents of the email.

4.5.9 Spamming and message flooding

Unique files ratio	7.8%
Estimated attackers ratio	9.3%

Many users still seem to use spam as a technique to make profit on the Internet. Some of the scripts we found are indeed mailers, i.e., scripts used to send out spam to a large number of recipients in an automated way. Some other scripts were email or SMS flooders, that are instead used for launching DoS attacks.

Our honeypots collected around 600 such scripts. As an example, on February 21st, a script called *a1.php* was uploaded from a Nigerian IP address. This script is a highly customizable mailer, and allows sending spam to a list of recipients in plain text or HTML format, with many options. It can also be configured to log in to a remote SMTP server in order to send spam through an authenticated account,

and to disconnect and reconnect to the server after a certain threshold of sent emails is reached, probably with the purpose of avoiding bans.

4.5.10 Link Farming & Black Hat SEO

Unique files ratio	2.7%
Estimated attackers ratio	1.0%

Link farms are groups of web sites linking to each other, usually creating web pages with a very dense link structure, whose aim is to boost the search engine ranking of the web sites of the group. Black-hat SEO, instead, refers to using illicit or unethical techniques, such as cloaking, to boost the search engine ranking of a website, or to manipulate the way in which search engines and their spiders see and categorize a web pages. If we exclude automated posts on the forum web application, where a high percentage of posts contained links to link farming networks, this kind of behavior has not been observed very frequently on our honeypots.

An interesting attack that created a big amount of web pages on our honeypots was launched on March 19th. Somebody installed an fully functional CMS, comprising hundreds of static HTML pages, to one of our honeypots. All the generated pages were installed on the *images/rf/* subdirectory of our e-commerce web application, and contained Russian text, along with images, CSS and JavaScript files used for presentation purposes. This page structure seems to be generated through a blog or CMS creation engine, as all the pages have a very dense link structure and point one another using absolute links (that had been customized and contained our honeypot website's domain name). We expect this to be part of an attempt to create a link farming network, or simply to be a marketing campaign for some counterfeit goods, as most of the pages we analyzed were actually advertising the sale of replica watches.

Finally, on a smaller scale, we also saw some attackers creating pages with ads or inserting links to partner sites on their uploaded pages. The reason for this is still making profit out of ads, or improving their or their partners' ranking on search engines.

4.5.11 Proxying and traffic redirection

Unique files ratio	0.6%
Estimated attackers ratio	0.6%

Online criminals always look for reliable ways to hide their tracks, and as time goes by, it becomes more and more difficult to rely only on open proxy networks, the TOR network, or open redirection web pages to conduct malicious activities. In fact, these services are often overloaded with (malicious) traffic and as such have very bad average performances and are very likely to be monitored by the authorities. In this scenario, the possibility of tunneling traffic on infected hosts seems idyllic, as it is quite easy to turn a webserver into a proxy, and often webservers running on hosting providers premises have high bandwidths, making them

a very valuable target. We saw some attackers uploading proxy scripts or traffic redirection systems (TDS) to our honeypots, for the purpose of redirecting traffic anonymously (proxies) or redirecting users to malicious sources or affiliate websites (TDSs).

As an example, an archive of 504KB was uploaded on one of our honeypots on February 22, 2012. The archive contained a proxy tool called *VPSProxy*, publicly available at *http://wonted.ru/programms/vpsproxy/*; it is a PHP proxy fully controllable through a GUI client. Apparently, among all its features, if installed on more than one server, the tool makes it easy for the person using it to bounce between different connections. We believe tools like this can be very useful to criminals trying to hide their traces on the Internet.

4.5.12 Custom attacks

Unique files ratio	1.9%
Estimated attackers ratio	2.6%

This category groups all attacks that were either built on purpose for exploiting specific services, or that had no other matching category. For example, attacks in this category include programs whose aim is to scan and exploit vulnerable web services running on the server, such as the *config.php* script that was uploaded to one of our websites on April the 9th. This PHP script presents a panel for finding and attacking 9 of the most known Content Management Systems: if any of these is found on the machine, the attacker can automatically tamper with its configuration. The tool also contained other scripts to launch local and remote exploits.

4.5.13 DOS & Bruteforcing tools

Unique files ratio	4.6%
Estimated attackers ratio	2.9%

This category includes programs that launch Denial of Service or brute-force attacks against specific applications and services (e.g., bruteforcing tools for FTP or web services, UDP and TCP flooding scripts).

An interesting example of this kind of behavior was the email brute-force script that was uploaded to one of our honeypots on April 7, 2012. An IP address from Azerbaijan used a web shell to upload a file called *n.php* and a wordlist containing 1508 words, called *word.txt*. The *n.php* file, once executed, uses the cURL PHP libraries to connect to the `box.az` email portal and the uses the wordlist to brute-force the password for a specific username that was hard-coded in the program. Our honeypots actually logged the upload of *n.php* several times, to three different domains. The attacker tried multiple times to execute the script (10 times in 16 minutes) and to edit it (4 times) as if looking for an error in the code. In reality, the script traffic was simply blocked by our firewall.

4.6 Conclusions

This chapter described the implementation and deployment of a honeypot network based on a number of real, vulnerable web applications. Using the collected data, we studied the behavior of the attackers before, during, and after they compromise their targets.

This study allowed us to analyze, for the first time in academic literature, how and why attacks against common websites are carried out. We believe the results presented in this chapter provided interesting insights on the current state of exploitation behaviors on the web. On one side, we were able to confirm known trends for certain classes of attacks, such as the prevalence of eastern European countries in comment spamming activity, and the fact that many of the scam and phishing campaigns are still operated by criminals in African countries [45]. Pharmaceutical ads appear to be the most common subject among spam and comment spamming activities, as found by other recent studies [20].

On the other hand, we were also able to observe and study a large number of manual attacks, as well as many infections aimed at turning web servers into IRC bots. This suggests that some of the threats that are often considered outdated are actually still very popular (in particular between young criminals) and are still responsible for a large fraction of the attacks against vulnerable websites.

Chapter 5

Web Attacks from the User's Side

After analyzing how web attackers behave and how hosting providers handle compromises, it is time to turn our point of view on the target of each web attack: the user. Users are typically the final target of web attacks, as it is from their personal information that criminals are able to obtain financial gain.

However, while many aspects of web attacks have been carefully studied by researchers and security companies, the reasons that make certain users more "at risk" than others are still unknown. Why do certain users never encounter malicious pages while others seem to end up on them on a daily basis?

To answer this question, in this chapter, we present a comprehensive study on the effectiveness of risk prediction based only on the web browsing behavior of users. Our analysis is based on a telemetry dataset collected by a major AntiVirus vendor, comprising millions of URLs visited by more than 100,000 users during a period of three months. For each user, we extract detailed usage statistics, and distill this information in 74 unique features that model different aspects of the user's behavior.

Once we extract all features, we perform a correlation analysis to see if they are correlated with the probability of visiting malicious web pages. Afterwards, we leverage machine learning techniques to provide predictions for users that are exposed to risk. The results of experiments show that it is possible to predict with a reasonable accuracy the users that are more likely to be the victims of web attacks, only by analyzing their browsing history.

5.1 Introduction

A large amount of research has been conducted on the tools and techniques adopted by attackers, to automatically identify and mitigate software vulnerabilities, or to protect web browsers from exploitation. Despite this effort, the percentage of web pages that are either malicious or that have been compromised to serve malicious content is steadily increasing [55,116,127]. Even though this is certainly an alarming phenomenon, these global figures are computed on the entire Internet

population, and therefore fail to express what is the real risk for a single user to encounter a malicious page on her daily activity. The increasing number of dangerous sites does not necessarily affect everyone in the same way. For instance, it is possible that the majority of users only navigate in "safe neighborhoods" where malicious pages are still extremely rare. In this case, it should be possible to associate to each user, based on her usual online behavior, a certain *risk* profile. In other words, there should be a correlation between the *browsing habits* of a user and the probability of her visit to potentially harmful pages. This scenario is particularly attractive in the area of cyber-insurance [9], in which user profiling is an important step toward an accurate risk evaluation. For example, in the physical world insurance rates are normally computed based on a risk classification. For instance, car insurances are more expensive in large cities or for inexperienced drivers – because this conditions are known to be positively correlated to the probability of car accidents. Unfortunately, an equivalent measurement of risk factors in the virtual world is still missing.

While the hypothesis of a correlation between the risk and the browsing behavior is reasonable, this does not necessarily imply the presence of any causality relationship. It has to be underlined, however, that our work focuses only on the analysis of *voluntary* user browsing activity. We thus have no visibility over URL visits that do not originate from user actions on an Internet browser (such as visits to pages that are part of Command and Control infrastructures, or URLs visited directly via non-standard applications or malware).

When dealing with the analysis of user browsing behaviors, there are also other factors that one has to take into account. For instance, independently from their daily activity, users are often socially engineered to click on links sent to them over email. As a consequence, it is possible that other attributes such as the user experience in computer science, as discussed by Onarlioglu et al. [84] could be more important to determine the risk factor of a user than her browsing habits.

Unfortunately, few works have tried to answer this question and understand if there are certain behaviors or certain characteristics that may influence the probability of users to visit malicious web pages. As discussed in Chapter 2.3, some works have tried to answer similar questions by performing field studies on the computer usage of a limited number of subjects [61]. Others have speculated whether certain behaviors may be related to higher chances of being compromised, such as the relation between browsing porn sites and being subject to infections or malicious web master practices [130]. However, no study has so far been general enough to build user profiles and analyze this information in order to assess if there is any relation between specific user habits and the fact of visiting malicious web pages.

In this chapter, we conduct the first comprehensive study in this area by using the telemetry data collected by a popular antivirus company. In particular, we analyzed the webpages visited by 160,229 users over a period of 3 months (92 days). Using anonymized information, we first identified two classes of users: the *safe* ones who never visited malicious webpages during our experiments, and the

ones *at risk* who visited several malicious sites in the same timespan. Our goal was to see what kind of behavior can be used to differentiates the two classes. For this reason, we identified and extracted 74 attributes that can be used to summarize the user browsing behavior, and we correlated each of them with the users' class.

Our experiments confirm that the volume of user activity is one of the best indicators for the level of risk. The more pages a person browses everyday, and the more diverse is the set of pages, the more likely she would be to come across a malicious website. We also show that malicious pages are more likely to be encountered during the weekend and that people in the risk class are more active during the night than users who belong to the safe class. Looking at the website categories, we found that some of them – such as adult content and shortened URLs – are positively correlated to the probability of being at risk. Finally, the results of the experiments we performed indicate that it is possible to combine all this information and train a classifier to predict whether a user is at risk of infection, just by analyzing her browsing profile.

5.2 Dataset and Experiments Setup

We performed our analysis over a dataset we obtained through a collaboration with Symantec, that gave us access to part of the telemetry data collected by the WINE platform. This data is obtained from clients that voluntarily opt-in to let their computers share information on usage statistics and encountered threats. AntiVirus (AV) vendors typically employ this kind of client feedback with the purpose of identifying new threats and improve their products and services.

The dataset we obtained consists of a 3-month snapshot of the web browsing history of a subset of clients that had opted in to allow the company collect information on their browsing activity. The dataset covered all the web requests issued by approximately 160,000 distinct client machines in a three-month period, from August 1st, 2013 to October 31st, 2013. This consisted in a total of 202,306,687 URL visits, covering a total of 37,797,151 distinct URLs. The data collected by Symantec included only URLs of websites visited through the HTTP protocol. All information was provided in an anonymized form, and no private client information was available to us, with the exception of the client's country. It is important to note that customers who agreed in sharing their browsing history are aware that the company stores this information in anonymized form, and that client identifiers are anonymized too. This means it is not possible for the AV company to link back the collected data to the client from which the requests originated. The specific fields we have analyzed in our study include only the unique client identifier, the timestamp of the visit, and the URL of the web site. Moreover, to further improve the privacy of the users, we anonymize each URL by removing the path and eventual URL parameters – limiting our analysis to the fully qualified domain name.

Since our main goal is to perform a statistical analysis of the dataset, we focused our study on those clients who visited at least 100 web pages during our

timeframe. This prevents clients whose information has low statistical significance to pollute, or bias, our measurements. We believe the threshold of 100 pages over 3 months to be conservative enough to include almost all regular user behavior, while excluding those machines that are only sporadically used to browse the Web. It can be noticed that, with our dataset, visiting less than 100 web pages over three months means basically opening one URL per day or less: it would be very difficult to build a user profile based on such a limited browsing history.

5.2.1 Data Labeling

To be able to estimate if there is a correlation between risk and user behavior, we first need to define what the definition of *risk* is for our study. As explained in Section 5.3, we define the risk categories by setting an experimentally chosen threshold for the number of times a user visits distinct malicious URLs or domain names during the experimental period of 3 months.

We constructed our labeled set of malicious URLs from URLs detected to be malicious either by the Norton SafeWeb service [118] or by Google SafeBrowsing [105]. We further collected malicious domain names from several public services that provide a list of domains involved in various malicious activities, such as dropzones, drive-by-download, phishing, and scam web sites. In particular, we built the list by merging information collected by malware domain list [64], abuse.ch [4] and malc0de [69].

All this information allowed us to label each URL in our dataset as either *Benign*, *Malicious*, or hosted in a *Blacklisted* domain. We decided to keep this last class separated from the malicious URLs because domains have a larger granularity and therefore provides a less accurate classification. Please note that the labeling phase was performed in an automated way on Symantec's servers, thus prior to discarding the full URL path. Once the matching was completed, the rest of the analysis only operated, in an aggregated form, on the anonymized URLs.

5.2.2 Risk Categories

One of the goals of this work is to answer the question of whether it is feasible to identify a category of people that, while surfing the Internet, incurs in a higher chance of visiting malicious web pages, when compared to other users. To be able achieve this goal, we first need to separate users in different risk categories.

Following a classical insurance approach, we separate users based on their past experience. With a good approximation, users that never ended up visiting a malicious page during our three-month observation period can be considered *safe* users. We noticed, however, that the contrary is not necessarily true. Indeed, given the high number of factors contributing to the maliciousness of a website and the delay in updating popular blacklists, misclassifications are not too rare. For example, it happens even to trusted websites to serve malicious ads or to become victims of DNS hijacking attacks [122]. Thus, when looking at our classification scores, a

Value	Risk Category		
	safe	***uncertain***	***at risk***
Total number of visited URLs	743	1386	2411
Distinct URLs visited	231.3	452.4	873.7
Average number of URLs visited per day	16.8	23.8	36.6
Distinct URLs visited per day	5.8	8.5	14.0
Total number of malicious URLs visited	0	0.78	8.4
Total number of blacklisted domains visited	0	2.44	8.5
Distinct number of malicious URLs visited	0	0.5	4.0
Distinct number of blacklisted domains visited	0	0.9	2.8
Percentage of malicious URLs	-	0.14%	0.71%
Percentage of blacklisted domains	-	0.32%	0.4%
Number of users	80128 (50%)	49127 (31%)	30974 (19%)

Table 5.1: **Average** values of different indicators, for users in the three risk categories.

certain noise margin has to be taken into account. To handle this problem, we define a user to be *at risk* if she visited at least two distinct malicious URLs, or at least three blacklisted domains over the 3-month period. Again, the reason to use different thresholds for URLs and Domains is that the latter have a lower granularity and thus a higher probability of misclassification.

We put users who do not belong to the previous two categories into an *uncertain* middle category. For instance, the fact that a person visits a single dangerous URL over three months (with multiple visits to the same URL counting as one) may be just due to an error in classifying the URL. This is not sufficient for us to conclude that the user has a risky browsing behavior.

Table 5.1 shows the average number of different types of URLs visited by each category of users. Users who are "not at risk" appear to browse over five times less malicious URLs than *at risk* users. This means that typically, as the table shows, users in the *uncertain* category end up on malicious websites less frequently than *at risk* ones. Another clear difference is that *at risk* users typically visit more pages than other categories of users, and this factor may be related to the chance of ending up on malicious websites (we will discuss this hypothesis in more detail in Section 5.5). This is also valid in relation to the "variety" of visited websites, since for *at risk* users the average number of *distinct* URLs, and *distinct* URLs per day are about twice as much as the same values for the *uncertain* group. Finally, the table highlights that roughly one user out of five in our dataset belongs to the *at risk* category. If we consider the total number of users who are exposed at least once to malicious websites, then, this ratio increases to half of the entire user population. This is more than what found by a recent study on Australian customers by one

major AV company [56], that observed that one customer every eight was exposed to web threats.

5.3 Geographical and Time-based Analysis

This section describes the analysis we have conducted on the dataset we obtained from Symantec, and provides details on user habits, time trends and more than 70 other features that we extracted to model the users' behavior.

5.3.1 Daily and Weekly Trends

Question: *Is there a time of the day, or day of the week, when users are more likely to visit malicious web pages?*

We start our analysis of users' browsing behavior by looking at the weekly and hourly trends emerging from our dataset. First of all, Figure 5.1 shows that, as we expected, people surf less during the weekend. This trend is valid, with slight variations, all over the world, and country-wise daily trends do not differ much between each other. One can notice that there is a slight but significant increase in the percentage of malicious URLs visited during the weekend, compared to the trend of malicious hits during the rest of the week. This amounts approximately to a 10% increase in the chance of incurring in a malicious URL during the weekend, compared to the risk of doing so between Monday and Friday. The average *p-value* given by the Wilcoxon Signed Rank Test [129] when comparing the two distributions is of 6.44×10^{-7}, which shows the difference is indeed significant. As often found in literature [113], we consider to be statistically significant those differences showing computed *p-values* of less than 0.05.

Figure 5.2 shows instead the hourly trends for website visits, split between the two categories of users. As the hourly trends show, browsing trends for the *safe*, and *at risk* users do not differ much, even though *at risk* users are slightly more active during the night and less active in the morning. The statistical significance of these variations if confirmed by means of the Wilcoxon Signed Rank Test, that returned *p-values* significantly lower than the typical 0.05 significance level (e.g., the *p-value* of the test between 1 and 2am was of only 2.2×10^{-16}). However, the fact that users in the *at risk* category spend more time on the Internet at night does not imply that it is more risky to browse after midnight. Therefore, in Figure 5.3, we look at the same hourly trends, but from the point of view of the URLs instead of users. In this case, the graph shows that hits on blacklisted domains are higher than other malicious hits between 9pm and 2am, and lower than others during business hours. Hits on malicious URLs seem instead to be prevalent in the afternoon, between 3pm and 8pm. Again, the signed rank test confirmed the differences in these distributions to be statistically significant, producing *p-values* always below the significance level of 0.05.

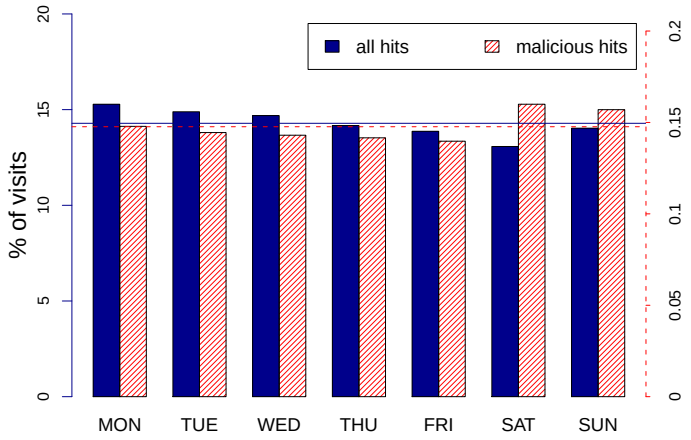

Figure 5.1: Global daily distribution of URL hits. The percentage of malicious hits is expressed as a fraction of the total hits on the same day.

Overall, these results confirm what found by a recent report on the Australian customers of a known security firm [56]. Indeed, as the mentioned study shows, also our analysis of time trends shows an increase in the percentage of malicious hits during nights and weekends. We are thus able to confirm that trends that have been reported on Australian users still hold when observing browsing statistics of users from all around the world.

5.3.2 Geographical Trends

Question: *Is there a set of countries where users are more likely to visit malicious web pages, when compared to others?*

Our dataset contains information about clients located in 167 different countries. Table 5.2 summarizes some general statistics for those countries for which we have at least 1000 users. Simply by looking at the outliers (emphasized in bold in the table), one can notice several interesting trends.

For instance, Japan appears to have by far the lowest per-user ratio of malicious hits, and the lowest percentage of users at risk. However, the absolute value of malicious pages visited by Japanese users is in line with the ones of other countries. Percentages are just lower because average users in Japan browse twice as many pages as their counterparts in other countries. At the other end of the scale, we have several Mediterranean countries (notably France, Spain, and Italy) that share similar high values of several risk indicators. These countries have a percentage

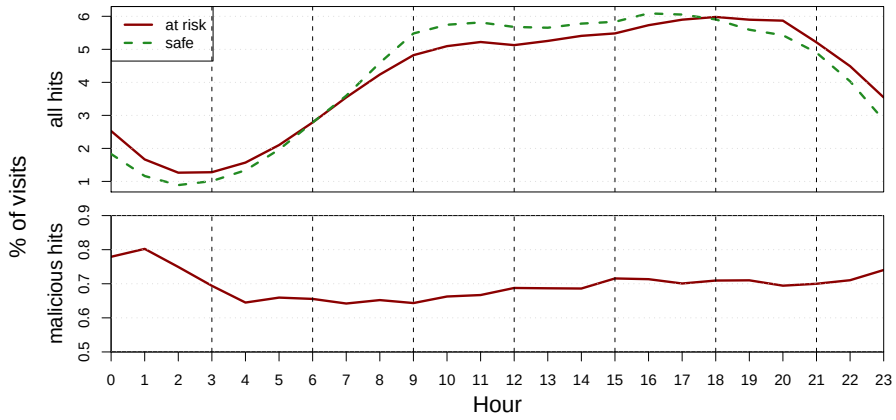

Figure 5.2: Hourly trends for, respectively, all the hits (upper) and malicious hits (lower) in our dataset. Malicious hits are expressed as percentage of the total hits for the same category of users, in the given hour.

of *a*t risk users ranging between 27% and 29%, three times higher than Japan, and approximately twice as much as other top countries.

Finally, the last column of the table shows the average number of languages of web pages visited by users in each country. As it can be expected, users from English speaking countries appear to visit pages in a limited number of languages compared to those visited by users in non-english speaking countries. In average, over the 3-months period, users in English speaking countries appeared to browse pages written in less than 4 different languages, while users from other countries visited pages in an average of 5.3 different languages. This fact seems however not to have any clear relation with the percentage of *a*t risk users in each country.

5.4 Feature Extraction for User Profiling

After looking at time patterns and geographical trends, we decided to focus in more detail on the behavior of users. The basic idea motivating this work is that we expect users that belong to the risky category (i.e., those that regularly visit malicious web pages) to behave differently, when browsing the Internet, than users who are safe. We thus define a user profile as a sort of a multi-dimensional template such that we can characterize the behavior of each user group. We model each user profile by using a combination of 74 unique variables, or *features*, designed to precisely capture many aspects of a user's browsing habit.

In the following paragraphs, we will describe the different categories that compose our features set, based on which aspect of the user behavior they are meant to represent.

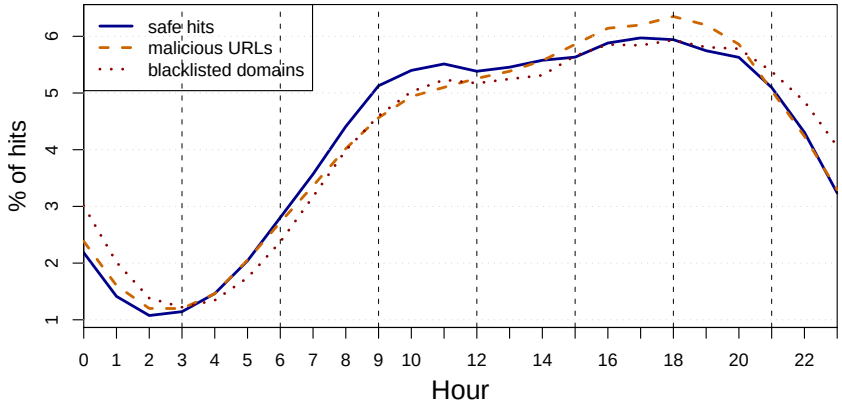

Figure 5.3: Hourly global trends for all hits and malicious hits in our dataset, showing also trends for the two separate sources of malicious hits

How Much a User Surfs the Web The first set of features are designed to capture the volume of user activity. The rationale behind this is that, as common sense suggests, the more time a person spends browsing the Web, the more likely she would be to encounter a malicious web site. For instance, this category of features includes the number of total (*hits*) and unique (*distinct_urls*) hits over the three month period, the average number of hours (*hours_per_day*) and web pages (*hits_per_day*) visited in a day, and the percentage of days in which the client was active during the period of the experiment (*days_active_perc*).

In Which Period of the Day a User is More Active When looking at time trends (see Section 5.3.1), we noticed that the distribution of malicious URLs is proportionally higher during the night and in the weekend. Therefore, we added a set of features to capture this difference. In particular, we introduced the percentage of the client's hits issued, respectively, during night time (*hits_night_perc*), business hours (*hits_bh_perc*), and the evening (*hits_evening_perc*). We consider midnight to 6 am as night, 6 am to 7 pm as business hours, and 7pm-midnight as evening. These time windows were chosen in order to be as conservative as possible in the evaluation of business hours, as these differ significantly between countries in the world, and thus we may otherwise wrongly categorize this value.

How Diversified is the Set of Websites Visited by a User Another possible root cause for being at risk when browsing could be related to browsing a very broad and diversified class of web sites as this might increase the likelihood of landing on a malicious page. We model this aspect by counting the number of visited hostnames (*hostnames*), domain names (*domains*), and the number of visited unique top level domains (TLDs) (*tlds*).

Country	Users	% users *at risk*	Average hits on		Visited Pages			# languages
			malicious URLs	blacklisted domains	total	distinct	domains	
US	67967	20.8	2.2 (0.22%)	2.0 (0.15%)	1250	422	194	3.6
UK	26204	17.8	1.5 (0.16%)	2.0 (0.16%)	1097	379	183	4.2
JP	16556	**10.0**	1.1 (**0.05**%)	3.1 (0.14%)	**1989**	**641**	205	3.8
CA	6798	20.9	2.0 (0.22%)	2.4 (0.17%)	1214	387	186	3.8
AU	6107	16.4	1.5 (0.17%)	1.5 (0.15%)	1007	343	173	3.7
DE	5606	22.3	2.0 (0.20%)	2.6 (0.23%)	1042	366	192	4.9
FR	4566	**29.1**	2.8 (0.27%)	3.3 (0.27%)	1127	390	209	4.5
NL	3415	15.9	1.1 (0.12%)	2.3 (0.21%)	1009	361	195	5.2
ES	1842	**28.3**	2.4 (0.23%)	3.9 (**0.33**%)	1121	391	200	**5.7**
SE	1755	15.3	1.9 (0.17%)	1.9 (0.14%)	1049	327	167	6.4
IT	1665	**27.4**	1.8 (0.18%)	**7.0** (**0.69**%)	1097	350	186	5.4
BE	1454	21.3	2.2 (0.21%)	2.5 (0.20%)	1126	396	208	5.5
NO	1208	**11.8**	1.1 (**0.10**%)	2.5 (0.11%)	1219	341	166	**6.1**

Table 5.2: **Average** values of several indicators, for users in the top 13 countries appearing in our dataset.

Among these features we also consider the percentage of distinct URLs calculated over the total number of hits for the given user (*distinct_urls_perc*) and the percentages of unique hostnames and of unique domain names over the total number of hits (*distinct_hostnames_perc*, *distinct_domains_perc*). The purpose of these features is to estimate if the user tends to revisit the same set of URLs or websites, or instead appears to browse a more diversified set of web sites.

Finally, we measured the number of languages (*n_languages*) of the web pages visited by a certain user. The language of websites was obtained from the same service we used to obtain the category of web pages, as explained in the following paragraph.

Which Website Categories the User is Mostly Interested in One of the main characteristics of a user profile is the categories of the visited web pages. To label each URL with the corresponding category, we used an internal website categorization system from Symantec. This service was designed to apply a set of heuristics to extract categories and languages from the URLs visited by the AV customers. Unfortunately, website categorization being based on heuristics, in some cases the categorization engine was able to retrieve only the main language used in the web page, but failed to properly capture the category, or vice versa. Therefore, we complemented the company database by using a number of publicly available website categorization services such as Alexa [1] and Open Directory Project [90], and a number of lists of known URL shorteners, bittorrent web sites, one-click hosting providers and porn websites [65, 119, 121, 131]. We employed these lists to complement the heuristic service provided by Symantec, as it is common belief that visiting websites from these categories yields to higher chances of being infected by malicious web code. As a result, we were able to cover 76% of the web sites in

our dataset (96% for the Alexa top 10,000 domains). The language coverage was instead 77% overall, and 70% for domains in the Alexa top 10,000.

Once website categorization phase was completed, we extracted a number of features to incorporate features that are extracted from the category information in the user profile. For example, we reported the percentage of activity in each of the following 8 categories: business websites, adult, communications and information search, general interest, hacking, entertainment and leisure, multimedia and downloading, uncategorized (sites for which we were not able to obtain a category).

Computer Type The main aspect we want to capture with this class of features is the difference between office and personal computers. The assumption we make to identify office computers is that computers that do not show any activity during the weekends are very likely to be office computers. We label all computers that are silent during the weekend as office and the others as personal computers (*work_pc*). In addition, for personal computers, we also compute the percentage of activity during the weekend (*hits_we_perc*).

The remaining features that are extracted to characterize the computer type are computed using properties of the anonymized IP addresses of the devices. Note that we do not have access to the absolute value of the IP addresses and to the name of the Internet Service Providers (ISP) they belong to. The AV company keeps the hashed values of IP addresses and their corresponding ISPs such that it is possible to calculate their distinct numbers (*n_ips* and *n_isps*). The final feature in this category is the number of countries from which the user appeared to be surfing the web from (*n_countries_user*). By using features, we aim at representing the user's mobility, and helping to assess whether a person is browsing the Internet from a static IP address or a dynamic one.

How Popular are the Websites Visited by the User This set of features are computed to model how common the websites visited by a user are, under the assumption that malicious pages are more commonly found in less popular sites.

The first indicator we look at is the percentage of `.com`, `.net` and `.org` top level domain (TLD) hits (*hits_comnetorg_perc*) that appear in each user's browsing history, and the number of visited URLs that belong to other TLDs (*no_comnetorg_tlds*). We also extracted a number of features related to the Alexa ranking of domains [2], namely the total number of hits and distinct websites visited in the Alexa top 500 (*n_hits_top500*, and *n_dist_sites_top500*), the total number of hits and of distinct websites visited in Alexa's top one million (*n_hits_top1M*, and *n_dist_sites_top1M*) and the number of hits and of distinct websites visited out of Alexa's top one million list (*n_hits_noAlexa*, *n_dist_sites_noAlexa*). All these features are computed both as absolute numbers and as percentage among all web sites visited by the user.

How Stable is the Set of Visited Pages To conclude our features set, we modeled the variability of a user's browsing activity. The rationale in this case is that users

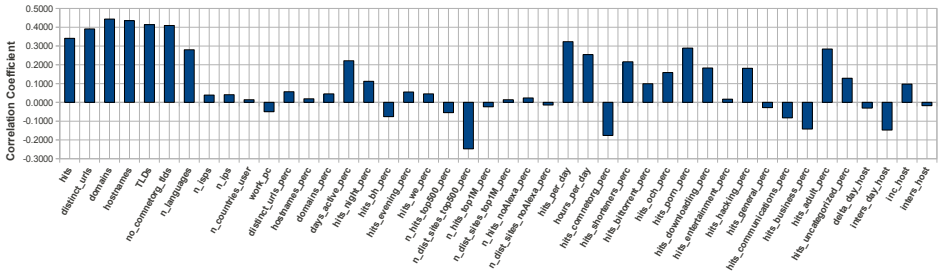

Figure 5.4: Spearman's Correlation Coefficient between user profile features and being *at risk*.

who always visit the same set of pages are less at risk than users who change their targets very often. In particular, it is possible that users are mostly exposed to malicious pages when they deviate from their usual interests and temporarily browse web sites they do not know very well. In order to obtain these features, we performed a one week training over the web browsing history of each client in our dataset. We thus recorded, for each client machine, the set of web sites visited during its first 7 days of activity (*training set*). For every other day, we recorded the set of visited websites and their intersection and difference with the *traning set*. We averaged these values, and obtained the average percentage of common web sites between the daily browsing session and the initial 7-day training period (*inters_day_host*), as well as the percentage of new web sites – not visited during the 7-day training window – browsed in average every day (*delta_day_host*). We also recorded, for each client machine, the whole set of web sites visited during all its browsing activity, and calculate its intersection with the *training set*. The size of this intersection is then scaled by the size of the training set to obtain *inters_host*, the percentage of hosts visited after the training period that cover the initial *training set*. Finally, we also calculate a measure of the increment in the number of web sites visited by the client during its entire browsing history, compared to the number of web sites in the initial *training set* (*inc_host*).

5.5 Evaluation

In this section, we present the results of our analysis. We first evaluate if any of the 74 features we have presented in the previous section are correlated to the fact that a user belongs or not to the *at risk* category. Afterwards, we build on top of these results to see if it is possible to use these features in a classifier, to predict the risk class of a user given her behavior.

Feature	At Risk	Safe	Percent Difference
hits	2411	742	106%
distinct_urls	873	231	116%
domains	331	88	116%
hostnames	388	108	113%
TLDs	17.5	7.9	76%
no_comnetorg_tlds	14.6	5.2	94%
n_languages	5.4	3.4	45%
days_active_perc	0.66	0.46	36%
hits_night_perc	0.09	0.07	27%
n_dist_sites_top500_perc	0.12	0.18	38%
hits_per_day	36.5	16.8	74%
hours_per_day	4.7	3.1	41%
hits_shorteners_perc	0.0034	0.0017	67%
hits_och_perc	0.0030	0.0018	50%
hits_porn_perc	0.0282	0.0112	86%
hits_downloading_perc	0.051	0.033	42%
hits_hacking_perc	0.0002	0.0000	113%
hits_business_perc	0.147	0.220	39%
hits_adult_perc	0.144	0.042	109%
inters_day_host	0.125	0.190	41%
inc_host	11.7	8.0	38%

Table 5.3: Comparison of the **average** values of certain features for *safe* and *at risk* users. Only features having a percentage difference greater than 25% are shown.

5.5.1 Feature Correlations

In the first part of our experiments, we extracted the feature values for all the users in our dataset and we used them to perform a correlation analysis with the risk class.

To start with, we compared the average values of each feature computed on *safe* and *at risk* users. While for the majority of them the difference was relatively small, in 22 features there was a difference of more than 25%. Those features, and the respective average values, are summarized in Table 5.3. The fact that several features clearly show up to a threefold increase between the activity of *safe* users and users *at risk*, suggested us to look at the correlation of these variables in more detail.

The correlation analysis we have adopted is based on the value of the Spearman's correlation coefficient [37]. Spearman's correlation is a statistical measure of the strength of a monotonic relationship between paired data. This coefficient by design is constrained between -1 and 1. While -1 indicates very strong negative correlation and +1 very strong positive correlation, the values close to 0 denote the absence of a monotonic relation between variables. We chose to employ the

Spearman's rank correlation because, unlike the Pearson's correlation coefficient, it does not require a normal distribution in the dataset.

After calculating the Spearman's correlation coefficient, we tested the confidence of the obtained results by performing a standard significance test. As already discussed in Section 5.3.1, we consider that the correlation value of a feature is statistically significant if the computed *p-value* (i.e, significance level) is less than 0.05. A set of selected features for which the *p-value* was under this threshold is summarized in Figure 5.4.

Note that in the literature [37], a Spearman's coefficient value below 0.20 is normally considered an indication of a *very weak* correlation. Similarly, values between 0.2 and 0.4 are considered *weak*, between 0.4 and 0.6 *moderate*, between 0.6 and 0.8 *strong*, and above 0.8 *very strong* indication of a correlation between the variables. As it can be clearly seen in Figure 5.4, most of the features we have extracted have weak or no correlation with the fact that a user is at risk.

In the weakly correlated category we find features related to the amount of daily web activity (hits and hours per day), the number of porn and adult websites visited by a user, the number of languages, and an inverse correlation with the percentage of visited websites falling in the top Alexa 500. In the moderate correlation interval we find again some absolute measures of the amount of URLs, domains, and hostnames visited by a user. Moreover, and more interestingly, we also find a correlation between being at risk and the number of web pages with a TLD different from `.org`, `.com`, and `.net`.

Not surprisingly, these results indicate that the more a user surfs the Internet, the more she might be exposed to the risk of encountering a malicious page. The category does not seem to matter much, with very little correlation found with the percentage of usage of URL shorteners, downloading, and hacking websites – and a small negative correlation with the percentage of business sites. The only exception, as discussed in more detail in Section 5.6, is the higher correlation with adult and porn categories.

5.5.2 Predictive Analysis

The results we obtained from the correlation analysis show that some of the features we examined, although not very strongly, have some mild correlation with the fact that a user is *at risk*. Therefore, the same footers might be helpful as well to predict whether a user is at risk or not. Motivated by this assumption, we have generated a number of prediction models leveraging state-of-the art machine learning techniques. Before opting for Logistic Regression [8], we experimented with many other machine learning approaches including decision trees [62, 96], support vector machines [18], and Bayesian classifiers [132]. In our tests, logistic regression achieved the best results in terms of accuracy and false positive rates.

Logistic regression is a probabilistic statistical classification model that aims at predicting a category from features presenting either continuous or discrete values [8]. Compared to other classification methods, one of its advantages is that it

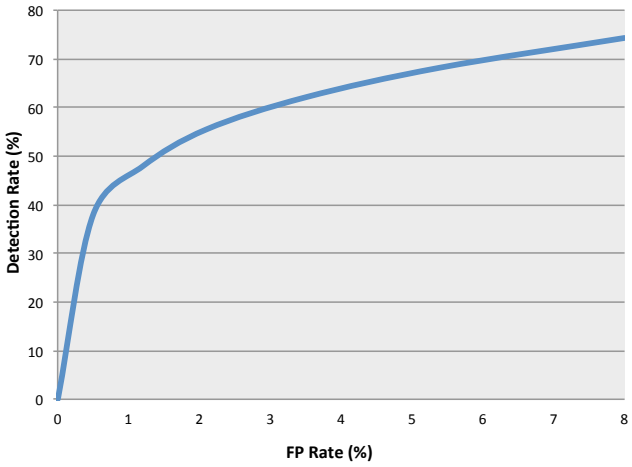

Figure 5.5: ROC Curve of the risk class classifier applied to the entire dataset

does not explicitly require features that are not correlated with each other. Moreover, when new data is available, logistic regression can efficiently update the models with the new input.

The Receiver Operating Characteristic (ROC) curve summarizing the true and false positive rates of our classifier, applied to the entire dataset, is shown in Figure 5.5. The curve has an area of 0.919. For instance, if used to detect *at risk* users, the system can be tuned to have a detection rate of 74% (i.e., users *at risk* properly classified as *at risk*) with 8% false positives (i.e., users not at risk misclassified as being *at risk*). These results, both in terms of the best classifier algorithm and of the area under the ROC curve, are in line with what has been measured in previous studies about the precision of classification algorithms for financial risk prediction [88].

Since, as explained in Section 5.3.2, the distribution and behaviors of users' risk classes are different in different countries, we decided to retrain our classifier on each country in isolation. The results are quite similar to the overall results, with the only exception of Japan, for which the system was more precise – achieving 73% detection with 1.9% false positives, and an area under the ROC curve of 0.958, as shown in Figure 5.6.

5.6 Discussion and Lessons Learned

Our experiments confirm the finding of a recent study by Levesque et al. [61] regarding the fact that the more websites a user visits the higher is her exposure to threats. However, we reach a different conclusion regarding the correlation of browsing adult content. Levesque et al. found that the amount of sport or Internet infrastructure websites visited by a user are more related to the fact of being in-

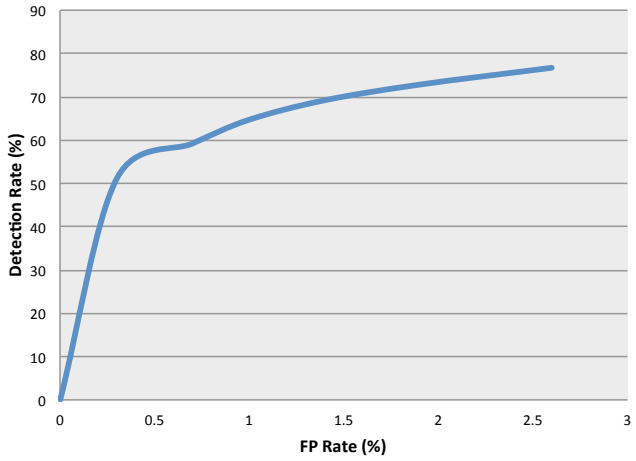

Figure 5.6: ROC Curve of the risk class classifier applied to the Japanese users only

fected than the number of porn websites [61]. However, while the absolute number of porn website may not be a good indicator, the percentage of time spent browsing porn seems to be a better feature. In fact, even though browsing adult web pages may not be a risky activity per se, from our results it looks like people who browse (in proportion) more porn and adult websites are more likely to also visit malicious pages in their daily activity. To better investigate this phenomenon, in Figure 5.7 we plotted the percentage of users at risk for different ranges (using a decile split) of the *hits_adult_perc* feature. The graph clearly shows that the percentage of safe users decreases as the ratio of adult hits increase.

Similar trends can be plotted also for the other variables. For instance, Figure 5.8 presents an even clearer picture of the relationship between the number of unique top level domains and the steep increase in the percentage of users at risk. While the large majority of safe users lie in the first half of the decile plot (more than 50% of the total number of safe users actually lie in the 1st decile – visiting less than 8 TLDs in the 3-month period), their percentage drops to less than 20% in the last three deciles of the plot, corresponding to users that visit at least 21 different TLDs. This is yet another confirmation that the variety, and not just the number, of the pages visited by a user is a strong indicator for its risk factor.

As mentioned in Section 5.3.2, also the geographical location of a user is very important. Citizens of certain countries (such as Norway and Japan) seem to be more careful in their browsing habits compared to their peers located, for example, in Italy or Spain.

While some of the features we used in our experiments were already discussed in previous works, the main finding of our study is the fact that by extracting and combining a much larger number of features from the URLs visited by a user in a certain period of time, it is possible to train a classifier to predict the risk class a user

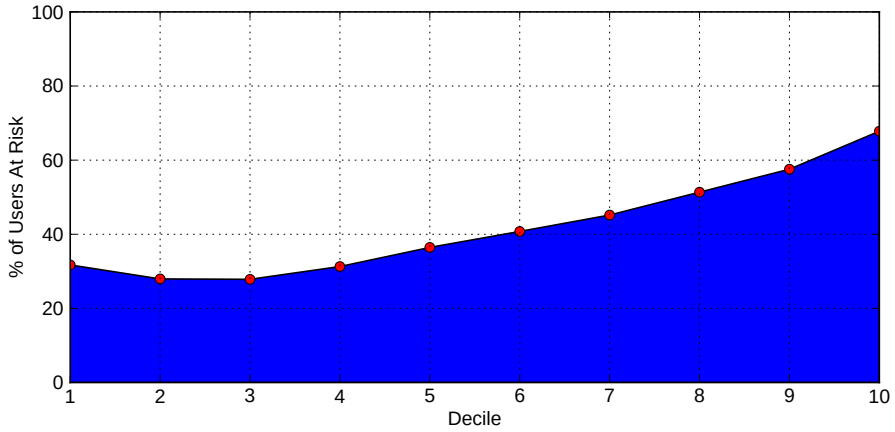

Figure 5.7: Decile plot for *at risk* users with respect to the percentage of hits on adult web sites.

belongs to. This is a very interesting finding from several points of views. First, our results can be obtained by looking only at HTTP requests, without any access to the end-user devices. This allows companies (or even ISPs) to silently profile their users and even combine their risk class into an aggregated risk factor at a company or network level. Second, while still far from perfect, the accuracy of the extracted models is sufficient to be used in a risk prediction scenario. Comparable models are used everyday to compute the risk associated to financial operations [88], such as when processing credit card requests. This opens the door to a simple yet effective way to implement a cyber-insurance mechanism that rewards users who show a safe browsing profile.

5.7 Conclusions

In this chapter, we have analyzed the behaviors of users who end up being victims of web attacks, and have presented a first step towards the prediction of users' risk when browsing the Internet. Our in-depth analysis of a large telemetry dataset collected by one of the major AV companies allowed us to gain a number of insights on the relation between users' browsing habits and their chances of visiting malicious web pages on the Internet. For example, we have been able to confirm some known trends, such as the fact that browsing the web late at night and during weekends is typically correlated with higher chances of ending up on malicious web sites. Another general trend confirmed by our work is that, in general, the more a user surfs the Internet, the more her chances are of ending up in some "unsafe neighborhood".

We have also been able to shed some light on whether profiling can be effectively used as a basis for predicting the risk for a user to end up on malicious web

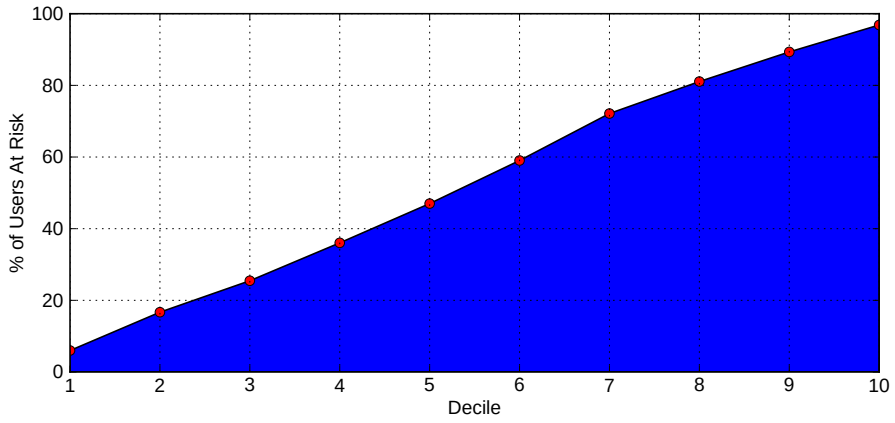

Figure 5.8: Decile plot for *at risk* users with respect to the number of different TLDs visited.

sites. By employing machine learning, we showed that user profiling could actually be employed, at least to some extent, in predicting the class of risk for a user on the web, similarly to what is currently done in the field of insurances and financial risk prediction.

Chapter 6

Detection of Malicious Web Pages by Companies and Researchers

When a web attack happens, the last actors to become involved in it are the security companies, or security researchers, trying to understand what happened and why. Also, security companies are spending a lot of efforts in trying to detect websites hosting malware as they pose a serious threat to all web users. In particular, this final chapter presents the point of view of a security company that has to deal with the analysis of a large volume of web pages on a daily basis.

As we know, in the last few years, researchers have developed a number of systems that analyze web pages for the presence of malicious code Most of these systems use dynamic analysis, i.e., they run the scripts associated with a web page either directly in a real browser (running in a virtualized environment) or in an emulated browser, and monitor the scripts' executions for malicious activity. While the tools are quite precise, the analysis process is costly, often requiring in the order of tens of seconds for a single page. Therefore, performing this analysis on a large set of web pages containing hundreds of millions of samples can be prohibitive.

One approach to reduce the resources required for performing large-scale analysis of malicious web pages is to develop a fast and reliable filter that can quickly discard pages that are benign, forwarding to the costly analysis tools only the pages that are likely to contain malicious code. This chapter describes the design and implementation of such a filter. The filter we propose, called *Prophiler*, uses static analysis techniques to quickly examine a web page for malicious content. This analysis takes into account features derived from the HTML contents of a page, from the associated JavaScript code, and from the corresponding URL. We automatically derive detection models that use these features using machine-learning techniques applied to labeled datasets.

To demonstrate the effectiveness and efficiency of *Prophiler*, we crawled and collected millions of pages, which we analyzed for malicious behavior. Our results show that our filter is able to reduce the load on a more costly dynamic analysis tools by more than 85%, with a negligible amount of missed malicious pages.

6.1 Introduction

The world wide web has become an integral part in the lives of hundreds of millions of people who routinely use online services to store and manage sensitive information. Unfortunately, the popularity of the web has also attracted miscreants who attempt to abuse the Internet and its users to make illegal profits.

A common scheme to make money involves the installation of malicious software on a large number of hosts. The installed malware programs typically connect to a command and control (C&C) infrastructure. In this fashion, the infected hosts form a botnet, which is a network of machines under the direct control of cyber criminals. As a recent study has shown [114], a botnet can contain hundreds of thousands of compromised hosts, and it can generate significant income for the botmaster who controls it.

Malicious web content has become one of the most effective mechanisms for cyber criminals to distribute malicious code. In particular, attackers frequently use drive-by-download exploits to compromise a large number of users. To perform a drive-by-download attack, the attacker first crafts malicious client-side scripting code (typically written in JavaScript) that targets a vulnerability in a web browser or in one of the browser's plugins. This code is injected into compromised web sites or is simply hosted on a server under the control of the criminals. When a victim visits a malicious web page, the malicious code is executed, and, if the victim's browser is vulnerable, the browser is compromised. As a result, the victim's computer is typically infected with malware.

Drive-by-download attacks have become pervasive over the last few years, and real-world examples show that legitimate (and presumably well-maintained) web sites are frequently compromised and injected with malicious code [33, 34].

Given the rising threat posed by malicious web pages, it is not surprising that researchers have started to investigate techniques to protect web users. Currently, the most widespread protection is based on URL blacklists. These blacklists (such as Google Safe Browsing [105]) store URLs that were found to be malicious. The lists are queried by a browser before visiting a web page. When the URL is found on the blacklist, the connection is terminated or a warning is displayed. Of course, to be able to build and maintain such a blacklist, automated detection mechanisms are required that can find on the Internet web pages containing malicious content.

The tools of choice for the identification of malicious web pages are (high-interaction) honeyclients. These honeyclients, such as the MITRE HoneyClient [43], Microsoft's HoneyMonkey [124], Capture-HPC [107], or Google Safe Browsing [92], run a web browser on a real operating system inside a virtual machine. The browser is pointed to a URL that should be analyzed. After the corresponding page is loaded, the honeyclient system checks for artifacts that indicate a successful attack, such as executable files on the file system or unexpected processes. While the presence of such artifacts is strong evidence that a page is malicious, the drawback of high-interaction honeyclients is the fact that the analysis is expensive. While parallelization can help in processing multiple pages more efficiently, still

the HTML page needs to be rendered and active content (such as JavaScript) needs to be executed. Moreover, after each successful exploit, the virtual machine needs to be restored, since the analysis platform can no longer be trusted. As a result, the analysis of a single URL can easily require several minutes.

In addition to high-interaction honeyclients, researchers have proposed alternative detection approaches for malicious web pages. In particular, a number of tools were proposed (such as Wepawet [17], PhoneyC [79], JSUnpack [54]) that rely on instrumented JavaScript run-time environments to detect the execution of malicious scripts, or only a certain kind of attacks (such as NOZZLE [98], a tool for the detection of heap-spraying on malicious web pages). Compared to high-interaction honeyclients, these systems provide more insights into the inner working of malicious scripts, and they require less effort to configure with a wide range of vulnerable plugins. However, they are not substantially faster, with analysis times ranging from seconds to a few minutes for a single page [17].

Unfortunately, the analysis time directly limits the scalability of these systems. As a result, it becomes very costly (if not impossible) to analyze millions of URLs in a day. This is problematic, both for organizations that try to maintain blacklists with good coverage (such as Google), but also, more generally, for everyone whose goal is to obtain a detailed and broad understanding of the malicious activity on the Internet with limited analysis resources.

One approach to address the limited scalability of current analysis systems is to devise an efficient filter that can quickly discard benign pages. By using such a filter as a front-end to a more sophisticated but resource-intensive back-end analysis system, one could save a large amount of resources, since the costly (but precise) back-end analysis is performed only on those pages that are likely to contain malicious content. Of course, one should be able to tune the sensitivity of the filter depending on the available analysis capacity and the acceptable level of false negatives (missed detections). In this context, false positives are less critical because even though they result in a waste of resources (that is, benign pages are analyzed using costly procedures), they are not increasing the exposure of users to threats.

This chapter presents the design and implementation of a filtering system, called *Prophiler*, to quickly distinguish between likely malicious and likely benign web pages. *Prophiler* statically analyzes features of the HTML page, of the embedded JavaScript code, and of the associated URL using a number of models that are derived using supervised machine-learning techniques. Pages that are found to be likely malicious by *Prophiler* can then be further analyzed with one of the more in-depth (and costly) detection tools, such as Wepawet.

Since the web page being analyzed is not rendered and no scripts are executed, the analysis is fast. Compared to previous work that attempts to detect malicious web pages based on page content, our analysis uses a significantly more comprehensive set of features, and, as a result, delivers more precise results. Researchers have also suggested identifying malicious pages based on features extracted from URLs alone. This approach delivers good results for scam and phishing pages,

since the corresponding URLs are often crafted by attackers to mislead users. However, when malicious content (such as a drive-by-download exploit) is injected into a legitimate page, the URL is not affected. Hence, systems based exclusively on URL features suffer from a substantial amount of false negatives, as shown in our experiments.

The need for a fast filter to enable the large-scale analysis of malicious web pages was previously recognized by Provos et al. [92] (some of the authors are also involved in Google's Safe Browsing efforts). Unfortunately, for obvious reasons, very few details have been revealed about Google's filter. In particular, the authors only provide examples of three page features and report that they use a proprietary machine-learning framework. Of course, the existence of Google's blacklist provides evidence that the overall system (combining the filter with the back-end analysis tools) works. Nevertheless, we feel that there are significant benefits in describing the technical details of our filtering approach in the literature: First, we introduce a comprehensive set of page and URL features for identifying malicious web pages. This allows others to build similar filters, making better use of their available analysis resources. Second, we discuss the trade-offs between false negatives and false positives, and we compare the performance of our filter to a number of previous systems. Third, we demonstrate that our filter allows us to dramatically improve the scale of the analysis that can be performed in the case of a publicly-available system, called Wepawet.

6.2 Approach

The goal of *Prophiler* is to classify pages collected by a web crawler as either likely malicious or likely benign, i.e., as likely to launch a drive-by-download attack or not, respectively. To perform this classification task, *Prophiler* uses a set of models that evaluate the features extracted from a page. These models are derived using supervised machine-learning techniques. In the following, we first describe the features extracted from a web page, and then we discuss how models are derived.

6.2.1 Features

The features extracted from a web page are the basis to determine if a page is malicious or not. Because, by design, our filter does not execute any code associated with the web page, the collected features are derived statically. By doing this, it is possible to perform the analysis faster than in the case of currently-used dynamic approaches.

We inspect two main sources of information for features: the page's contents (both its HTML and JavaScript code) and the page's URL (both its lexical and host-based characteristics). Some of the features we use have been proposed before (either for the detection of drive-by-download attacks or for the identification of

Class of features	Number of features					
	Prophiler	[109]	[63]	[28]	[67]	[108]
HTML	19	5	0	0	0	0
JavaScript	25	3	16	4	0	0
URL	12	0	0	0	4	0
Host	21	0	0	0	16	6
Total	77	8	16	4	20	6

Table 6.1: Comparison of the features, divided in four different feature classes, considered by our work and by the related approaches.

other threats, such as phishing). In this work, we introduce and evaluate 48 new features specifically designed for identifying pages involved in drive-by-download attacks.

Table 6.1 compares our approach to other relevant work in this area, in terms of the features used to evaluate web pages.

HTML features

HTML features are based both on statistical information about the raw content of a page (e.g., the page length or the percentage of whitespaces) and on structural information derived from parsing the HTML code (e.g., the location of specific elements in the page). To parse HTML pages, we use the Neko HTML Parser [15] because of its versatility and speed in parsing HTML code. Since some of our features detect anomalies in the structure of a web page (e.g., out-of-place tags), which are silently corrected by Neko, we also parse web pages with HTMLparser [85], which performs no error correction. We do not currently extract features from CSS files, even though some exploits rely on malicious style sheets. This is left as future work.

More precisely, we extract the following 19 features from HTML content: the number of `iframe` tags, the number of hidden elements, the number of elements with a small area, the number of script elements (both included via the `src` attribute, and inline), the presence of scripts with a wrong file name extension, the percentage of scripting content in a page, the percentage of unknown tags, the number of elements containing suspicious content, the number of suspicious objects, the percentage of whitespace in the page, the presence of `meta` refresh tags, the number of `embed` and `object` tags, the number of elements whose source is on an external domain, the number of out-of-place elements, the number of included URLs, the presence of double documents, the number of known malicious patterns, and the number of characters in the page.

These features capture characteristics that are commonly found in pages that have been compromised: the presence of injected content pointing to external domains, the use of obfuscation, and the presence of side effects (e.g., out-of-place tags) of the attacks used to compromise a web page. Notice that some of these

features are particularly difficult to evade for an attacker. For example, in SQL injection attacks (which are often used to inject malicious content in vulnerable web pages), attackers do not generally have complete control of the resulting page, and, as a consequence, cannot avoid the anomalies (such as malformed documents or repeated tags) that are detected by our features. Most of the features are self-explanatory. Below, we discuss some of the features that require additional discussion.

Number of elements with small area Most of the elements used to carry out a drive-by-download infection are hidden, on purpose, by the attacker. However, most drive-by-download exploits do not use visibility attributes to hide their elements, and instead set explicitly the width and height of the elements used to deliver the attack to very small values. So, we included a feature that records the number of elements of type `div`, `iframe`, or `object`, whose dimension is less then a certain threshold (30 square pixels for the area, or 2 pixels for each side).

Number of elements containing suspicious content This feature takes into account the number of elements whose content is "suspicious," i.e., the content between the start tag and the end tag of the element could be shellcode. We consider this content to be suspicious if it is longer than a certain threshold (128 characters) and contains less than 5% of whitespace characters. Note that we could use more sophisticated techniques to determine if specific content represents executable shellcode, but, in this case, we prioritize performance over precision.

Number of suspicious objects Suspicious objects are `object` elements that are included in the document and whose `classid` is contained in a list of ActiveX controls known to be exploitable. This list is taken from the PhoneyC tool [79] and has been expanded with a number of other ActiveX controls commonly found in real-world exploits.

Number of included URLs This feature counts the number of elements which, being not inline, are included specifying their source location. Elements such as `script`, `iframe`, `frame`, `embed`, `form`, `object` are considered in computing this feature, because they can be used to include external content in a web page. The `img` elements and other elements are not considered, as they cannot be used to include any executable code.

Number of out of place elements This feature counts the number of elements that reside out of their natural positioning in the HTML document. This feature is useful to detect web pages that have become malicious as the result of a stored XSS or SQL injection attack. In these cases, it is common to see scripts or `iframes` included in strange positions, such as between `title` tags or after the end of the

document (outside the `body` or `html` elements). `iframe`, `frame`, `form`, `script`, `object` and `embed` element positions are checked according to the allowed positioning, as defined by the HTML DTD specifications.

Presence of double documents This feature indicates whether a web page contains two or more `html`, `head`, `title`, or `body` elements. This is not allowed by the HTML specification, but can be seen in certain malicious web pages as a side-effect of the compromise of a web site.

Number of known malicious patterns This feature counts the number of occurrences of specific patterns commonly found in drive-by-download campaigns. The pattern list is compiled and updated by a human analyst. We currently identify only one of such patterns: the presence of a `meta` tag that causes the refresh of the page, pointing it to `index.php?spl=`, as this is very common in pages redirecting to exploit servers.

Prophiler extracts also a hash of the content of every HTML document (namely, an MD5 hash of the page), to avoid analyzing again a page that has already been analyzed, as well as a signature of the structure of the document, i.e., a signature of the tree representing its Document Object Model. This signature is used to determine if the page has a structure similar to one or more pages that have been analyzed before and determined to be malicious. If a match is found, the page is considered potentially malicious, and sent to the dynamic analysis tool.

JavaScript features

JavaScript features result from the static analysis of either a JavaScript file (such as the ones commonly served with a content type of `text/javascript` and similar), or of each script included in a web page via an inline `<script>` element. As for the HTML features, JavaScript features are both statistical and lexical.

Most malicious JavaScript scripts are obfuscated and packed, to make their analysis difficult. In some cases, malware authors adopt encryption schemes and techniques to prevent code debugging. To detect these characteristics, we implemented the extraction of some statistical measures (such as string entropy, whitespace percentage, and average line length). We also consider the structure of the JavaScript code itself, and a number of features are based on the analysis of the Abstract Syntax Tree (AST) extracted using the parser provided by Rhino [77]. For example, we analyze the AST of the code to compute the ratio between keywords and words, to identify common decryption schemes, and to calculate the occurrences of certain classes of function calls (such as *fromCharCode()*, *eval()*, and some string functions) that are commonly used for the decryption and execution of drive-by-download exploits.

We extract a total of 25 features from each piece of JavaScript code: the number of occurrences of the *eval()* function, the number of occurrences of the *setTimeout()*

and *setInterval()* functions, the ratio between keywords and words, the number of built-in functions commonly used for deobfuscation, the number of pieces of code resembling a deobfuscation routine, the entropy of the strings declared in the script, the entropy of the script as a whole, the number of long strings, the maximum entropy of all the script's strings, the probability of the script to contain shellcode, the maximum length of the script's strings, the number of long variable or function names used in the code, the number of string direct assignments, the number of string modification functions, the number of event attachments, the number of fingerprinting functions, the number of suspicious objects used in the script, the number of suspicious strings, the number of DOM modification functions, the script's whitespace percentage, the average length of the strings used in the script, the average script line length, the number of strings containing "iframe," the number of strings containing the name of tags that can be used for malicious purposes, the length of the script in characters. Hereinafter, we provide some details about a subset of these features.

Keywords-to-words ratio This feature represents the ratio between the number of keywords (i.e., reserved words) and other strings occurring in a piece of JavaScript code. This feature is useful to detect malicious pages because in most exploits the number of keywords (e.g., `var`, `for`, `while` and few others) is limited while there are usually a large number of other operations (such as instantiations, arithmetical operations, function calls). This usually does not happen in benign scripts, where the occurrence of keywords is usually higher.

Number of long strings This feature counts the number of "long" strings used in this script. A string is considered long if its length is above a certain threshold. This threshold is learned during the training phase by examining the length of strings in both known benign and known malicious pages (40 characters in our experiments).

Presence of decoding routines This feature expresses whether the JavaScript script contains snippets of code that resemble decoding routines. More precisely the AST of the JavaScript segment is analyzed to identify loops in which a "long" string is used (where "long" is defined according to the feature described before). This feature is very effective in detecting routines used to decode obfuscated scripts.

Shellcode presence probability This number expresses the probability that a JavaScript script contains shellcode. We analyze the long strings contained in the script to check if their structure resembles shellcode. We use three methods to determine if the string is likely to represent shellcode. The first method considers the number of non-printable ASCII characters in the string. The second one detects shellcode composed only of hexadecimal characters, i.e., it checks if the string is a consecutive block of characters in the ranges a-f, A-F, 0-9. The third method

checks if certain characters are repeated at regular intervals in the string, because sometimes the bytes of the shellcode are concatenated using custom separators, so that decryption routines can split the string over the specified separator(s) for further processing. The final shellcode probability for a certain script is set to the maximum of the results produced by the three individual detection methods.

Number of direct string assignments This feature counts the number of string assignments in the script. To extract this feature, we analyze the structure of the AST generated by the parser. We consider a number of ways in which a JavaScript program can instantiate a string. More precisely, we count string assignments done through direct assignment, setting of properties, direct string declaration, instantiations inside the conditional operator '?', and arrays. The rationale behind this feature is that malicious scripts tend to have an unusually large number of string assignments, as a side effect of deobfuscation and decryption procedures.

Number of DOM-modifying functions This feature counts the number of functions used to modify the Document Object Model that are referenced in the source code. Drive-by-download exploits usually call several of these functions in order to instantiate vulnerable components and/or create elements in the page for the purpose of loading external scripts and exploit pages. We consider the most-commonly-used DOM functions implemented in all the major browsers, plus a small set of functions which are only available in Microsoft Internet Explorer's JavaScript engine, such as *clearAttributes()*, *insertAdjacentElement()*, and *replaceNode()*. Note that we perform some limited static (data flow) analysis to identify cases where basic DOM elements (e.g., the `document` variable) are assigned to other variables that are later modified.

Number of event attachments This is the number of calls to functions used to set event handlers on certain actions. Not all events are interesting for us, as drive-by-download attacks usually need only to be triggered as the page loads or to disable error reporting in case something goes wrong. So, we only count event attachments related to these events: *onerror*, *onload*, *onbeforeunload*, *onunload*. The functions that can be used to attach an event handler are *addEventListener()*, *attachEvent()*, *dispatchEvent()*, and *fireEvent()*.

Number of suspicious object names This feature represents the number of objects with a suspicious name. These objects are identified using the list of exploitable objects already used by HTML features (see Section 6.2.1). However, since most of the exploits dynamically insert objects and ActiveX controls into web pages using JavaScript, we have to check for these components also in the JavaScript code.

Number of suspicious strings This feature has been added after manually analyzing several dozens of malicious scripts and noticing that most of them, if not obfuscated, tend to use certain strings as variable or function names. Thus, we check whether a script contains such tell-tale signs (common strings are, for example, "evil," "shell," "spray," and "crypt"), and we count how many occurrences of these strings are found.

Number of "iframe" strings This feature counts how many strings containing "iframe" are present in a script. This feature is motivated by the fact that malicious scripts often inject several iframes into a web page, and, if the script is not obfuscated, it is possible to identify when the script modifies the DOM to inject an `iframe` element.

Number of suspicious tag strings Similarly to the previous feature, this feature counts the number of times that certain tag names appear inside strings declared in JavaScript code. In fact, instead of injecting iframes, sometimes malicious scripts write other scripts or objects inside the page. This feature counts the appearance of `script`, `object`, `embed`, and `frame` inside JavaScript strings.

Each piece of JavaScript code is also characterized by a hash of the content (to avoid analyzing a previously-seen script) and a signature of the AST of the document. This is used to identify similar scripts that have been already analyzed and that have been found to be malicious. If a match is found, the web page is considered malicious, and it is sent to the dynamic analysis tool for further processing. To prevent simple obfuscation techniques from hiding the similarity with other scripts, this AST signature does not take into account variable names and the structure of arrays, and it is invariant to the place where functions are declared.

URL and host-based features

As shown by previous work [67], it is often possible to predict if a certain web page is malicious by looking only at its URL. Even though detecting drive-by-download pages using URL features is more complex than in the case of phishing pages or scam pages, some information contained in the URL and associated with the referenced host can be used to help in the detection of malicious web pages. For example, malware campaigns are often hosted on untrusted hosting providers, and the corresponding `whois` information reveals short registration time frames or sanitized (anonymized) registration information. Also, it is very common for malicious web pages to include content from sites with no DNS name, or hosted on domains with a certain TLD (e.g., `.cn`, `.ru`).

The collected features are syntactical (the domain name length, whether the original URL is relative, the presence of a suspicious domain name, the TLD of this URL, the presence of suspicious patterns, the length of the file name appearing in the URL, the presence of a suspicious file name, the absence of sub-domain, the

presence of an IP address in the URL, the presence of port number, the absolute and relative length of this URL), DNS-based (resolved PTR record, whether the PTR record is equal to the A record for this IP, and, for each of the A, NS, MX records for this host: first IP address returned, number of corresponding IP addresses, TTL of the first IP address, Autonomous System number of the first IP), whois-based (registration date, update date, expiration date), and geoip-based (country code, region, time zone, netspeed). We use a total of 33 features derived from the analysis of URL and host information. Below, we discuss some details about a subset of the features.

Number of suspicious URL patterns Analyzing the URLs of several pages launching drive-by-download exploits, we observed that many of them shared common names or recurring patterns in their paths (we speculate that this is an indication that different attacks are performed using the same exploit toolkits, e.g., MPack, Eleonore, and CrimePack). Some examples of these patterns are file names such as `swfNode.php` or `pdfNode.php`. Thus, we use this feature to count how many patterns from a list of known bad patterns appear in the URL. We derive known bad patterns from known exploit kits. We currently identify 10 different suspicious URL patterns.

Presence of a subdomain in URL We noted that, frequently, malicious web pages refer to the domains serving malware without specifying a subdomain (e.g., `example.com` instead of `www.example.com`). This feature keeps track of whether a subdomain is present in the URL.

Presence of IP address in URL Some web sites hosting malware are not associated with domain names but are addressed by their IPs instead. A common reason for this is that the malware is hosted on a victim machine on a public network that was compromised. This feature records if an IP address is present as the host part in the URL.

Value of the TTL for the DNS A record This feature examines the Time To Live (TTL) of the DNS entry of the first IP address returned by the DNS A query for a host name. Shorter TTLs are usually associated with services that are likely to be moved to another IP address in the near future. This can be the case for DNS entries associated with malicious (fast-flux) hosts.

Value of the TTL for the DNS NS record This feature examines the Time To Live of the first NS entry for the host name under analysis. This feature is useful for identifying malicious web pages because criminals often use different DNS records to redirect requests to a different IP address once one of their command-and-control servers is shut down.

Relationship between PTR and A records This feature indicates whether the resolved PTR record equals the IP address for the host under examination. For benign web servers, the values should be consistent.

Registration date This feature examines the registration date for the host name (domain), if it is available via the Whois service. Registration dates are commonly used to distinguish between benign and malicious domains, since most of the command-and-control and exploit servers reside on domains whose registration date is recent and/or whose expiration date is in the near future. This is because attackers often buy domain names for short time frames, since they expect that those names will be blocked quickly.

Country Code This feature leverages the country code to which the IP address of the host belongs. This feature is extracted via a geoip query *.

Unlike previous work [67], we do not consider the domain registrar as one of our features. Even though we extract and store this information, the models we derived during the training process did not identify the registrar as a relevant feature for determining if a web page is malicious or not.

6.2.2 Discussion

Models and classification In *Prophiler*, a model is a set of procedures that evaluate a certain group of features. More precisely, the task of a model is to classify a set of feature values as either likely malicious or likely benign. A model can operate in training or detection mode. In training mode, a model learns the characteristics of features as found in sets of web pages that are known to be either malicious or benign. In detection mode, the established models are used to classify pages as either likely malicious or likely benign.

Using a *training* dataset, we derived a number of models to detect likely malicious web pages, based on the features described earlier. The model learning process is further explained in Section 6.4. After training, we evaluated the effectiveness of our models on a *validation* dataset. Once we were confident that the models were able to effectively classify malicious web pages, we deployed them as a filter for our dynamic analysis tool (Wepawet). This resulted in a tenfold increase in the amount of web pages that the system can process with a given set of resources.

Machine learning As with all classification problems, our learning-based approach to the detection of malicious web pages faces several challenges [111].

∗. Geoip queries are used to retrieve location information about an IP address. Usually, this information includes the country, region, and city to which the IP address belongs, as well as some other information such as the Internet Service Provider of this address, depending on the geoip service in use.

Here, we discuss in particular the assumptions at the basis of our analysis and the techniques we used to ensure that these assumptions hold in our setting. First, we assume that the distribution of feature values for malicious examples is different from benign examples. To ensure this, we carefully selected the features that we use in our system on the basis of the manual analysis of a large number of attack pages. We note that individual feature values may appear in both malicious and benign pages (e.g., some benign pages are obfuscated, thus they would "trigger" features that capture obfuscation). However, it is unlikely that the combination of all the features we consider is similar in benign and malicious pages. A second assumption is that the datasets used for model training share the same feature distribution as the real-world data that is evaluated using the models. We address this issue by training our models with large datasets of recent malicious and benign pages, and by continuously evaluating the effectiveness of our filter in detecting web pages with respect to the results provided by (more costly) dynamic analysis tools. A final requirement is that the ground truth labels for the training datasets are correct. To this end, we use state-of-the-art tools and manual inspection to ensure that our labeled datasets are correct.

Evasion The attentive reader will notice that some of our features could be evaded by malicious scripts. For example, the detection of tags with a small area (one of our HTML features) could be thwarted by dynamically generating these elements (e.g., via an obfuscated call to *eval()*). However, our set of features is comprehensive and covers characteristics that are common to malicious scripts (e.g., the use of obfuscated code). As a consequence, as our experiments show, our system is not easily evaded by current malicious web pages. Moreover, it is easy to extend *Prophiler* with additional features to cover future attacks. We always send to the back-end analysis (honeyclient) a small fraction of random pages that our system has classified as benign. This allows us to detect systemic false negatives, and to update our feature sets and models accordingly.

Even with full knowledge of our feature set, it is not trivial for an attacker to disguise his malicious code. First, in certain cases, the freedom of an attacker is limited with regard to the parts of the infected web page that he can modify. In particular, this is true when attackers use SQL injection vulnerabilities, which often result in out of place HTML elements that cannot be cleaned up (and which are picked up by our system). Second, many of our features do not target specifics of particular exploits, but general properties of entire classes of attacks. For example, artifacts that are the result of obfuscated JavaScript are hard to disguise. Of course, an attacker could opt to send the exploit code in the clear, but in doing so, he risks that signature-based solutions detect and block the malicious code.

Attackers could also try to fingerprint, detect, and consequently evade, our tool when it visits a malicious website. This is a problem every malware detection tool faces, and we address it in two ways. First, we configure our system so that it closely mimics a real browser (for example, by setting the user-agent

of the crawler component as described in Section 6.3). Second, we try to detect fingerprinting attempts by using features that check for the presence of JavaScript routines commonly used for this task (as discussed in Section 6.2.1).

Trade-offs Even though we put great care in the selection of the features and the derivation of models, we do not expect our filter to be as accurate as honeyclients, which can rely on the dynamic characteristics of a web page for the detection of malicious behavior. Instead, we expect the filter to provide useful information that can be used to quickly discard benign web pages, and to send *likely* malicious pages to dynamic analysis tools, which can perform more detailed analysis.

In this context, it is critical to minimize false negatives, i.e., missed detections. In fact, if a page that is indeed malicious is incorrectly classified as benign by our filter, it will be immediately discarded without being further analyzed. Therefore, the malicious page will evade the detection of the combined filter/honeyclient system. Conversely, false positives are not as problematic: If a benign page is incorrectly flagged as likely malicious by our filter, it will simply be forwarded for analysis to the honeyclient, which (we assume) will ultimately mark it as benign. The net effect is that resources are wasted (because the back-end honeyclient has to analyze a benign page). However, in this case, no incorrect detection will be made by the overall detection system.

6.3 Implementation and setup

We implemented *Prophiler*, with the aim of using it as a filter for an existing dynamic analysis tool, called Wepawet [17] (which is publicly available at `http://wepawet.cs.ucsb.edu/`). However, *Prophiler* can be used unchanged as a filter for any of the other, publicly available honeyclient systems. The overall architecture of the system is shown in Figure 6.1.

Prophiler is fed by a modified instance of Heritrix [39], which crawls † a list of seed URLs fetched daily from three search engines (namely, Google, Yahoo, and Bing). The crawls are seeded by using the current Twitter, Google, and Wikipedia trends as search terms. These trends are used as a basis for the searches because most malware campaigns use Search Engine Optimization (SEO) techniques to increase their ranking in the search engines' results associated with popular terms [40,41]. Another source of seeds for our crawler is a list of links extracted from a feed of spam emails. The list of links is updated daily and provides us with an average of 2,000 URLs per day.

We modified the crawler to be able to set the "Referer" header when fetching a seed URL. This header has to be set to the search engine from which the seed URL was extracted. This is necessary because some malicious web pages deliver

†. Most drive-by-download attacks use browser fingerprinting to decide whether to 'render themselves' as malicious or benign. We decided to set up our crawler's user-agent as *MS Internet Explorer 6* on Windows XP, to trigger malicious behavior in most cases.

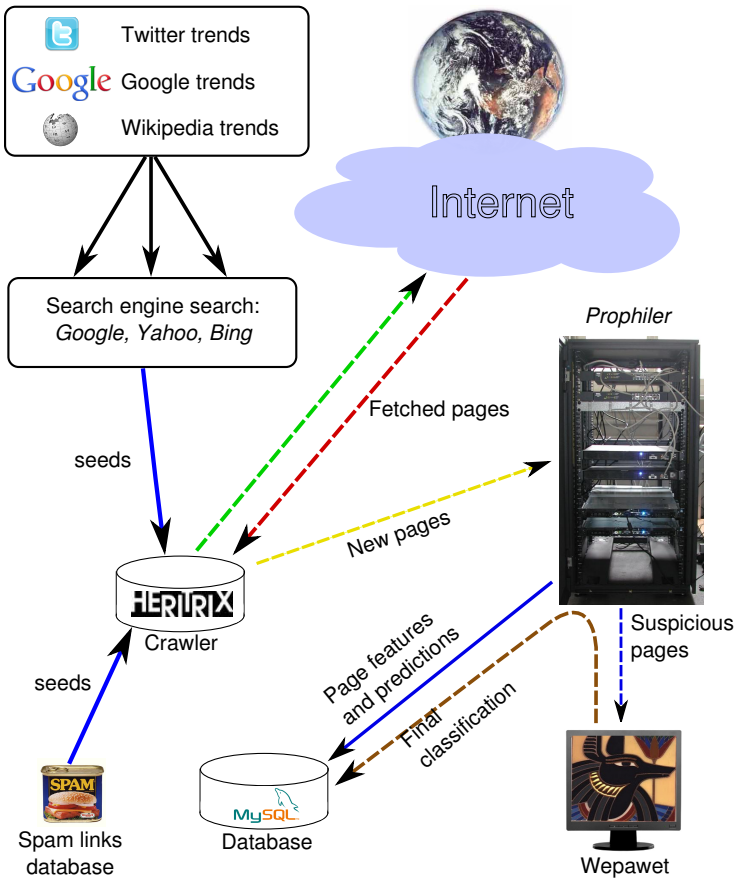

Figure 6.1: Architecture of the system.

malicious content only when the request appears to be the result of a user clicking on the search results.

The crawler fetches pages and submits them as input to *Prophiler*, which analyzes each page and extracts and stores all the features. Once all features have been extracted from a page, *Prophiler* uses the models learned in the previous training phase to evaluate its maliciousness. If a page has been identified as likely malicious, it is forwarded to the dynamic analysis tool (Wepawet, in our case). This tool then confirms that the page is indeed malicious or it flags it as a false positive.

The system was installed on a server running Ubuntu Linux x64 v 9.10, with an 8-core Intel Xeon processor and 8 GB of RAM. The crawler and the analysis system are both running on this machine. The system in this configuration is able to analyze on average 320,000 pages/day. Taking into account that a single page can contain multiple links to JavaScript programs, frames, and objects (which are all automatically included by the browser when rendering the page), the analysis must examine around 2 million URLs (objects) each day.

Dataset name	Benign pages	Malicious pages	Total pages
Training	51,171	787	51,958
Validation	139,321	13,794	153,115
Evaluation	N/A	N/A	18,939,908
Comparison	9,139	5,861	15,000

Table 6.2: Datasets used for our experiments.

6.4 Evaluation

In this section, we evaluate the effectiveness and performance of *Prophiler*. More precisely, we first discuss how the models used to detect malicious pages were automatically derived from a training dataset. Then, we evaluate *Prophiler* on a number of datasets, both labeled and unlabeled, adding up to almost 20 million web pages. Finally, we quantitatively compare our approach with those that were proposed in the past.

Model derivation To derive our detection models, we collected a labeled dataset composed of both malicious and benign pages. We refer to this dataset as the *training dataset*. As shown in Table 6.2, the training dataset comprises 787 pages that are known to be triggering drive-by-download attacks. These pages were extracted from Wepawet's database. Furthermore, we confirmed by manual inspection that these pages indeed contain malicious code used to launch drive-by-download attacks. We also collected a set of 51,171 benign web pages by crawling the first two levels of the top 100 Alexa web sites [3]. In this case, our assumption was that these extremely popular web sites are unlikely to have been compromised and used for malware distribution, as they are visited daily by millions of users, as well as continuously analyzed by experts and anti-virus programs. Furthermore, we used the Google Safe Browsing API to remove any malicious pages from this set.

We extracted our detection models from this dataset using the WEKA machine-learning platform [38]. We experimented with a number of standard models, such as naïve Bayes, random forest, decision tree, and logistic regression classifiers. In order to choose a suitable classifier (i.e., the one providing the lowest possible number of false negatives, and a reasonably small amount of false positives) we used our training dataset to build several models, each with a different classifier and/or different parameters. The models were extracted from and tested on the training dataset, using 10-fold cross-validation.

Note that we built three different models that operate on the three different feature sets that we defined previously (HTML features, JavaScript features, and features related to the URL and host name). This allows us to evaluate the effectiveness of individual feature sets and to experiment with different machine learning

Feature class	Classifier	% FN	% FP
HTML	Random Tree	30.4	0.8
	Random Forest	20.5	2.4
	Naive Bayes	16.4	44.1
	Logistic	25.6	17.1
	J48	36.6	0.8
	Bayes Net	15.1	23.2
JavaScript	Random Tree	22.4	0.2
	Random Forest	18.1	0.5
	Naive Bayes	51.5	1.0
	Logistic	81.0	0.0
	J48	21.4	0.3
	Bayes Net	39.9	1.7
URL + HOST	Logistic	9.3	1.0
	J48	9.6	0.6

Table 6.3: False Negatives (FN) and False Positives (FP) ratios for the tested classifiers. The class of features related to the URL and host information has been tested against fewer classifiers because most of them do not support date attributes.

models. For the final classification of a page, the output of the three models needs to be combined, as discussed below.

The results for the three classifiers (using 10-fold cross validation on the *training set*) are presented in Table 6.3. It can be seen that the classifiers that produced the best results were the Random Forest classifier for the HTML features, the J48 classifier for the JavaScript features, and the J48 classifier for the URL- and host-based features. In the rest of the experiments, we configured *Prophiler* to use these classifiers.

Interestingly, as shown in Table 6.3, it can be seen that a single class of features is not sufficient to reliably determine the maliciousness of web pages. In fact, individual models yield both high false positive and high false negative rates. For example, when analyzing JavaScript features alone, even J48 (one of the best performing models for this class) produces 21.4% false negatives (with 0.3% false positives). However, as we will show with a number of tests on various datasets, combining models for all the feature classes substantially improves the detection capability of our tool. In *Prophiler*, we declare a page as malicious when one or more of the individual classifiers declare a page as malicious. The rationale for this decision is that a page's maliciousness may be determined by causes (e.g., an iframe tag or an HTML-based redirect) that are modeled by only one class of features. Therefore, whenever the model associated with a class of features classifies a page as likely malicious, *Prophiler* raises an alert. As a result, by combining models, we can substantially reduce the false negatives of the filter by accepting

a minor increase in false positives (which are much less problematic, as discussed previously).

Effectiveness of new features In the next step, we inspected the models generated by WEKA to determine the importance of the new features that we added compared to previous work. We found that some of these features were particularly effective in the detection of web pages launching drive-by downloads. Regarding the JavaScript features, some of the most important new features are shellcode presence probability (which is at the first level in the decision tree of the chosen J48 classifier), the presence of decoding routines, the maximum string length, and the entropy of the scripts and of the strings declared in it. Several new features related to HTML content appear to be very effective in the detection of malicious or infected web pages. Such features are the number of included URLs, the number of elements containing suspicious content, the number of iframes, the number of elements with a small area, the whitespace percentage of the web page, the page length in characters, the presence of `meta` refresh tags and the percentage of scripts in the page. As for URL and host related information, the most effective novel features introduced by our work are the TLD of the URL, the absence of a subdomain in the URL, the TTL of the host's DNS A record, the presence of a suspicious domain name or file name, and the presence of a port number in the URL.

Validation After the model derivation, we validated *Prophiler* by running it on a second labeled dataset, which we refer to as the *validation dataset*. This dataset contained 153,115 pages that were submitted to the Wepawet service by its users over a period of 15 days. We labeled each page with the result produced by Wepawet: in total, there were 139,321 benign pages and 13,794 malicious ones. On this dataset, *Prophiler* produced a false positive rate of 10.4% and a false negative rate of 0.54%. In other words, if used as a filter on this dataset, *Prophiler* would immediately discard 124,906 benign pages, thus saving valuable resources of the more costly (dynamic) analyzer.

Table 6.4 shows which models triggered the detection of malicious web pages when running the system on the *validation* dataset. One can see that most of the pages are considered malicious because of their HTML features, and secondly because of the JavaScript ones.

Large-scale evaluation We performed a large-scale evaluation of *Prophiler* by running it over a 60-day period on a dataset containing 18,939,908 pages. This dataset (which we refer to as the *evaluation dataset*) was built by leveraging the infrastructure described in Section 6.3. All the pages in the evaluation dataset are unlabeled.

Prophiler flagged 14.3% of these pages as malicious, thus achieving an 85.7% reduction of the load on the back-end analyzer (in our setup, the Wepawet service).

Number of pages	Reason of suspiciousness
124,906	**None (classified as benign)**
14,520	HTML
9,593	JavaScript
1,268	Request URL
814	JavaScript + HTML
806	Request URL + HTML
467	Included URL(s)
189	Request URL + JavaScript
181	Included URL(s) + HTML
130	Request URL + JavaScript + HTML
119	Request URL + Included URL(s)
46	Request URL + Included URL(s) + JavaScript + HTML
28	Request URL + Included URL(s) + HTML
17	Request URL + Included URL(s) + JavaScript
16	Included URL(s) + JavaScript
15	Included URL(s) + JavaScript + HTML

Table 6.4: Results on the *validation* dataset.

With the current implementation of Wepawet, this corresponds to a saving of over 400 days of processing. Figure 6.2 shows in more detail the analysis statistics for the 60-day analysis period. (The variation in the number of pages processed per day depends mainly on the number of URLs used as seeds, and the type and complexity of the visited pages, for example, the number of external resources fetched by each page.)

After Wepawet had analyzed all the pages that were marked as malicious by *Prophiler*, we could determine the false positive of our filter. We found that the false positive rate for this dataset was 13.7%. Recall that a false positive in our filter simply determines undesired load on the back-end analyzer (which is forced to inspect benign pages), but does not result in actual alert. Quantifying false negatives in these settings is more challenging, since the dataset is unlabeled and complete manual analysis is infeasible given the sheer size of the dataset. To estimate the false negative rate on the evaluation dataset, we processed with Wepawet 1% of the pages that *Prophiler* classified as benign (the pages to be further inspected were chosen at random). Of these 162,315 pages, only 3 were found to be malicious.

Comparison with previous work We compared *Prophiler* against a number of previously-proposed systems that rely on lightweight analysis techniques to detect malicious web pages, and that, thus, could be used as fast (pre)filters.

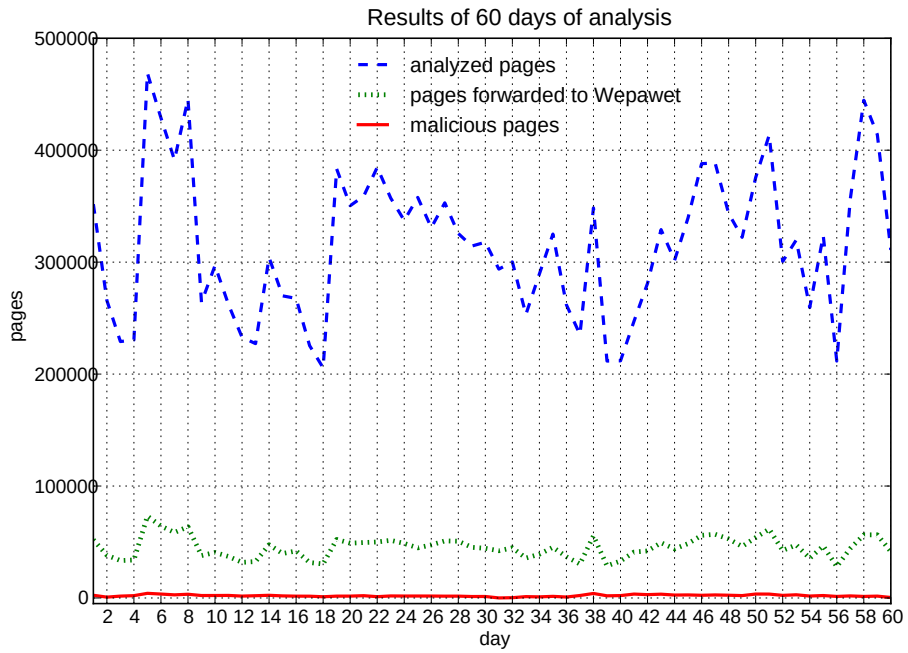

Figure 6.2: Analysis of the *evaluation* dataset. On average, 1,968 pages every day were confirmed as malicious by Wepawet.

More precisely, we considered the approach presented in [109], which relies on five HTML features and three JavaScript features to detect drive-by-download attacks; and the approach of [67], which analyzes URL features to detect malicious URLs. In addition, to better understand the effectiveness of the novel features that we introduced with respect to those that were proposed in the past, we created a classifier that combines all the features previously proposed in [28,63,67,109]. We did not compare our system to [108] since obtaining four out of the six features they use (the ones related to redirects) was not possible for us; the other two features they propose are already included.

Unfortunately, we were also not able to compare our filter to the one used by Google [92]. The reason is that their filtering framework is proprietary (and not available to us) and is not publicly described in detail. Moreover, when using Google's Safe Browsing API, one is only able to check whether a page has been deemed malicious by Google's entire analysis framework, which is based on the use of honeyclients. However, no information can be retrieved about the false positive and false negative ratios of their initial filtering system.

To compare *Prophiler* with the above-mentioned systems, we built a labeled dataset (the *comparison dataset*) of 15,000 web pages and associated URLs. This dataset contains 5,861 pages involved in drive-by-download attacks; the remaining pages are benign. We modified our filter so that it would use only the features

described in [109] (to reproduce the system described therein) and those presented in [28, 63, 67, 109] (to test the detection that can be achieved with all previously-known features combined). Finally, we asked the authors of [67] to analyze the URLs of the comparison dataset using their system.

Work	**Features collection time**	**Classification time**	**FP %**	**FN %**	**Considered feature classes**
[109]	0.15 s/page	0.034 s/page	13.70	14.69	HTML, JavaScript
[67]	3.56 s/URL	0.020 s/URL	14.83	8.79	URL,Host
Union of [28, 63, 67, 109]	N/A	N/A	17.09	2.84	HTML, JavaScript, URL, Host
Prophiler	3.06 s/page	0.237 s/page	9.88	0.77	HTML, JavaScript, URL, Host
Prophiler's top 3*	N/A	N/A	25.74	5.43	HTML, JavaScript, URL, Host
Prophiler's top 5*	N/A	N/A	5.46	4.13	HTML, JavaScript, URL, Host

Table 6.5: Comparison between *Prophiler* and previous work.
*These are, respectively, models built using only the top 3 and top 5 features appearing in the decision trees of *Prophiler*'s original machine learning models.

The results (in terms of average URL processing and testing time, false positives, and false negatives) are shown in Table 6.5. For the approach described in [67], we report the best results, which were achieved using an SVM classifier with an RBF kernel. *Prophiler* clearly outperforms existing approaches in terms of detection rates. In particular, it is interesting to observe that *Prophiler* has lower false positives and false negatives than the system that combines the features of [28, 63, 67, 109], indicating that the novel features and the models we use are effective and improve detection compared to the state of the art. Finally, the experiment also shows that *Prophiler*'s feature collection time is very low, despite the fact that it extracts a larger number of features than the other approaches. By profiling our tool, we found that the most of the feature collection time is due to the extraction of host-based features from the URLs (such as DNS information, Whois data, and geoip information). Note that the value of *Prophiler*'s features collection time refers to a "from scratch" run of the system, i.e., with an empty database. However, we found that a few hours after deployment, the database contains in-

formation about the majority of the hosts analyzed. Therefore, in steady state, our system is considerably faster, reaching a processing time of about 0.27 s/page.

6.5 Conclusions

As malware on the Internet spreads and becomes more sophisticated, anti-malware techniques need to be improved in order to be able to identify new threats in an efficient, and, most important, automatic way. This chapter described the development and evaluation of *Prophiler*, a system whose aim is to provide a filter that can reduce the number of web pages that need to be analyzed dynamically to identify malicious web pages. We have deployed our system as a front-end for Wepawet, a well-known, publicly-available dynamic analysis tool for web malware. The results show that *Prophiler* is able to dramatically reduce the load of the Wepawet system with a very small false negative rate. The tool is currently available upon request, as an open source project provided under the GNU Affero General Public License (AGPL). We believe the approach described in this chapter can be employed by any security company or research institution in order to effectively classify and filter hundreds of thousands of web pages per day on a single machine. Moreover, by coupling such a system with a dynamic analysis tool, *Prophiler* would allow to proactively build blacklists of web pages hosting drive-by download exploits in a very efficient way, allowing for a timely protection of web users.

Chapter 7

Conclusions and Future Work

In recent years, we have been witnessing an always increasing number of security incidents on the web. As we presented in the beginning of this dissertation, four are the main actors involved in the vast majority of web attacks that spread malicious code or compromise web sites on the Internet. These actors are: the attackers, the vulnerable websites hosted on the premises of hosting providers, the victims of attacks (web users), and the security companies and researchers who fight cyber criminals by detecting and analyzing malicious code on the web.

Past literature has often analyzed each of these four actors by adopting an *external* point of view, e.g., by analyzing a posteriori the traces of a compromise, but without visibility on what the attacker actually did during the attack. It is thus necessary to adopt a novel approach in order to analyze web attacks from the point of view of each of these four actors. This dissertation is, to our knowledge, the first study of web attacks and website compromises that analyzed these phenomena by adopting the four distinct points of view of its different actors. Moreover, all of the four considered points of view have been analyzed by means of large-scale measurement studies, which renders our analyses significant on a worldwide scale.

In Chapter 3, we have presented a worldwide-scale study on the way shared web hosting companies handle the security of their customers. Results show that whereas a good fraction of the hosting companies put in place simple attack prevention mechanisms, almost none of them is able to detect obvious signs of compromise on their customers' accounts. Also, we have showed that abuse complaints are not always handled in a timely and appropriate manner, leaving sometimes space for a malicious user to trick providers into taking down legitimate web sites.

Chapter 4 then reported on a large-scale study of the behaviors of web attackers, performed by setting up a network of 500 fully functional vulnerable web sites and observing the actions of attackers *while* and *after* they exploit vulnerabilities on the web applications. This allowed us to gain an unique point of view on the reasons behind attacks that hit the majority of medium and small size websites on the Internet. In particular, while the majority of attacks seem to employ a combination of automated scanning and exploitation bots for compromising known vulner-

abilities on web sites, we still witnessed a non negligible fraction of attacks from "novice" hackers, who carry out attacks manually and seem to perform attacks only for fun or bragging.

We have also presented, as reported in Chapter 5, a first study towards web user-based risk profiling, where we have analyzed a large dataset comprising the 3-month web browsing history of the customers of a known security firm. This allowed us to analyze and understand relations between the behavior of a user, and his or her probability of visiting malicious websites. For example, we witnessed an increase in the percentage of malicious websites visited during night time and weekends, and an increase in the chance of visiting malicious websites for users who browse the Internet a lot or who browse a wide variety of web sites. We also showed that user profiling could be a viable way to predict the class of risk for a user on the web, similarly to what is currently done in the field of insurances and financial risk prediction.

Finally, as presented in Chapter 6, we introduced a novel approach for security companies and researchers to detect web pages launching drive-by-download attacks. Our approach, implemented in a system called *Prophiler*, uses static analysis and machine learning techniques for the large-scale detection of malicious web pages. We show that this system can be successfully applied in real-world scenarios, and is able to efficiently filter out benign pages from large amounts of web pages, allowing for a more costly analysis to be performed only on the suspicious ones.

While some of the points of view we explored in this thesis have been widely studied in academia (e.g., the detection of malicious web pages by researchers), others have not. For this reason, it can be envisaged to work on future directions in these areas of research. For instance, in order to provide even more complete insights on human risk factors linked to visiting malicious web pages, it would be interesting to have access to users' "social" features, such as their sex, age, profession and personal interests, in a similar study to the one we presented in Chapter 5. This would improve the completeness, and probably the risk detection accuracy, of the user profiles built with such an approach. Unfortunately, there are some ethical concerns to be addressed before being able to grant researchers the right to access all this information, and to publish the results of their work. This is why, in our study, we have limited ourselves to analyze anonymized information about clients' browsing habits.

Other studies could be conducted to advance our research on web attackers, for example by deploying honeypot websites on big and renowned websites, such as online news platforms, online retailers, banks and governmental websites. This would allow researchers to gain some insights also on the behavior of attackers aiming at high profile targets – which, we believe, should be quite different from the average attacker we depicted in Chapter 4.

A similar extension could be envisaged also for a more complete analysis on the behavior of hosting providers in relation with website compromises. Apart from shared hosting providers and cheap website protection services, in fact, no

research has still been conducted on the way high-end hosting providers (e.g., those providing cloud servers, bare-metal or virtual dedicated servers) and costly website security services handle compromises on their customer's websites. This could be an interesting extension of the study we presented on the point of view of hosting providers on malicious and compromised websites.

In conclusion, as this dissertation has shown, ensuring users can browse the web safely is today a very challenging task. Cyber criminals are numerous and well organized, and often have access to good technical and financial resources. Luckily, on one side, many security companies and researchers are constantly scanning the Internet in search of malicious code, with the purpose of stopping criminals and taking down malicious websites. The same security companies also typically provide users and businesses with endpoint protection mechanisms that can be used to protect the regular user browsing the Internet, up to a certain extent. On the other side, though, we have seen that malicious and compromised websites are the method of choice for criminals to spread malicious code on the Internet. As Chapter 3 showed, the majority of shared web hosting providers, which host the majority of websites on the web, do very little or nothing to detect even the simplest signs of compromise, and often fail even in handling abuse complaints. Both good compromise detection capabilities, and prompt responses to abuse notifications are necessary means for allowing the timely detection – and takedown – of malicious websites. For these reasons, we believe that web hosting providers should play a key role in making the Internet a more secure place. Even a small improvement in the way in which they handle website compromises and abuse reports, would mean far higher chances of stopping web attacks at their earliest stages, compared to what today is done. As a result, setting up malicious or compromised websites, and keeping them online for long periods of time, would both become much harder tasks for the vast majority of web attackers.

Chapter 8

Résumé

L'incroyable développement subi par le World Wide Web a permis la création de nouveaux métiers, marchés, activités récréatives, ainsi que de nouveaux moyens de partage de connaissance et d'argent. Toutefois, le web attire malheureusement aussi de plus en plus de malfaiteurs, qui le considèrent comme un moyen facile pour gagner de l'argent en exploitant les services et la propriété d'autrui.

Cette thèse propose une étude des sites web compromis et malicieux sous plusieurs axes d'analyse. Même si les attaques web peuvent être de nature très compliquées, on peut quasiment toujours identifier quatre acteurs principaux dans chaque cas. Ceux sont les attaquants, les sites vulnérables hébergés par des fournisseurs d'hébergement, les utilisateurs qui sont victimes des attaques, et les sociétés de sécurité qui parcourent Internet à la recherche des criminels et de sites web compromis à être bloqués. Dans cette thèse, nous analysons premièrement les attaques web du point de vue des hébergeurs, en montrant que, même si des outils simples et gratuits pourraient permettre de détecter des signes simples de compromission, la majorité des hébergeurs échouent dans cette épreuve. Nous passons en suite à l'analyse des attaquants et des motivations de leurs attaques, en étudiant les attaques web collectés par une plate-forme constitué de plusieurs centaines de sites web vulnérables. Ensuite, nous étudions le comportement de milliers de victimes d'attaques web, en analysant leurs habitudes pendant la navigation sur le web, afin d'estimer s'il est possible de créer des "profils de risque" pour les utilisateurs web, de façon similaire à ce que les compagnies d'assurance font aujourd'hui. Enfin, nous adoptons le point de vue des sociétés de sécurité, en proposant une solution efficace pour la détection d'attaques web convoyées par sites web compromis, capables d'infecter des milliers d'ordinateurs chaque jour.

8.1 Introduction

Au cours des dernières années, le World Wide Web (WWW) a changé radicalement la société et la façon dont les gens vivent et se comportent quotidiennement. Il a évolué à partir d'un ensemble de documents d'hypertexte statiques,

servant une communauté restreinte de scientifiques, tel qu'il était à sa naissance en 1991, à un réseau de dispositifs informatiques inter-connectés à niveau mondial, qui fournissent des services sophistiqués, génèrent dynamiquement du contenu, et permettent aux gens de communiquer, de partager des connaissances, des données et de l'argent à partir de n'importe où dans le monde. Un grand nombre de services en ligne sont disponibles, permettant aux gens de faire un usage plus efficace et plus rentable de leur temps et ressources.

Courriels, comptes bancaires en ligne, achats en ligne, déclarations d'impôt en ligne, et même le vote en ligne ne sont que quelques exemples de services qui, au cours de de la dernière décennie, ont révolutionné la façon dont les gens vivent, font des affaires, interagissent les uns avec les autres et avec les institutions.

Même si ces changements contribuent à améliorer la qualité de vie du nombre croissant de personnes qui utilisent le Web de jour en jour, Internet pose également un certain nombre de menaces pour ses utilisateurs. Précisément, la popularité d'Internet a également attiré mécréants qui, en permanence, tentent d'abuser de ses services et ses utilisateurs afin d'obtenir des profits illégaux. Toute personne, organisation, gouvernement est donc une cible potentielle de différents types de attaques qui peuvent provenir du web.

8.1.1 Code Malveillant sur le Web

Avec la popularité croissante des services en ligne, les criminels et les utilisateurs malveillants regardent le WWW avec beaucoup d'intérêt. En fait, un nombre croissant d'utilisateurs d'Internet signifie pour eux un nombre croissant de victimes potentielles, et donc de profits.

Les services et sites Web en ligne de grande taille stockent aujourd'hui les données personnelles de plusieurs millions d'utilisateurs. L'accès à ces données peut s'avérer très rentable pour un criminel [95] et provoquer des graves pertes financières ou même la faillite de l'organisation ciblée par l'attaque [106].

Parallèlement à des activités criminelles, aussi les gouvernements commencent à exploiter les faiblesses des utilisateurs et des services Internet comme un moyen d'obtenir tout type de renseignements, à des fins de défense nationale [32,59], mais aussi pour attaquer ou perturber les services et installations des pays ennemis [36]. Ce phénomène est souvent appelé *Cyber Warfare*.

Dans ce contexte, nous utilisons *code malveillant* comme un terme général pour décrire tout système, script, morceau de logiciel ou de code qui peut potentiellement causer des effets indésirables sur un dispositif informatique, en attaquant un de ses "attributs de sécurité": confidentialité, intégrité et disponibilité (souvent abrégé "CIA" dans la communauté de la sécurité informatique). Exemples de ces effets indésirables peuvent être: endommager un système, voler des informations, perturber des services, ou prendre le contrôle total ou partiel d'un dispositif.

Le web est aujourd'hui le lieu de choix pour la diffusion de *code malveillant*, parce que la nature interdépendante de l' Internet permet d'accéder instantanément à grandes surfaces d'attaque vulnérables, et effectuer attaques à grande échelle qui

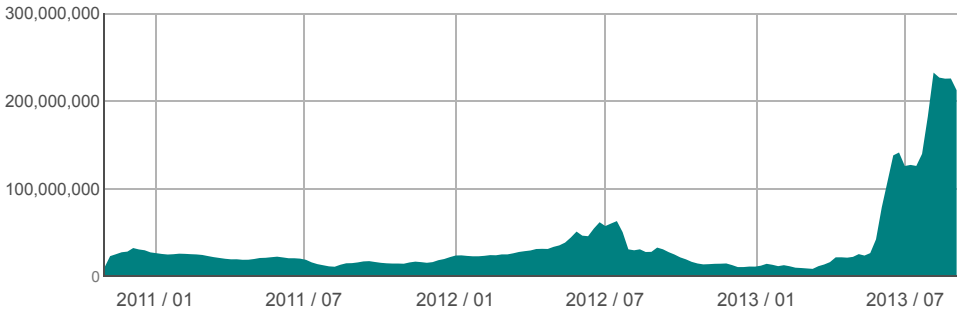

FIGURE 8.1 – Avertissements de navigation montrées aux internautes chaque semaine par Google SafeBrowsing [35]

ne seraient pas possibles dans d'autres scénarios. Aussi, il permet souvent, aux attaquants expérimentés, de couvrir leurs traces de manière efficace.

Cette tendance est bien visible en Figure 8.1, qui montre que le nombre d'avertissements de navigation soulevée par le service Google SafeBrowsing [105] – utilisé par les navigateurs les plus populaires sur le marché (à savoir Google Chrome, Mozilla Firefox, et Apple Safari) - a eu une augmentation de près de 10 fois entre Janvier 2012 et Septembre 2013. Ceci est soutenu aussi par les rapports des entreprises de sécurité, tels que Websense, que, dans son Threat Report 2013 a annoncé une augmentation de près de 600% du nombre de sites Web malveillants sur Internet [126].

Une remarque encore plus intéressante est que, selon le rapport de Websense, 85% des sites qui hébergent du code malveillant sont des hôtes légitimes qui ont été compromis. Des augmentations similaires dans le nombre de domaines malveillants sont signalés également dans le 2013 Internet Security Threat Report de Symantec [117]. Malheureusement, comme le montre le rapport, l'augmentation n'est pas seulement dans le nombre de domaines malveillants, mais aussi dans nombre de vulnérabilités ciblant les cinq premiers navigateurs, passant du nombre de 351 en 2011, jusqu'à 891 vulnérabilités totales déclarées en 2012.

Cela explique pourquoi, au cours des dernières années, les vulnérabilités dans les applications web et dans les navigateurs sont devenus les moyens les plus répandus et de succès utilisés pas les criminels afin de propager du *code malveillant* sur Internet.

Au même temps, les entreprises de sécurité et les grands fournisseurs de services Internet dépensent de plus en plus d'énergies en essayant d'arrêter les acteurs malveillants et la propagation du code malveillant sur le web. Des bonnes pratiques et ressources en terme de sécurité sont souvent portées à l'attention des utilisateurs par le biais de campagnes de sensibilisation, par les écoles, les banques et initiatives nationales (par exemple, le National Cyber-Security Awareness Month aux Etats-Unis et en Europe [23, 27]).Les entreprises emploient généralement des Systèmes de Détection d'Intrusion (IDS) et de Prévention des Intrusions (IPS) [81, 102] afin de détecter ou de bloquer la diffusion de *code malveillant* sur leurs réseaux, tan-

dis que les fournisseurs de services et les moteurs de recherche protègent leurs utilisateurs en utilisant des listes noires (blacklists) ou des méthodes de détection des menaces plus sophistiquées (par exemple, [13, 105, 120]). Néanmoins, toute mesure de sécurité a un coût, et protéger des personnes et des organisations de manière efficace sur Internet est encore un problème ouvert. Nous sommes confrontés à un jeu de chat et souris: les entreprises de sécurité et les chercheurs proposent des techniques avancées pour détecter et bloquer du *code malveillant*, mais les criminels sont en mesure de développer des logiciels malveillants de plus en plus sophistiqués, souvent capables de contourner la protection de ces systèmes.

8.1.2 Modèle d'Attaque

Comme expliqué au début de ce chapitre, le World Wide Web est devenu un écosystème très complexe. Sa nature distribuée, et la grande variété de services et entités y agissant, rendent très difficile l'énumération de tous les types possibles d'attaques qui peuvent arriver sur le web. Cependant, il y a un modèle d'attaque très commun qui aujourd'hui peut être appliquée à une grande variété d'attaques web. Nous ferons référence à ce modèle en tant que *modèle d'attaque web*, et nous aurons l'occasion de l'analyser en profondeur au cours de cette thèse. Notre modèle d'attaque Web prend en considération quatre acteurs principaux: les *attaquants*, les sites vulnérables hébergées par des *fournisseurs d'hébergement*, les *internautes* qui finissent par être victimes d'attaques, et, enfin, les *entreprises de sécurité et les chercheurs* qui surveillent en permanence le web afin de repérer sites web malveillants ou compromis.

Le scénario typique représentant le modèle d'attaque Web que nous adoptons dans cette thèse est représenté en Figure 8.2. Ce modèle comprends les quatre acteurs différents que nous avons mentionnés; cependant, il est une représentation générale, que l'on peut utiliser pour décrire un certain nombre de scénarios d'attaques web spécifiques. Dans chacun de ces scénarios, l'ensemble des acteurs, ou seulement une partie d'entre eux, interagissent d'une manière spécifique, qui peut être complètement différente d'un cas d'attaque à l'autre.

Par exemple, le modèle d'attaque représenté en Figure 8.2 peut être utilisé pour décrire des attaques **drive-by download**. Ces attaques sont aujourd'hui l'un des mécanismes les plus efficaces pour les cybercriminels afin d'infecter des internautes. Dans un scénario drive-by download typique, les attaquants hébergent du code malveillant ciblant les navigateurs les plus communs, sur des sites Web malveillants (ou légitimes et compromis).

En particulier, les attaques drive-by-download installent des logiciels malveillants sur les machines des victimes en exploitant des vulnérabilités dans le navigateur de l'utilisateur ou dans l'un des ses plugins. Pour que cela fonctionne, l'attaquant injecte habituellement du code malveillant (typiquement JavaScript), dans une page web. Quand la victime visite la page malveillante conçue par l'attaquant, le code malveillant est exécuté, et, si le navigateur de la victime est vulnérable, le

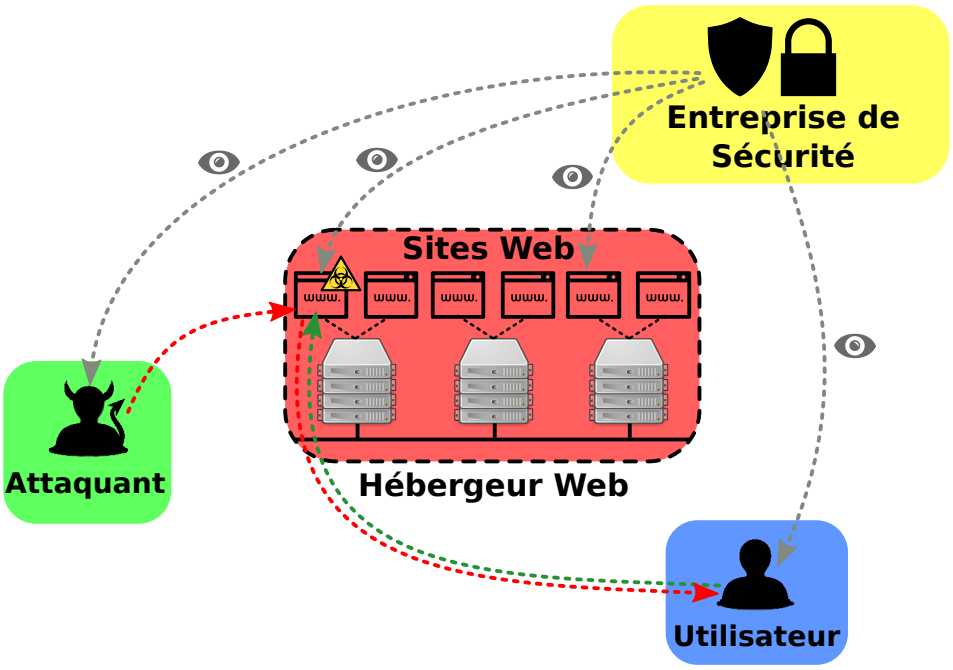

FIGURE 8.2 – Un modèle général d'attaque Web

navigateur est compromis. Cela se conclue typiquement en l'installation de logiciels malveillants sur la machine de la victime. Malheureusement, ce type d'attaque est devenu omniprésent au cours des dernières années [33, 34].

Attaques de ce genre permettent aux criminels d'installer des logiciels malveillants sur une multitude d'ordinateurs, avec relativement peu d'effort, et gagner de l'argent facilement en vendant les informations volées par ces moyens à d'autres criminels.

Une fois une machine est infecté, le programme malveillant se connecte à un serveur de commande et contrôle (C&C) sous le contrôle de l'attaquant et, généralement, reste en attente d'instructions. Cependant, il collecte typiquement des informations privées de la victime, tels que mots de passe, numéros de carte de crédit, adresses email, et texte frappé sur le clavier. De cette façon, les machines infectées forment un *botnet*, qui est un réseau de machines sous le contrôle direct des cyber-criminels. Comme des études récentes ont montré [110, 114], un botnet peut contenir des centaines de milliers d'hôtes compromis, et il peut générer des revenus importants pour le botmaster qui le contrôle. Ce genre d'attaques est étudiée en détail dans le Chapitre 6 de cette thèse.

Un autre type d'attaque très courant, et qui peut être modélisé par le modèle d'attaque Web adoptée dans cette thèse, est le **phishing**. Lors d'une attaque de phishing, le criminel crée un site Web ou un ensemble de pages qui ressemblent en tous points un site cible sur le quel, en général, les utilisateurs gardent des données

sensibles. Il s'agit donc typiquement de sites de banques en ligne, réseaux sociaux ou sites qui fournissent des services de e-mail ou de messagerie privée. Si un utilisateur est poussé avec succès à présenter ses données personnelles (tels que mot de passe, numéro de carte de crédit, etc.) au site faux mis en place par l'attaquant, le criminel peut facilement récupérer les données volées et les vendre sur le marché noir.

Les attaques phishing et drive-by download sont généralement mis en place par des attaquants sur des sites Web compromis. En général, le site compromis est lui-même un autre cas d'attaque Web, consistant généralement en l'accès abusif, de la part de l'attaquant, à la machine hébergeant le site Web. Cet accès abusif est généralement possible grâce à une vulnérabilité dans une application Web utilisé par le site en question (par exemple, vulnérabilité de input validation), ou par d'autres moyens tels que le vol de identifiants d'accès au site. Les deux types d'attaques peuvent être modélisées par notre modèle d'attaque Web, en considérant que deux acteurs: l'attaquant et le fournisseur d'hébergement du site Web cible.

La compromission d'un site Web est elle-même un cas très général d'attaque web, et comme mentionné, peut être effectuée grâce à des moyens différents. Il peut y avoir différentes raisons qui poussent les criminels à compromettre des sites Web. L'une d'eux, comme expliqué précédemment, est d'utiliser l'infrastructure compromise pour lancer de nouvelles attaques contre les visiteurs du site (comme dans le cas des drive-by downloads et du phishing). Dans d'autres cas, toutefois, l'attaque peut être destiné directement à voler des informations sensibles sur la machine qui e été compromise.

En effet, le vol de données sensibles est aujourd'hui l'une des raisons les plus courantes derrière les attaques Web. L'information, et, spécialement, les documents confidentiels ou données privées des utilisateurs et des entreprises, sont très recherchés car peuvent être vendus à des prix élevés sur les marchés noirs ou aux entreprises et aux gouvernements intéressés à voler des secrets industriels ou militaires.

Les grandes fuites d'informations frappent toujours les news en raison de leur portée mondiale, qui généralement concerne des comptes et des informations d'utilisateurs à l'échelle nationale ou mondiale.

Comme la Figure 8.3 montre, un nombre incroyablement élevé de dossiers personnels ont été volés dans des entreprises et des organisations pendant les dernières 9 années. Le nombre total de dossiers volés a déjà dépassé le milliard, et ce qu'en prenant en compte les plus grandes violations de données qui ont frappé les chaînes d'information les plus importantes. Outre le nombre élevé de données volés chaque année, les dernières trois années ont été marquées par un nombre croissant d'attaques pendant les quels des informations très très sensibles ont été volés (montrés dans l'image en couleurs sombres), tels que l'adresse e-mail et mot de passe ou numéro de carte de crédit et informations de compte bancaire. Les chiffres indiqués dans le graphique sont, bien sûr, seule une limite inférieure pour le nombre réel des

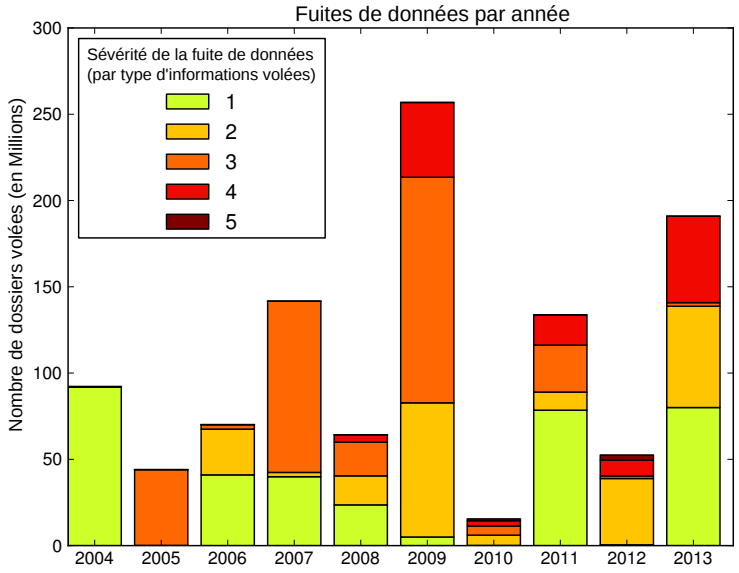

FIGURE 8.3 – Les vols de données les plus connus et documentés publiquement depuis 2004 [46]

données qui ont été divulgués par les organisations dans les dernières années, car il prend en compte que les incidents qui ont été signalés publiquement.

Par ailleurs, le graphique ne tient pas compte de l'énorme quantité de données que les criminels volent des internautes chaque jour, en les attirant avec pages de phishing et ingénierie sociale, ou en infectant leurs ordinateurs avec des logiciels malveillants.

Publiciser des attaques web, et en particulier les vols de données, est une autre tendance de certains des cyber-attaques d'aujourd'hui. Des organisations telles que Anonymous [128] par exemple, diffusent publiquement leurs attaques, parfois même avant l'attaque ait lieu. Tout ça est fait généralement dans le but de frapper les news et diffuser des slogans éthiques ou politiques. La raison évoquée, en faite, pour la plupart des attaques du groupe est "la lutte contre la censure et la promotion de la transparence et de la liberté de l'information". Un nouveau mot a été défini pour décrire ces événements, *hacktivisme*, de *hacker* et *activisme*. Il est à noter que, contrairement aux premiers jours de hacking "vieille école" , l'être un hacktiviste n'est pas un rôle d'élite, réservé à quelques membres élus. Au lieu de cela, tout le monde peut participer à des discussions et des opérations hacktivistes [6] et utiliser des outils gratuits pour anonymiser ses propres traces tout en lançant des attaques de la part de la communauté.

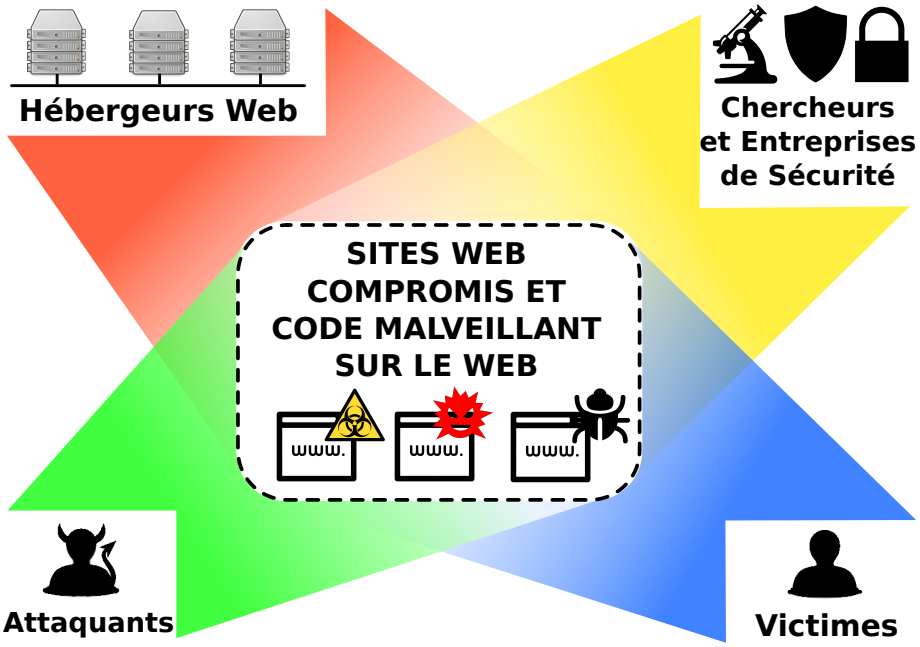

FIGURE 8.4 – Points de vue adoptés dans cette thèse

Enfin, comme montré dans la figure représentant le modèle d'attaque Web, les sociétés de sécurité (ou les chercheurs), même si pas impliqués directement dans les attaques comparés aux autres acteurs, sont généralement présents dans les coulisses de toutes les attaques Web. Lorsqu'ils n'interviennent pas directement en essayant d'arrêter les attaquants ou en aidant les fournisseurs d'hébergement et leurs utilisateurs, en fait, ces entreprises surveillent en permanence les mouvements des criminels connus, analysent sites à la recherche de code malveillant, et recueillent des informations auprès de leurs clients afin de détecter en temps opportun et pouvoir analyser les plus nouvelles menaces d'Internet.

8.1.3 Objectifs

Au cours des dernières années, nous avons assisté à une augmentation du nombre d'attaques web et de sites web compromis installés par les criminels afin de propager leur code malveillant sur Internet. Malheureusement, aujourd'hui cette tendance ne risque pas de s'arrêter, et les entreprises et les gouvernements, aujourd'hui plus que jamais, dépensent énormes quantités d'argent et de ressources, pour lutter contre les activités malveillantes sur Internet.

Un bon plan de prévention d'attaque commence par une bonne compréhension de l'écosystème qui est l'objet de l'attaque: le web, dans notre cas.

C'est la raison pour la quelle, dans cette thèse, nous étudions le problème des sites web compromis à partir de quatre axes d'analyse différents. Ces perspectives nous permettent d'étudier le phénomène des attaques web du point de vue

de chacun des acteurs qui sont typiquement impliqués dans ces types d'attaques: les attaquants, les fournisseurs d'hébergement web qui hébergent sites qui ont été attaqués ou infectés, les chercheurs ou les entreprises de sécurité qui tentent de détecter et de prévenir les attaques, et enfin les utilisateurs, qui représenter les victimes des attaques web. Figure 8.4 représente graphiquement les quatre points de vue que nous adoptons dans cette thèse. Chacun d'eux correspond à un chapitre de cette thèse.

Bien que l'étude du code malveillant sur le web soit un sujet assez commun dans la littérature de la sécurité informatique contemporaine, très peu de travaux ont étudié ce sujet en l'observant à partir des points de vue de ses multiples acteurs.

L'objectif de ce travail est donc de fournir au lecteur une vue d'ensemble du phénomène des sites web compromis et du code malveillant sur le web, et de plonger plus profondément dans ce sujet dès ses quatre points de vue différents décrits ci-dessus. Cela aura l'avantage de donner une vue d'ensemble complète de l'écosystème, en l'observant par ses différents faces. En particulier, nous allons analyser comment les attaquants trouvent leurs cibles sur Internet, comment ils les exploitent et quels sont leurs objectifs une fois qu'ils ont accès à un système compromis. Deuxièmement, nous allons adopter le point de vue des fournisseurs d'hébergement web, qui hébergent des sites qui peuvent être compromis ou infectées par des criminels. Dans ce cas, nous allons étudier comment les hébergeurs agissent en matière de prévention des utilisations abusives de leurs systèmes, comment ils détectent les comptes clients compromis, et la façon dont ils gèrent les incidents et les notifications d'abus. Troisièmement, nous allons étudier comment les chercheurs et les entreprises de sécurité peuvent parcourir Internet à la recherche de pages web infectés portant du code malveillant, en pouvant donc contribuer à la fermeture de sites malveillants avant qu'ils soient capables d'infecter des victimes. Enfin, nous concentrons notre analyse sur le comportement des internautes, afin de comprendre si certaines catégories d'utilisateurs sont plus à risque que d'autres pendant la navigation sur le web, et, si oui, pour étudier comment nous pouvons le prédire en analysant uniquement leur comportement.

8.1.4 Contributions

Cette thèse analyse la diffusion de code malveillant sur le web au cours des dernières années, en se concentrant en particulier sur les sites web compromis. Notre travail de recherche nous permet d'étudier ce phénomène du point de vue des acteurs qui sont couramment impliqués dans celui-ci: les internautes, les fournisseurs d'hébergement, les attaquants et les entreprises de sécurité ou les chercheurs (voir la Figure 8.4): Dans l'ensemble, nous apportons les contributions suivantes au domaine de la sécurité web:

– Nous développons une nouvelle approche pour la collecte et l'analyse des attaques web et de code malveillant, sur la base de la création d'un réseau de sites web vulnérables et entièrement fonctionnels (pots de miel web). Cela nous permet d'étudier et de mieux comprendre les tendances actuelles dans

le domaine des attaques web, et les raisons de ces actions de la part des attaquants.
- Nous présentons une étude de mesure sur comment les sociétés d'hébergement web mutualisé gèrent la sécurité de leurs clients, à l'échelle mondiale. Ceci inclut des tests pour vérifier si des mécanismes de prévention d'attaque sont mis en place, si les hébergeurs sont en mesure de détecter des signes évidents de compromission sur les comptes des leurs clients, et si les plaintes d'abus sont traités d'une manière opportune et appropriée.
- Nous introduisons une nouvelle approche qui utilise analyse statique et techniques de "machine learning" pour la détection à large échelle des pages web qui lancent des attaques de type drive-by-download. Nous montrons en outre que cette approche peut être appliquée avec succès dans un environnement réel dans le but de filtrer efficacement des pages web bénignes, et de transmettre seulement les pages suspectes à des systèmes d'analyse dynamique, plus coûteux en termes de performances.
- Nous développons une nouvelle approche pour la construction de profils d'utilisation pour les internautes, en fonction de leur historique de navigation. Nous analysons si les caractéristiques de chaque profil peuvent être corrélés à des chances plus ou moins élevés de visiter des sites web malveillants, et nous présentons une étude approfondie sur l'efficacité de la prédiction des risques fondée sur les comportements de navigation des utilisateurs du web.

8.2 Fournisseurs d'Hébergement

Nous commençons notre analyse en étudiant les attaques web du point de vue des fournisseurs d'hébergement Web, pour deux raisons. Tout d'abord, la majorité des sites Web sur Internet sont hébergés sur des plates-formes d'hébergement web mutualisé et, généralement, la plupart des attaques web commencent par la compromission d'un site ou d'un compte d'hébergement. Deuxièmement, en étant le point de départ de chaque attaque, si les fournisseurs d'hébergement sont en mesure de détecter des compromissions sur les comptes de leurs clients d'une manière précise et en temps opportun, la plupart des attaques web d'aujourd'hui devraient échouer ou avoir une très courte durée de vie. D'où l'importance d'étudier la détection de compromission du point de vue des fournisseurs d'hébergement.

Dans ce chapitre, nous testons la capacité des fournisseurs d'hébergement web de détecter les sites web compromis et de réagir aux plaintes des utilisateurs. Notre étude couvre également six services spécialisés qui fournissent des services de surveillance pour sites web pour une somme modique.

Pendant une période de 30 jours, nous avons installé nos propres sites vulnérables sur 22 plates-formes d'hébergement mutualisé, y compris 12 des plus populaires. Nous avons exécuté, à plusieurs reprises, cinq types d'attaque différentes contre chacun d'eux. Nos tests comprenaient une infection par botnet, une attaque drive-by download, le téléchargement de fichiers malveillants, une injection SQL

avec vol de données client, et un kit de phishing pour une célèbre banque américaine. En outre, nous avons également généré du trafic en simulant des victimes du phishing. Nous montrons que la plupart de ces attaques auraient pu être détectés par des outils gratuits et libres disponibles sur le web. Après 25 jours, si aucune activité malveillante était détectée, nous avons commencé à déposer des plaintes d'abus aux fournisseurs. Cela nous a permis d'étudier la réaction des fournisseurs d'hébergement Web à des plaintes réelles et fictives.

L'image générale que nous avons pu tirer de notre étude est tout à fait alarmante. La grande majorité des fournisseurs, ou "services add-on de sécurité", sont incapables de détecter les plus simples signes d'activité malveillante sur les sites hébergés.

8.2.1 Introduction

Posséder et opérer un site web est devenu une activité très courante dans de nombreuses régions du monde, et des millions de sites sont crées, tous les jours, à la fois pour un usage personnel et professionnel. Les gens n'ont plus besoin d'être des gurus de l'informatique pour pouvoir gérer un site Web: un navigateur Web, une carte de crédit avec quelques dollars, et des compétences de base en informatique sont généralement assez pour commencer ce type d'activité.

De toutes les façons possibles pour accueillir un site Web, l'hébergement partagé est généralement l'option la plus économique. Elle consiste à avoir un site Web hébergé sur un serveur Web où d'autres sites peuvent résider et partager les ressources de la machine. Grâce à son faible prix, l'hébergement mutualisé est devenu le choix typique pour accueillir la majorité des sites web pour particuliers et PMEs dans le monde entier.

En étant si commun, toutefois, les sites d'hébergement mutualisé ont aussi des grandes chances d'être la cible d'attaques web, et devenir des moyens pour les criminels de propager leur code malveillant sur Internet. En plus, ces sites sont souvent exploités par des utilisateurs avec peu ou pas d'expérience en matière de sécurité, qui sont peu susceptibles d'être en mesure de détecter les attaques ou se servir d'outils de sécurité pour la prévention et la détection d'attaques.

L'analyse présentée dans ce chapitre se concentre sur les services d'hébergement Web mutualisé, et présente une étude sur ce que les fournisseurs d'hébergement font pour aider leurs clients à détecter des signes de compromission sur leur site web. Il s'agit d'un engagement important, compte tenu du fait que les clients des services d'hébergement mutualisé sont les plus vulnérables aux attaques web [58]. En outre, même un client avec des connaissances en matière de sécurité ne serait jamais en mesure de protéger et de surveiller son compte sans la coopération du fournisseur. En fait, dans une plate-forme partagé, l'utilisateur a peu de privilèges sur la machine et n'est pas autorisé à exécuter ou installer des applications de surveillance ou des systèmes de détection d'intrusions, ni gérer le pare-feu, ou les paramètres de sécurité. Ainsi, afin de protéger son site Web, l'utilisateur

doit se fier entièrement sur les mesures de sécurité employées par le fournisseur d'hébergement.

Nous avons également testé les réactions des fournisseurs aux plaintes, et les capacités de détection d'attaque de six services spécialisés dans la surveillance de la sécurité des sites web.

Dans un récent sondage [16], Commtouch et StopBadware ont rapporté les résultats d'un questionnaire dans lequel 600 propriétaires de sites Web compromis ont été posées des questions sur les attaques qui ont visé leurs sites Web. Il est apparu que, parmi les interrogés, 49% des utilisateurs ont été mis au courant de la compromission par un avertissement de leur navigateur, tandis que dans moins de cas, ils ont été informés par leur fournisseur d'hébergement (7%) ou par un service de sécurité (10%). En outre, 14% des utilisateurs qui ont participé à l'enquête ont déclaré que leur fournisseur d'hébergement a retiré le contenu malveillant de leur site web après l'infection. Enfin, seulement 12% des clients étaient satisfaits de la façon dont leur fournisseur d'hébergement a géré la situation, tandis que 28% des utilisateurs qui ont participé à l'enquête envisagent de passer à un nouveau fournisseur à cause de cette expérience.

Inspiré par le rapport StopBadware, nous avons décidé d'analyser systématiquement, sur une plus grande échelle et de façon automatisée, comment les sociétés d'hébergement web se comportent en ce qui concerne la détection des sites web compromis, quelles sont leurs réactions en cas de plaintes d'abus, et comment ils procèdent pour informer un client si son site vient d'être compromis.

8.2.2 Résultats

Ce chapitre présente la première analyse d'envergure mondiale de la qualité et de la fiabilité des activités de surveillance de sécurité réalisées par les fournisseurs d'hébergement web mutualisé. Malheureusement, le résultat général que nous avons tiré de nos analyses est tout à fait alarmant: la grande majorité des fournisseurs et "services add-on de sécurité" sont incapables de détecter les signes les plus simples d'activités malveillantes sur les sites web qu'ils hébergent. Il est important de noter que nous ne voulons pas blâmer ces fournisseurs de ne pas protéger leurs clients, puisque ce service n'est souvent pas partie des conditions du contrat pour le quel les utilisateurs paient. Cependant, nous croyons qu'il serait dans l'intérêt des fournisseurs et du grand public de mettre en œuvre des simples mécanismes de détection, afin de pouvoir identifier rapidement quand un site a été compromis et il est utilisé pour effectuer des activités malveillantes.

Nous pouvons résumer les principales conclusions de nos expériences autour des cinq points suivants:

Inscription - Les majeurs hébergeurs investissent des efforts considérables pour recueillir des informations sur les utilisateurs qui s'inscrivent chez eux. Cette procédure est efficace afin d'empêcher les criminels d'abuser du service d'hébergement et installer des sites web malveillants.

Prévention - Environ 40% des hébergeurs ont déployé quelques mécanismes de sécurité avec le but de bloquer des simples attaques (injections SQL, exploitation des vulnérabilités les plus courantes contre des applications web)

Détection - Une fois que le client est enregistré, la plupart des fournisseurs ne font rien pour détecter des activités malveillantes ou des sites Web compromis, offrant ainsi très peu d'aide à leurs clients. Nous avons été surpris de découvrir que 21 des 22 fournisseurs testés n'ont même pas exécuté un antivirus une fois par mois (ou ils les exécutent avec des signatures très anciennes et pas à jour) sur les sites hébergés. En outre, aucun d'entre eux a considéré comme suspect un site ayant des multiples tentatives de connexion sortantes vers un serveur IRC.

Plaintes d'Abus - Seul le 36% des fournisseurs ont réagi à nos notifications d'abus. Quand ils ont répondu rapidement, la plupart du temps leur réponse était inappropriée ou excessive. Aucun des fournisseurs *globaux* et un seul des *régionaux* étaient en mesure de gérer correctement, à la fois, les plaintes réelles et fausses en temps opportun.

Services de Sécurité - L'utilisation de services de sécurité pas chers comme "add-on" ne fournit aucune couche supplémentaire de sécurité dans nos expériences. En outre, les services qui ont été configurées pour analyser le contenu de nos sites via FTP n'ont pas réussi à découvrir les fichiers malveillants.

Les principales différences entre hébergeurs *globaux* et *régionaux* semblent être en termes de vérification d'inscription (en faveur des *globaux*) et la réaction à de plaintes réelles (en faveur des *régionaux*).

Comme nous l'avons déjà mentionné au début de ce chapitre, les fournisseurs d'hébergement web sont en mesure de jouer un rôle clé dans la sécurité du Web. En fait, ils accueillent des millions de sites qui sont souvent mal gérés par des utilisateurs inexpérimentés, et qui sont susceptibles d'être compromis pour propager du code malveillant ou héberger des kits de phishing. Malheureusement, tous les fournisseurs d'hébergement web partagé que nous avons testé, ont manqué cette occasion.

8.3 Attaquants

Dans ce chapitre, nous analysons les détails de la façon dont les sites Web sont attaqués, à partir du point de vue des attaquants et des criminels qui les exploitent. Les attaques web sont aujourd'hui l'une des principales menaces sur Internet, et plusieurs études les ont analysé, fournissant des détails sur la façon dont ils sont réalisés et comment ils se propagent. Cependant, aucune étude semble avoir suffisamment analysé le comportement typique d'un attaquant *après* un site web a été compromis.

Dans ce chapitre, nous présentons la conception, la mise en œuvre et le déploiement d'un réseau de 500 sites web pot-de-miel (honeypot) entièrement fonctionnels, accueillant une gamme de services, dont le but est d'attirer les attaquants et recueillir des informations sur ce qu'ils font pendant et après leurs attaques. En

100 jours de collecte de données, notre système a reçu, normalisée, et regroupé plus de 85 000 fichiers qui ont été créés pendant environ 6000 attaques. L'étiquetage des groupes nous a permis d'obtenir une vue générale des attaques web, d'identifier le comportement derrière chaque action réalisée à la fois pendant et après l'exploitation d'une application web (tels que l'installation d'une page de phishing, un script de botnet, ou un exploit locale sur la machine compromise.)

8.3.1 Introduction

Les attaques Web sont une des sources les plus importantes de la perte de propriété intellectuelle et financière. Dans les dernières années, ces attaques ont évolué en nombre et en sophistication, en ciblant les gouvernements et les grandes entreprises, en volant des informations personnelles des utilisateurs et en entraînant des importantes pertes financières, jusqu'à plusieurs millions d'euros à chaque occasion. En outre, le nombre de personnes parcourant le web grâce à des ordinateurs, tablettes et smart phones est en constante augmentation, ce qui rend les attaques web un moyen très attrayante pour les criminels.

On retrouve également cette tendance dans le sujet de la recherche universitaire. En fait, un regard rapide sur les articles publiés au cours des dernières années montre comment un grand nombre d'entre eux concernent le WWW et ses attaques et défenses. Certaines de ces études se concentrent sur les vulnérabilités courantes liées aux applications web, serveurs web ou les les navigateurs, et sur la façon dont ces composants peuvent être compromis. D'autres analysent les détails de chaque attaque ou de certaines campagnes [11, 51, 74], ou proposent des nouveaux outils ou mécanismes de protection pour atténuer les attaques existants.

Le résultat est que presque tout le panorama des infections web a été étudié à fond: la façon dont les attaquants scannent le web, l'utilisation de "google dorks" pour trouver des applications vulnérables, la façon dont ils gèrent les attaques automatisées, et la façon dont ils fournissent contenu malveillant pour les utilisateurs finaux. Cependant, il y a encore une pièce manquante dans le puzzle. En fait, avant nous, aucun projet de recherche semble avoir suffisamment détaillé le comportement d'un attaquant typique *pendant* et *après* la compromission d'un site web. Parfois, les attaquants sont seulement à la recherche des informations stockées dans le service lui-même, par exemple lorsque l'objectif est de voler des informations d'identification de l'utilisateur par le biais d'une injection SQL. Mais dans la majorité des cas, l'attaquant veut maintenir l'accès à la machine compromise et l'insérer dans le cadre d'une infrastructure malveillante de plus grande taille (par exemple, pour agir comme un serveur C&C pour un botnet ou servir des documents malveillants aux utilisateurs qui visitent la page).

Bien que la littérature récente souvent met l'accent sur des sujets accrocheurs, comme drive-by-download et black-hat SEO, ceux-ci ne sont que la pointe de l'iceberg. En fait, il y a une grande variété d'activités malveillantes effectuées sur Internet sur des bases quotidiennes, avec des objectifs qui sont souvent différents de

ceux des criminels notoires qui attirent les médias et l'attention des entreprises de sécurité.

La principale raison pour la quelle aucun travail précédent a été fait dans cette direction de recherche est que presque tous les projets existants basés sur les pots de miel Web utilisent des applications simulées. Cela signifie qu'aucune attaque réelle peut être effectuée avec succès, et donc, dans le cas général, que toutes les actions qu'un attaquant pourrait effectuer après l'exploitation ne seront pas collectées.

Par conséquent, afin de mieux comprendre la motivation des différentes catégories d'attaquants, les éditeurs de logiciels antivirus ont souvent compté sur des informations communiquées par leurs clients. Par exemple, dans une étude récente menée par Commtouch et l'organisation StopBadware [16], 600 propriétaires de sites Web compromis ont été invités à remplir un questionnaire afin de rendre compte de ce que l'attaquant avait fait après l'exploitation de leur site. Les résultats sont intéressants, mais l'approche ne peut pas être automatisé, et il est donc difficile de le répéter. Aussi, il n'y a aucune garantie que les utilisateurs (la plupart du temps pas des experts en matière de sécurité) soient été en mesure de distinguer avec succès une classe d'attaque de l'autre.

En ce chapitre, nous fournissons, pour la première fois, une analyse approfondie et globale du comportement des attaquants sur le web. Nous concentrons notre analyse sur deux aspects distincts: i) la phase d'exploitation, dans laquelle nous examinons comment les attaques sont effectuées jusqu'au point où l'application est compromise, et ii) la phase de post-exploitation, dans laquelle nous examinons ce que les attaquants font après qu'ils prennent le contrôle de l'application. La première partie traite des méthodes et des techniques (le *"comment"*) utilisée pour attaquer des applications web, tandis que la seconde partie essaie de déduire les raisons et les objectifs (le *"pourquoi"*) derrière ces attaques.

Pour cette raison, cette étude n'analyse pas des attaques ordinaires de type injections SQL ou cross-site scripting. Au lieu de cela, nos pots de miel sont conçus pour attirer et contrôler les criminels qui sont intéressés à acquérir (et maintenir) le contrôle des applications web. Nos résultats montrent des tendances intéressantes sur la façon dont la majorité de ces attaques sont effectuées à l'état sauvage. Par exemple, nous identifions quatre phases distinctes et 13 buts différents qui sont couramment poursuivis par les attaquants. Nous fournissons également un aperçu de quelques scénarios d'attaque intéressants que nous avons identifiés lors de l'opération de nos pots de miel.

8.3.2 Résultats

Cette étude nous a permis d'analyser, pour la première fois dans la littérature académique, comment et pourquoi les attaques contre les sites communs sont effectués. Nous croyons que les résultats présentés dans cet étude ont fourni des indications intéressantes sur l'état actuel des comportements d'exploitation sur le web. D'un côté, nous avons été en mesure de confirmer les tendances connues pour certaines catégories d'attaques, telles que la prévalence de pays d'Europe orientale

dans l'activité de spamming de commentaires pour forums, et le fait que la plupart des campagnes de phishing et scams sur Internet sont encore exploités par des criminels dans les pays africains [45]. Les annonces pharmaceutiques semblent être les sujet les plus fréquents dans les activités de spamming, comme on retrouve dans d'autres études récentes [20].

D'autre part, nous avons également pu observer et étudier un grand nombre d'attaques manuelles, ainsi que des nombreuses infections qui visent à transformer des serveurs Web en bots IRC. Cela suggère que certaines des menaces qui sont souvent considérées comme dépassées sont effectivement encore très populaires (en particulier entre les jeunes criminels) et sont toujours responsables d'une grande partie des attaques contre les sites vulnérables.

8.4 Les Utilisateurs

Après avoir analysé comment les attaquants Web se comportent et comment les fournisseurs d'hébergement gèrent les compromissions de sites web, il est temps de tourner notre point de vue sur la cible de chaque attaque: l'utilisateur. Les utilisateurs sont généralement la cible finale des attaques web, et il est de leurs données personnelles que les criminels sont typiquement en mesure d'obtenir un gain financier.

Cependant, alors que de nombreux aspects d'attaques web ont été soigneusement étudiés par les chercheurs et les entreprises de sécurité, les raisons qui font que certains utilisateurs sont plus "à risque" que d'autres sont encore inconnues. Pourquoi certains utilisateurs ne rencontrent jamais des pages malveillantes, tandis que d'autres semblent se tomber sur eux sur une base quotidienne ?

Pour répondre à cette question, dans ce chapitre, nous présentons une étude approfondie sur l'efficacité de la prédiction du risque basée uniquement sur le comportement des utilisateurs pendant leur navigation web. Notre analyse est basée sur un ensemble de données de télémétrie recueillies par un des majeurs fournisseurs AntiVirus, comprenant des millions d'URL visités par plus de 100.000 utilisateurs pendant une période de trois mois. Pour chaque utilisateur, nous extrayons les statistiques d'utilisation détaillées, et distillons ces informations dans 74 caractéristiques uniques, qui servent à modeler les différents aspects du comportement de l'utilisateur.

Une fois que nous extrayons toutes les fonctionnalités, nous effectuons une analyse de corrélation pour voir si elles sont en corrélation avec la probabilité de visiter des pages web malveillantes. Ensuite, nous misons sur des techniques d'apprentissage automatique pour fournir des prévisions pour les utilisateurs qui sont exposés au risque. Les résultats des expériences montrent qu'il est possible de prédire avec une précision raisonnable quels sont les utilisateurs les plus susceptibles d'être victimes d'attaques web, en analysant uniquement leur historique de navigation.

8.4.1 Introduction

Une grande quantité de recherches ont été menées sur les outils et techniques adoptées par des attaquants, sur l'identification automatique et l'atténuation des vulnérabilités logicielles, ou sur comment protéger les navigateurs web de l'exploitation. Malgré cet effort, le pourcentage de pages Web qui sont soit malveillantes ou qui ont été compromises pour servir du contenu malveillant est en constante en augmentation [55,116,127]. Même si ceci est certainement un phénomène alarmant, ces chiffres globales sont calculées sur l'ensemble de la population d'Internet, et donc ne parviennent pas à exprimer ce qui est le risque réel pour un utilisateur unique de rencontrer une page malveillante pendant son activité quotidienne. Le nombre croissant de sites dangereux n'affecte pas nécessairement tout le monde de la même façon. Par exemple, il est possible que la majorité des utilisateurs ne naviguent que dans des voisinages "propres" où les pages malveillantes sont encore extrêmement rares. Dans ce cas, il devrait être possible d'associer à chaque utilisateur, en fonction de son comportement habituel, un certain *profil de risque*. En autres termes, il devrait y avoir une corrélation entre les *habitudes de navigation* et la probabilité de se rendre sur des sites potentiellement nocifs. Ce scénario est particulièrement intéressant dans le domaine de la cyber-assurance [9], dans lequel le profil des utilisateurs est une étape importante vers une évaluation précise des risques. Par exemple, dans le monde physique, les prémiums des assurances sont normalement calculés sur la base d'une classification des risques. Par exemple, les assurances automobile sont plus chères dans les grandes villes ou pour les conducteurs inexpérimentés, parce que ces conditions sont connues pour l'être positivement corrélées à la probabilité d'accidents de voiture. Malheureusement, une mesure équivalente de facteurs de risque dans le monde virtuel est toujours inexistante.

Si l'hypothèse d'une corrélation entre le risque et le comportement de navigation est raisonnable, cela n'implique pas nécessairement la présence d'une relation de causalité. Il doit être souligné, cependant, que notre travail se concentre uniquement sur l'analyse de l'*activité volontaire* de navigation des utilisateurs. Nous n'avons donc aucune visibilité sur des visites de URLs qui ne sont pas originées par des actions de l'utilisateur sur un navigateur Internet (telles que des visites à des pages qui font partie des infrastructures de C&C ou des URLs visitées directement via des applications non standard, ou des logiciels malveillants).

Lorsque on traite l'analyse des comportements de navigation des utilisateurs, il y a aussi d'autres facteurs que l'on doit prendre en compte. Par exemple, indépendamment de leur activité quotidienne, les utilisateurs sont souvent poussés à cliquer sur des liens qui leur sont envoyés par e-mail, à travers de méthodes d'ingénierie sociale. Par conséquent, il est possible que d'autres attributs tels que l'expérience informatique d'un utilisateur, comme constaté par Onarlioglu et al. [84] pourraient être importants pour déterminer les facteurs de risque d'un utilisateur pendant sa navigation.

Malheureusement, peu de travaux ont essayé de répondre à cette question et comprendre si il y a certains comportements ou certaines caractéristiques qui peuvent influer sur la probabilité pour des utilisateurs de visiter des pages Web malveillantes. Comme indiqué dans le chapitre 2.3, certains travaux ont tenté de répondre à des questions similaires en effectuant des études de terrain sur l'utilisation de l'ordinateur d'un nombre limité de sujets [61]. D'autres ont proposé l'hypothèse que certains comportements peuvent être liés à de plus grandes chances d'être compromis, comme la relation entre navigation sur des sites porno et le fait d'être sujets à des infections [130]. Cependant, aucune étude n'a jusqu'à présent été assez générale pour créer des profils d'utilisateurs et analyser ces informations afin d'évaluer s'il existe une relation entre les habitudes spécifiques des utilisateurs et le fait de visiter des pages Web malveillantes.

Dans cet étude, nous menons la première enquête exhaustive dans ce domaine, en utilisant les données de télémétrie collectées par une grande société d'antivirus. En particulier, nous avons analysé les pages Web visitées par 160,229 utilisateurs sur une période de 3 mois (92 jours). Par le biais des informations anonymes qui nous ont été fournies, nous avons identifié deux premières catégories d'utilisateurs: les *sûres* – ceux qui n'ont jamais visité des pages Web malveillantes au cours de nos expériences, et les *à risque*, qui ont visité plusieurs sites malveillants dans le même laps de temps. Notre objectif était de voir si ce genre de comportement peut être utilisé pour différencier les deux classes. Pour cette raison, nous avons identifié et extrait 74 attributs qui peuvent être utilisés pour résumer le comportement de navigation d'un utilisateur, et nous avons corrélé chacun d'eux avec la classe d'utilisateur.

Nos expériences confirment que le volume d'activité de l'utilisateur est l'un des meilleurs indicateurs de la niveau de risque. Le plus une personne navigue tous les jours, et le plus diversifiée est l'ensemble de pages visitées, plus il est probable que la personne tombe sur un site Web malveillant. Nous montrons également que les pages malveillantes sont plus susceptibles d'être rencontrées au cours du week-end et que les utilisateurs le plus à risque sont plus actifs pendant la nuit que ceux qui appartiennent à la classe d'utilisateurs sûrs. En regardant les catégories de sites Web, nous avons constaté que certains d'entre eux – tels que les sites à contenu adulte et les "URL shorteners" – sont positivement corrélés à la probabilité d'être à risque. Enfin, les résultats des expériences que nous avons effectuées indiquent qu'il est possible de combiner toutes ces informations dans un classificateur qui permette de prédire si un utilisateur est à risque d'infection, tout en analysant son profil de navigation.

8.4.2 Résultats

Cet étude nous a permis d'analyser les comportements des utilisateurs qui finissent par être victimes d'attaques web. Nous avons présenté une première étape vers la prédiction du risque des utilisateurs lorsqu'ils naviguent sur Internet. Notre analyse en profondeur d'un grand ensemble de données de télémétrie recueillies par

l'une des principales sociétés AntiVirus nous a permis d'acquérir un certain nombre de points de vue sur la relation entre les habitudes de navigation des utilisateurs et leurs chances de visiter des pages Web malveillantes sur Internet. Par exemple, nous avons été en mesure de confirmer certaines tendances connues, telles que le fait que la navigation sur le Web tard dans la nuit et le week-end est généralement corrélée avec plus de chances de se retrouver sur des sites web malveillants. Une autre tendance générale confirmée par notre travail est que, en général, plus un utilisateur navigue sur Internet, plus ses chances sont de se retrouver dans certains "voisinages dangereux".

Nous avons également été en mesure d'attester si le profilage des utilisateurs peut être efficacement utilisé comme une base pour prédire le risque pour un usager de se retrouver sur des sites web malveillants. En utilisant des techniques de machine learning, nous avons montré que ces profils pourraient effectivement être utilisés, au moins dans une certaine mesure, dans la prédiction de la classe de risque d'un utilisateur sur le web, de façon similaire à ce qui se fait actuellement dans le domaine des assurances et de la prédiction du risque financier.

8.5 Entreprises de Sécurité et Chercheurs

Les derniers acteurs qui ont une importante implication dans l'étude et les recherches liées aux attaques web sont sont les entreprises de sécurité, ou des chercheurs en sécurité. Les entreprises de sécurité dépensent beaucoup de ressources afin de détecter les logiciels malveillants et les sites qui les hébergent, car ce derniers représentent une menace sérieuse pour tous les internautes. En particulier, cette partie de la thèse présente le point de vue d'une de entreprise de sécurité qui doit faire face à l'analyse d'un grand nombre de pages web sur une base quotidienne.

Comme nous le savons, au cours des dernières années, les chercheurs ont mis au point un certain nombre de systèmes qui analysent les pages Web pour détecter la présence de code malveillant. La plupart de ces systèmes utilisent techniques d'analyse dynamique, c'est à dire qu'ils exécutent les scripts associés à une page Web directement dans un vrai navigateur (en cours d'exécution dans un environnement virtualisé) ou dans un navigateur émulé, et surveillent le code exécuté pour détecter des activités malveillantes. Alors que ces outils sont très précis, le processus d'analyse est coûteux, nécessitant souvent des dizaines de secondes, ou voir plusieurs minutes, pour analyser une seule page. Par conséquent, l'exécution de cette analyse sur un grand nombre de pages Web peut être prohibitive.

Une approche pour réduire les ressources nécessaires pour effectuer une telle analyse sur grande échelle est de développer un filtre rapide et fiable qui peut rapidement rejeter toutes les pages qui sont bénignes, en transmettant à un système d'analyse dynamique seules les pages qui sont susceptibles de contenir du code malveillant. Nous décrivons ici la conception et la mise en œuvre d'un tel filtre. Le filtre que nous proposons, appelé *Prophiler*, utilise des techniques d'analyse sta-

tique pour examiner rapidement une page web. Cette analyse prend en compte les caractéristiques dérivées à partir du contenu HTML d'une page, à partir du code JavaScript associé, et de l'URL correspondant. Nous obtenons automatiquement des modèles de détection qui utilisent ces fonctionnalités en utilisant des techniques d'apprentissage automatique appliquées aux ensembles de données étiquetées.

Afin de démontrer l'efficacité de *Prophiler*, nous avons recueilli des millions de pages, que nous avons analysées pour individuer des comportements malveillants. Nos résultats montrent que notre filtre est capable de réduire le temps d'analyse de plus de 85% par rapport à l'usage d'un outil d'analyse dynamique, avec une quantité négligeable de détections manqués.

8.5.1 Introduction

Les attaques à partir de pages web malveillantes sont devenus l'un des mécanismes les plus efficaces pour les cyber-criminels d'infecter des utilisateurs sur Internet. En particulier, les attaquants utilisent souvent des exploits drive-by-download afin de compromettre un grand nombre d'utilisateurs. Pour effectuer une attaque drive-by-download, l'attaquant développe du code malveillant côté client (généralement écrit en JavaScript) qui cible une vulnérabilité dans un navigateur Web ou dans l'un des plugins du navigateur. Ce code est injecté dans des sites Web compromis ou est simplement hébergé sur un serveur sous le contrôle des criminels. Quand une victime visite une page web malicieuse, le code malveillant est exécuté, et, si le navigateur de la victime est vulnérable, le navigateur est compromis. En conséquence, l'ordinateur de la victime est généralement infecté par des logiciels malveillants.

Les attaques drive-by-download sont devenus omniprésents au cours des dernières années, et des exemples du monde réel montrent que même des sites web légitimes (et probablement bien entretenus) sont souvent compromis et injectés avec du code malveillant [33, 34].

Compte tenu de la menace grandissante posée par les pages Web malveillantes, il n'est pas surprenant que les chercheurs ont commencé à étudier les techniques pour protéger les internautes. Actuellement, la protection la plus répandue est basée sur des listes noires d'URLs. Ces listes noires (comme Google SafeBrowsing [105]) réunissent des listes de URLs qui ont été détectés comme malveillants. Les listes sont interrogés par un navigateur avant de visiter une page web. Lorsque l'URL se trouve sur la liste noire, la connexion est interrompue ou un avertissement est affiché. Bien sûr, pour être en mesure de construire et d'entretenir une telle liste noire, des mécanismes de détection automatisées sont nécessaires.

Les outils de choix pour l'identification des pages Web malveillantes sont les (high-interaction) honeyclients. Ces honeyclients, comme le MITRE HoneyClient [43], Microsoft HoneyMonkey [124], Capture-HPC [107], ou Google Safe Browsing [92], lancent un navigateur sur un véritable système d'exploitation dans une machine virtuelle. Le navigateur est dirigé vers le URL qui doit être analysé. Après la page web est chargée, les système vérifie la présence d'objets qui in-

diquent une attaque réussie, tels que les fichiers exécutables sur le système ou l'exécution de procès inattendus. Bien que la présence de ces objets soit une preuve solide qu'une page est malveillante, l'inconvénient des systèmes à haute interaction est le fait que l'analyse est chère. Alors qu'une parallélisation peut aider dans le traitement de plusieurs pages plus efficacement, le contenu actif (comme le JavaScript) doit toujours être exécuté. En outre, après chaque exploit réussi, la machine virtuelle doit être restauré, car la plate-forme d'analyse ne peut plus être considérée comme fiable. En conséquence, l'analyse d'une seule URL peut facilement nécessiter plusieurs minutes.

Outre que les honeyclients à haute interaction, les chercheurs ont proposé des approches alternatives pour la détection de pages Web malveillantes. En particulier, un certain nombre d'outils ont été proposés (comme Wepawet [17], PhoneyC [79], JSUnpack [54]) qui s'appuient sur l'instrumentation de l'environnement d'exécution JavaScript afin de détecter l'exécution de scripts malveillants, ou seulement un certain type d'attaques (comme NOZZLE [98], un outil pour la détection des attaques de type heap-spraying sur les pages Web malveillantes). Par rapport aux honeyclients à haute interaction, ces systèmes fournissent une meilleure vue dans le fonctionnement interne des scripts malveillants, et ils exigent moins d'effort pour le support d'une large gamme de plugins vulnérables. Cependant, ils ne sont pas sensiblement plus rapides, avec des temps d'analyse allant de quelques secondes à quelques minutes pour une seule page [17].

Malheureusement, le temps d'analyse limite directement l'échelle à la quelle ce type de systèmes peuvent être appliqués. Par conséquent, il devient très coûteux (sinon impossible) d'analyser des millions d'URL en une journée. Cette situation est problématique, tant pour les organisations qui cherchent à maintenir des listes noires avec une bonne couverture (comme Google), mais aussi, plus généralement, pour tous ceux dont le but est d'obtenir une vaste et détaillé compréhension des activités malveillantes sur Internet. Une approche pour dépasser ce problème est de concevoir un filtre efficace qui puisse rapidement "jeter" les pages bénignes. En utilisant un tel filtre en frontal à un système d'analyse plus coûteux, on pourrait économiser une grande quantité de ressources, car l'analyse plus précise mais coûteuse est réalisée uniquement sur les pages qui sont susceptibles de contenir du code malveillant. Bien sûr, il faut être en mesure de régler la sensibilité du filtre en fonction de la capacité d'analyse disponible et le niveau acceptable de faux négatifs (détections manquées). Dans ce contexte, les faux positifs sont moins critiques parce que même s'ils conduisent à un gaspillage de ressources (c'est à dire pages bénignes analysés en utilisant des procédures coûteuses), ils n'augmentent pas l'exposition des utilisateurs aux menaces.

Nous présentons ici la conception et la mise en œuvre d'un système de filtrage, appelé *Prophiler*, dont le but est de distinguer rapidement entre les pages qui sont susceptibles d'être malveillantes, et les bénignes. *Prophiler* analyse les caractéristiques de la page HTML, du code JavaScript, et de l'URL associé de façon statique, à l'aide d'un certain nombre de modèles qui sont dérivés en utilisant un apprentissage automatique (machine learning) supervisé. Les pages qui sont détectés comme

malveillantes par *Prophiler* peuvent ensuite être analysés plus en profondeur par un système de détection plus précis (et coûteux), tel que Wepawet.

Vue que la page Web en cours d'analyse n'est pas rendue et aucun scripts est exécuté, l'analyse est rapide. Par rapport aux travaux antérieurs qui tentent de détecter les pages Web malveillantes en fonction du contenu de la page, notre analyse s'appuie sur un ensemble beaucoup plus complet de fonctionnalités, et, par conséquent, donne des résultats plus précis. Les chercheurs ont également suggéré l'identification des pages malveillantes basée sur des caractéristiques extraites de la seule URL. Cette approche fournit de bons résultats pour des pages de scam ou phishing, car les URL correspondantes sont souvent fabriqués par les attaquants afin de tromper les utilisateurs. Toutefois, lorsque le contenu malveillant (tel qu'un drive-by-download) est injecté dans une page légitime, l'URL n'est pas affecté. Par conséquent, dans notre scénario, les systèmes basés exclusivement sur les caractéristiques d'URL souffrent d'une quantité importante de faux négatifs, comme indiqué dans nos expériences.

La nécessité d'un filtre rapide pour permettre l'analyse à grande échelle des pages malveillantes a déjà été reconnue par Provos et al. [92]. Malheureusement, pour des raisons évidentes, très peu de détails ont été révélés sur le filtre de Google. Bien sûr, l'existence de la liste noire de Google apporte la preuve que l'ensemble du système fonctionne. Néanmoins, nous pensons qu'il y a des avantages significatifs en décrire les détails techniques de notre approche de filtrage dans la littérature: d'abord, nous introduisons un ensemble complet de caractéristiques de la page et de l'URL pour identifier les pages web malveillantes. Cela permet de construire d'autres filtres similaires, faisant un meilleur usage des ressources disponibles. Deuxièmement, nous discutons des compromis entre les faux négatifs et les faux positifs, et nous comparons les performances de notre filtre à un certain nombre de systèmes précédents. Troisièmement, nous démontrons que notre filtre nous permet d'améliorer considérablement la portée de l'analyse qui peut être effectuée dans le cas d'un système accessible au public, appelé Wepawet.

8.5.2 Résultats

Nous avons ici décrit la mise au point et l'évaluation de *Prophiler*, un système dont le but est de réaliser un filtre qui permet de réduire le nombre de pages Web qui doivent être analysés de manière dynamique, afin d'identifier les pages Web malveillantes. Nous avons déployé notre système comme un front-end pour Wepawet, un système bien connu pour l'analyse dynamique de pages web, qui est accessible à tout public. Les résultats montrent que *Prophiler* est en mesure de réduire considérablement la charge du système Wepawet avec un très faible taux de faux négatifs. L'outil est actuellement disponible sur demande, en tant que projet open source fourni sous la licence GNU Affero General Public License (AGPL). Nous croyons que l'approche décrite dans ce chapitre peut être utilisé par toute entreprise de sécurité ou de recherche afin de classer et de filtrer des centaines de milliers de pages web par jour de manière efficace. Par ailleurs, en couplant un tel

système avec une outil d'analyse dynamique, *Prophiler* permettrait de construire de manière pro-active des listes noires de pages web hébergeant des exploits drive-by download d'une manière très efficace, permettant la protection des internautes en temps opportun.

8.6 Conclusions

Au cours des dernières années, nous avons été témoins d'un nombre toujours croissant d'incidents de sécurité sur le web. Comme nous l'avons présenté au début de cette thèse, quatre sont les principaux acteurs impliqués dans la grande majorité des attaques web qui se propagent à partir de code ou de sites web compromis et malveillants. Ces acteurs sont: les attaquants, les sites vulnérables hébergées dans les locaux des fournisseurs d'hébergement, les victimes d'attaques (utilisateurs Internet), et les entreprises de sécurité et les chercheurs qui combattent les cyber-criminels en détectant et en analysant toute sorte de code malveillant sur le web.

La littérature existante a souvent analysé chacun de ces quatre acteurs en adoptant un point de vue *externe*, par exemple, en analysant a posteriori les traces d'une compromission, mais sans avoir visibilité sur ce que l'attaquant a effectivement fait lors de l'attaque. Il était donc nécessaire d'adopter une nouvelle approche pour analyser les attaques web du point de vue de chacun de ces quatre acteurs. Cette thèse est, à notre connaissance, la première étude d'attaques web et des sites web compromis qui ait analysé ces phénomènes en adoptant les quatre points de vue différents de ses différents acteurs. En outre, tous les quatre points de vue considérés ont été analysées à travers des études à grande échelle, ce qui rend nos analyses importantes et significatives sur une échelle mondiale.

Dans le chapitre 3, nous avons présenté une étude à échelle mondiale sur la façon dont les sociétés d'hébergement partagé gèrent la sécurité de leurs clients. Les résultats montrent que, même si une bonne partie des sociétés d'hébergement mettent en place des mécanismes de prévention d'attaque simples, presque aucun d'entre eux est capable de détecter des signes évidents de compromission sur les comptes de leurs clients. En outre, nous avons montré que les plaintes d'abus ne sont pas toujours traitées de manière opportune et appropriée, laissant parfois l'espace à un utilisateur malveillant de tromper les fournisseurs en visant des sites Web légitimes avec des plaintes illégitimes.

Le chapitre 4 a ensuite rapporté sur une étude à grande échelle des comportements des attaquants web, réalisé par la mise en place d'un réseau de 500 sites web entièrement fonctionnels et vulnérables, et en observant les actions des attaquants *pendant* et *après* l'exploitation des vulnérabilités sur les applications web. Cela nous a permis de gagner un point de vue unique sur les raisons derrière les attaques qui frappent la majorité des sites de petites et moyennes entreprises sur Internet. En particulier, alors que la majorité des attaques semblent utiliser une combinaison de techniques d'exploitation automatisés pour compromettre les vulnérabilités

connues sur les sites web, nous avons vérifié qu'il y a encore une fraction non négligeable d'attaques réalisés par des "novices", qui mènent des attaques à la main et semblent effectuer ce type d'actions seulement pour le plaisir ou pour se vanter.

Nous avons également présenté, comme indiqué dans le chapitre 5, une première étude des risques web basé sur le profil des utilisateurs, où nous avons analysé un grand ensemble de données comprenant 3 mois d'historique de navigation des clients d'une des majeurs entreprises de sécurité. Cela nous a permis d'analyser et de comprendre les relations entre le comportement d'un utilisateur, et sa probabilité de tomber sur des sites Web malveillants. Par exemple, nous avons assisté à une augmentation du pourcentage de sites Web malveillants visités pendant la nuit et les week-ends, et une augmentation de la possibilité de visiter les sites Web malveillants pour les utilisateurs qui naviguent sur Internet beaucoup, ou qui visitent une grande variété de sites Web. Nous avons également montré que le profil des utilisateurs pourrait être un moyen viable pour prédire la classe de risque d'un utilisateur sur le web, de façon similaire à ce qui se fait actuellement dans le domaine des assurances et de la prédiction du risque financier.

Enfin, tel que présenté dans le chapitre 6, nous avons introduit une nouvelle approche pour les entreprises de sécurité et les chercheurs de détecter les pages Web qui lancent des attaques de type drive-by-download. Notre approche, mise en œuvre dans un système appelé *Prophiler*, utilise analyse statique et des techniques d'apprentissage automatique pour la détection à grande échelle des pages Web malveillantes. Nous montrons que ce système peut être appliqué avec succès dans des scénarios du monde réel, et qu'il est capable de filtrer efficacement les pages bénignes à partir de grandes quantités de pages Web, permettant d'épargner l'analyse coûteuse de la plupart des pages web non suspectes.

Bien que certains des points de vue que nous avons explorés dans cette thèse ont été largement étudiés dans les universités (par exemple, la détection de pages Web malveillantes par des chercheurs), ceci n'est pas le cas pour d'autres. Pour cette raison, il peut être envisagé de travailler sur des orientations futures dans ce domaine de recherche. Par exemple, afin de fournir même un aperçu plus complet sur les facteurs humains liés à la visite de pages Web malveillantes, il serait intéressant d'avoir accès à des informations sociales des utilisateurs, telles que leur sexe, âge, profession et les intérêts personnels, dans une étude similaire à celle que nous avons présenté dans le chapitre 5. Ceci permettrait d'améliorer l'exhaustivité, et sans doute la précision de détection des risques, des profils d'utilisateurs construits avec une telle approche. Malheureusement, il y a des soucis éthiques à considérer avant d'être en mesure d'accorder aux chercheurs le droit d'accéder à toutes ces informations, et de publier les résultats de leurs travaux. C'est pourquoi, dans notre étude, nous nous sommes limités à analyser des informations anonymisées sur les habitudes de navigation des clients.

D'autres études pourraient être menées pour faire avancer nos recherches sur les attaquants web, par exemple en déployant des sites pot de miel sur des sites célèbres ou de grande taille, tels que des plates-formes d'information, des magasins en ligne, des banques et des sites web gouvernementaux. Cela permettrait aux

chercheurs d'acquérir des connaissances aussi sur le comportement des attaquants visant à des objectifs de haut niveau, qui, nous croyons, devrait être tout à fait différent des buts de l'attaquant moyen que nous avons dépeint dans le chapitre 4.

Une extension similaire pourrait être prévue également pour une analyse plus complète sur le comportement des fournisseurs d'hébergement en relation avec le sites web compromis. Outre les fournisseurs d'hébergement mutualisé et les services de protection de sites web pas chers, en fait, aucune recherche n'a encore été menée sur la façon dont les hébergeurs haut de gamme (par exemple, ceux qui fournissent des serveurs cloud ou des serveurs dédiés virtuels) et les services coûteux de sécurité du site Web gèrent les compromissions sur les sites de leurs clients. Cela pourrait être une extension intéressante de notre étude.

En conclusion, cette thèse a montré que l'assurer aux utilisateurs de pouvoir naviguer sur le Web en toute sécurité est aujourd'hui une tâche très difficile. Les cyber-criminels sont nombreux et bien organisés, et ont souvent accès à des bonnes techniques et ressources financières. Heureusement, d'un côté, de nombreuses entreprises de sécurité et des nombreux chercheurs sont constamment en train de monitorer Internet, à la recherche de code malveillant, dans le but d'arrêter les criminels et la mise en place de sites Web malveillants. Les mêmes sociétés de sécurité protègent habituellement les utilisateurs et d'autres entreprises avec des moyens de protection qui peuvent être utilisés pour protéger un utilisateur régulier pendant sa navigation Internet, jusqu'à un certain point. De l'autre côté, cependant, nous avons vu que les sites web compromis et malveillants sont la méthode de choix pour les criminels de propager leur code malveillant sur Internet. Comme montré dans le chapitre 3, la majorité des fournisseurs d'hébergement web, qui accueillent la majorité des sites sur le web, font très peu ou rien afin de détecter même les signes les plus simples de compromission, et ne sont souvent même pas à l'hauteur de gérer des plaintes d'abus. Une bonne capacité de détection de compromission, et la réponse rapide aux plaintes d'abus sont les moyens nécessaires pour permettre de détecter – et arrêter – en temps opportun des sites web malveillants. Pour ces raisons, nous croyons que les fournisseurs d'hébergement Web doivent jouer un rôle clé pour faire d'Internet un lieu plus sûr. Même une petite amélioration dans la façon dont ils gèrent les compromissions de sites Web et les notifications d'abus, signifierait beaucoup plus de chances d'arrêter les attaques Web à leurs premiers stades, par rapport à ce qui est fait aujourd'hui. En conséquence, la mise en place de sites web compromis et malveillants deviendrait sans doute une tâche beaucoup plus difficile pour la grande majorité des attaquants web.

Bibliography

[1] Alexa. Alexa Browse by Category. http://www.alexa.com/topsites/category/Top, 2013.

[2] Alexa. Alexa Top Websites. http://www.alexa.com/topsites, 2013.

[3] Alexa.com. Alexa Top Global Sites. http://www.alexa.com/topsites/.

[4] amada.abuse.ch. Malware Database (AMaDa) :: AMaDa Blocklist. http://amada.abuse.ch/blocklist.php?download=domainblocklist, 2013.

[5] Angelo Dell'Aera. Thug: a new low-interaction honeyclient. http://www.honeynet.org/files/HPAW2012-Thug.pdf, March 2012.

[6] Anonymous. How to join Anonymous. http://anoninsiders.net/how-to-join-anonymous-1527/index.html, April 2013.

[7] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell. State of the art: Automated black-box web application vulnerability testing. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 332–345. IEEE, 2010.

[8] C. M. Bishop. nformation Science and Statistics. In *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] R. Böhme and G. Schwartz. Modeling cyber-insurance: Towards a unifying framework. In *Ninth Workshop on the Economics of Information Security (WEIS)*, 2010.

[10] K. Borgolte, C. Kruegel, and G. Vigna. Delta: Automatic identification of unknown web-based infection campaigns. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2013.

[11] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: The commoditization of malware distribution. In *Proceedings of the USENIX Security Symposium*, 2011.

[12] X. Chen, B. Francia, M. Li, B. Mckinnon, and A. Seker. Shared information and program plagiarism detection. *Information Theory, IEEE Transactions on*, 50(7):1545–1551, 2004.

[13] Cisco Systems, Inc. SpamCop. http://www.spamcop.net/.

[14] Clam AntiVirus. http://www.clamav.net/, 2010.

[15] A. Clark and M. Guillemot. CyberNeko HTML Parser. http://nekohtml.sourceforge.net/.

[16] Commtouch and StopBadware. Compromised Websites - An Owner's Perspective. http://stopbadware.org/pdfs/compromised-websites-an-owners-perspective.pdf, February 2012.

[17] M. Cova, C. Kruegel, and G. Vigna. Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code. In *Proceedings of the International World Wide Web Conference (WWW)*, 2010.

[18] N. Cristianini and J. Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. In *Cambridge University Press*, 2000.

[19] C. Curtsinger, B. Livshits, B. G. Zorn, and C. Seifert. Zozzle: Fast and precise in-browser javascript malware detection. In *USENIX Security Symposium*, pages 33–48, 2011.

[20] Cyberoam Technologies and Commtouch. Internet Threats Trend Report October 2012. http://www.cyberoam.com/downloads/ThreatReports/Q32012InternetThreats.pdf, october 2012.

[21] W. de Vries. Hosting provider antagonist automatically fixes vulnerabilities in customers' websites. https://www.antagonist.nl/blog/2012/11/hosting-provider-antagonist-automatically-fixes-vulnerabilities-in-customers-websites, November 2012.

[22] J. Delgado and R. Davidson. Knowledge bases and user profiling in travel and hospitality recommender systems. In *Proceedings of the ENTER 2002 Conference*, pages 1–16. Citeseer, 2002.

[23] Department of Homeland Security. Stop.Think.Connect. https://www.dhs.gov/stopthinkconnect.

[24] Dshield web honeypot project. https://sites.google.com/site/webhoneypotsite/, 2009.

[25] B. Eshete, A. Villafiorita, and K. Weldemariam. Binspect: Holistic analysis and detection of malicious web pages. In *Security and Privacy in Communication Networks*, pages 149–166. Springer, 2013.

[26] S. Esser. evalhook. http://www.php-security.org/downloads/evalhook-0.1.tar.gz, may 2010.

[27] European Cybercrime Centre. European Cybercrime Centre (EC3) calls on young crime fighters everywhere. https://www.europol.europa.eu/ec3/safer-internet-day-2013, 2013.

[28] B. Feinstein and D. Peck. Caffeine Monkey: Automated Collection, Detection and Analysis of Malicious JavaScript. In *Proceedings of the Black Hat Security Conference*, 2007.

[29] fyicenter.com. Credit card number generator - test data generation. http://sqa.fyicenter.com/Online_Test_Tools/Test_Credit_Card_Number_Generator.php, 2010.

[30] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A Framework for Detection and Measurement of Phishing Attacks. In *Proceedings of the Workshop on Rapid Malcode (WORM)*, 2007.

[31] Google Hack Honeypot. http://ghh.sourceforge.net/, 2005.

[32] Global Research & Analysis Team (GReAT), Kaspersky Lab. The "Red October" Campaign - An Advanced Cyber Espionage Network Targeting Diplomatic and Government Agencies. http://www.securelist.com/en/blog/785/The_Red_October_Campaign_An_Advanced_Cyber_Espionage_Network_Targeting_Diplomatic_and_Government_Agencies, January 2013.

[33] D. Goodin. SQL injection taints BusinessWeek.com. http://www.theregister.co.uk/2008/09/16/businessweek_hacked/, September 2008.

[34] D. Goodin. Potent malware link infects almost 300,000 webpages. http://www.theregister.co.uk/2009/12/10/mass_web_attack/, December 2010.

[35] Google Security Team. Making the web safer. http://www.google.com/transparencyreport/safebrowsing/.

[36] Gregg Keizer. Is Stuxnet the 'best' malware ever? http://www.infoworld.com/print/137598, September 2010.

[37] Y. G.U. and K. M.G. *An Introduction to the Theory of Statistics (14th ed.)*. .Charles Griffin & Co., 1968.

[38] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10–18.

[39] Heritrix. http://crawler.archive.org/.

[40] M. Hines. Malware SEO: Gaming Google Trends and Big Bird. http://securitywatch.eweek.com/seo/malware_seo_gaming_google_trends_and_big_bird.html, November 2009.

[41] W. Hobson. Cyber-criminals use SEO on topical trends. http://www.vertical-leap.co.uk/news/cybercriminals-use-seo-on-topical-trends/, February 2010.

[42] M. Hofer and S. Hofer. ftp-deploy. http://bitgarten.ch/projects/ftp-deploy/, 2007.

[43] HoneyClient Project Team. HoneyClient. http://www.honeyclient.org/, 2010.

[44] A. Ikinci, T. Holz, and F. Freiling. Monkey-Spider: Detecting Malicious Websites with Low-Interaction Honeyclients. In *Proceedings of Sicherheit, Schutz und Zuverlässigkeit*, 2008.

[45] Imperva Inc. Imperva's Web Application Attack Report. `http://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed2.pdf`, january 2012.

[46] informationisbeautiful.net. Information is Beautiful: Data Breaches (public). `http://bit.ly/bigdatabreaches`, July 2013.

[47] L. Invernizzi, P. M. Comparetti, S. Benvenuti, C. Kruegel, M. Cova, and G. Vigna. Evilseed: A guided approach to finding malicious web pages. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 428–442. IEEE, 2012.

[48] G. Jacob, P. M. Comparetti, M. Neugschwandtner, C. Kruegel, and G. Vigna. A static, packer-agnostic filter to detect similar malware samples. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 102–122. Springer, 2013.

[49] J. Jang, M. Woo, and D. Brumley. Towards Automatic Software Lineage Inference. In *Proceedings of the USENIX Security Symposium*, 2013.

[50] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi. deseo: combating search-result poisoning. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 20–20, Berkeley, CA, USA, 2011. USENIX Association.

[51] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi. deSEO: Combating Search-Result Poisoning. In *Proceedings of the USENIX Security Symposium*, 2011.

[52] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi. Heat-seeking honeypots: design and experience. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 207–216, New York, NY, USA, 2011. ACM.

[53] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi. Heat-seeking honeypots: design and experience. In *Proceedings of the International World Wide Web Conference (WWW)*, 2011.

[54] JSUnpack. `http://jsunpack.jeek.org`, 2010.

[55] Kaspersky. Kaspersky Security Bulletin 2012. `http://www.securelist.com/en/analysis/204792255/Kaspersky_Security_Bulletin_2012_The_overall_statistics_for_2012`, 2012.

[56] C. Ke, J. Oliver, and Y. Xiang. Analysis of the Australian Web Threat Landscape. `http://www.trendmicro.com.au/cloud-content/au/pdfs/security-intelligence/white-`

papers/australian_web_threat_landscape_-v7.pdf, May 2013.

[57] J. Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, 3, Supplement(0):91 – 97, 2006.

[58] Larry Ullman. Understand your hosting, five critical e-commerce security tips in five days. Peachpit Blog, Feb. 2011. http://www.peachpit.com/blogs/blog.aspx?uk=Understand-Your-Hosting-Five-Critical-E-Commerce-Security-Tips-in-Five-Days.

[59] T. B. Lee. Here's Everything We Know About PRISM to Date. http://www.washingtonpost.com/blogs/wonkblog/wp/2013/06/12/heres-everything-we-know-about-prism-to-date/, June 2013.

[60] C. Leita and M. Dacier. Sgnet: A worldwide deployable framework to support the analysis of malware threat models. In *Dependable Computing Conference, 2008. EDCC 2008. Seventh European*, may 2008.

[61] F. L. Lévesque, J. Nsiempba, J. M. Fernandez, S. Chiasson, and A. Somayaji. A Clinical Study of Risk Factors Related to Malware Infections. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, Nov. 2013.

[62] A. Liaw and M. Wiener. Classification and regression by randomforest. In *R News*, volume 2/3, page 18, 2002.

[63] P. Likarish, E. Jung, and I. Jo. Obfuscated Malicious Javascript Detection using Classification Techniques. In *Proceedings of the Conference on Malicious and Unwanted Software (Malware)*, 2009.

[64] M. D. List. Malware Domains List. http://www.malwaredomainlist.com/, 2013.

[65] URL Shortening Services - A List of URL Shorteners. http://longurl.org/services, 2013.

[66] L. Lu, R. Perdisci, and W. Lee. Surf: detecting and measuring search poisoning. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 467–476, New York, NY, USA, 2011. ACM.

[67] J. Ma, L. Saul, S. Savage, and G. Voelker. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2009.

[68] G. Maier, A. Feldmann, V. Paxson, R. Sommer, and M. Vallentin. An assessment of overt malicious activity manifest in residential networks. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 144–163. Springer, 2011.

[69] Malcode. Malcode. `http://malc0de.com/bl/BOOT`, 2013.

[70] D. W. McDonald and M. S. Ackerman. Expertise recommender: a flexible recommendation system and architecture. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 231–240. ACM, 2000.

[71] S. E. Middleton, N. R. Shadbolt, and D. C. De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88, 2004.

[72] T. Moore and R. Clayton. Examining the impact of website take-down on phishing. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, eCrime '07, pages 1–13, New York, NY, USA, 2007. ACM.

[73] T. Moore and R. Clayton. The consequence of non-cooperation in the fight against phishing. In *eCrime Researchers Summit, 2008*, pages 1 –14, oct. 2008.

[74] T. Moore and R. Clayton. Evil searching: Compromise and recompromise of internet hosts for phishing. In *Financial Cryptography*, pages 256–272, 2009.

[75] T. Moore and R. Clayton. Financial cryptography and data security. chapter Evil Searching: Compromise and Recompromise of Internet Hosts for Phishing, pages 256–272. Springer-Verlag, Berlin, Heidelberg, 2009.

[76] A. Moshchuk, T. Bragin, S. Gribble, and H. Levy. A Crawler-based Study of Spyware in the Web. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2006.

[77] Mozilla Foundation. Rhino: JavaScript for Java. `http://www.mozilla.org/rhino/`.

[78] M. Müter, F. Freiling, T. Holz, and J. Matthews. A generic toolkit for converting web applications into high-interaction honeypots, 2007.

[79] J. Nazario. PhoneyC: A Virtual Client Honeypot. In *Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2009.

[80] V. Nicomette, M. Kaâniche, E. Alata, and M. Herrb. Set-up and deployment of a high-interaction honeypot: experiment and lessons learned. *Journal in Computer Virology*, june 2010.

[81] NIST. Guide to Intrusion Detection and Prevention Systems (IDPS). `http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf`, 2007.

[82] Number 7. osCommerce 'categories.php' Arbitrary File Upload Vulnerability, November 2010. `http://www.securityfocus.com/bid/44995/info`.

[83] L. Olejnik, C. Castelluccia, and A. Janc. Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns. In *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012)*, Vigo, Espagne, July 2012.

[84] K. Onarlioglu, U. O. Yilmaz, E. Kirda, and D. Balzarotti. Insights into User Behavior in Dealing with Internet Attacks. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, Feb. 2012.

[85] D. Oswald. HTMLParser. `http://htmlparser.sourceforge.net/`.

[86] OWASP foundation. OWASP Top Ten Project. `https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project`, 2013.

[87] OWASP foundation and TrustWave SpiderLabs. Owasp modsecurity core rule set project. `https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project`, 2012.

[88] Y. Peng, G. Wang, G. Kou, and Y. Shi. An empirical study of classification algorithm evaluation for financial risk prediction. *Applied Soft Computing*, 11(2):2906 – 2915, 2011. <ce:title>The Impact of Soft Computing for the Progress of Artificial Intelligence</ce:title>.

[89] F. Pouget, M. Dacier, and V. H. Pham. V.h.: Leurre.com: on the advantages of deploying a large scale distributed honeypot platform. In *In: ECCE 2005, E-Crime and Computer Conference*, pages 29–30, 2005.

[90] O. D. Project. DMOZ Open Directory Project. `http://www.dmoz.org/`, 2013.

[91] N. Provos. A virtual honeypot framework. In *Proceedings of the USENIX Security Symposium*, pages 1–14, 2004.

[92] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All Your iFrames Point to Us. In *Proceedings of the USENIX Security Symposium*, 2008.

[93] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The ghost in the browser analysis of web-based malware. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, pages 4–4, Berkeley, CA, USA, 2007. USENIX Association.

[94] N. Provos, M. A. Rajab, and P. Mavrommatis. Cybercrime 2.0: When the cloud turns dark. *Queue*, 7(2):46–47, Feb. 2009.

[95] QiQ Research, Inc. Data leak probe to PI industry. `http://www.japanpi.com/jp_investigations/pinews/data-leak-probe-concerning-pi-industry.html`, September 2012.

[96] J. Quinlan. C4.5: Programs for machine learning. In *Morgan Kaufmann Publishers*, 1993.

[97] D. Ramsbrock, R. Berthier, and M. Cukier. Profiling attacker behavior following ssh compromises. In *in Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2007.

[98] P. Ratanaworabhan, B. Livshits, B., and Zorn. Nozzle: a defense against heap-spraying code injection attacks. In *Proceedings of the USENIX Security Symposium*, 2009.

[99] K. Rieck, T. Krueger, and A. Dewald. CUJO: Efficient Detection and Prevention of Drive-by-Download Attacks. Technical Report TR-2010-10, Berlin Institute of Technology, 2010.

[100] L. Rist, S. Vetsch, M. Koßin, and M. Mauer. Glastopf. `http://honeynet.org/files/KYT-Glastopf-Final_v1.pdf`, november 2010.

[101] Robots IP Address Ranges. `http://chceme.info/ips/`.

[102] M. Roesch. Snort – Lightweight Intrusion Detection for Networks. In *Proceedings of LISA '99: 13th Systems Administration Conference*, November 1999.

[103] V. Roussev. Data fingerprinting with similarity digests. In K.-P. Chow and S. Shenoi, editors, *Advances in Digital Forensics VI*, volume 337 of *IFIP Advances in Information and Communication Technology*, pages 207–226. Springer Boston, 2010.

[104] A. Saebjornsen, J. Willcock, T. Panas, D. Quinlan, and Z. Su. Detecting code clones in binary executables. In *Proceedings of the eighteenth international symposium on Software testing and analysis*, ISSTA '09, pages 117–128. ACM, 2009.

[105] Google Safe Browsing API. `http://code.google.com/apis/safebrowsing/`, 2008.

[106] Security Ninja. Share prices and data breaches. `https://www.securityninja.co.uk/data-loss/share-prices-and-data-breaches/`, May 2011.

[107] C. Seifert and R. Steenson. Capture-HPC. `https://projects.honeynet.org/capture-hpc`, 2008.

[108] C. Seifert, I. Welch, and P. Komisarczuk. Identification of Malicious Web Pages Through Analysis of Underlying DNS and Web Server Relationships. In *Proceedings of the LCN Workshop on Network Security (WNS)*, 2008.

[109] C. Seifert, I. Welch, and P. Komisarczuk. Identification of Malicious Web Pages with Static Heuristics. In *Proceedings of the Australasian Telecommunication Networks and Applications Conference (ATNAC)*, 2008.

[110] S. Shin and G. Gu. Conficker and beyond: a large-scale empirical study. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 151–160, New York, NY, USA, 2010. ACM.

[111] R. Sommer and V. Paxson. Outside the Closed World: On Using Machine Learning For Network Intrusion Detection. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2010.

[112] IP Addresses of Search Engine Spiders. `http://www.iplists.com/`.

[113] S. Stigler. Fisher and the 5 *CHANCE*, 21(4):12–12, 2008.

[114] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2009.

[115] B. Stone-Gross, M. Cova, C. Kruegel, and G. Vigna. Peering through the iframe. In *INFOCOM, 2011 Proceedings IEEE*, pages 411 –415, april 2011.

[116] Symantec. 2013 Internet Security Threat Report. `http://www.symantec.com/security_response/publications/threatreport.jsp`, 2013.

[117] Symantec. Internet Security Threat Report 2013. `http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_appendices_v18_2012_221284438.en-us.pdf`, April 2013.

[118] Symantec. Norton Safe Web. `https://safeweb.norton.com/`, 2013.

[119] TBLOP - The Big List of Porn. `http://www.tblop.com/`, 2013.

[120] The Spamhaus Project Ltd. SpamHaus. `http://www.spamhaus.org/`.

[121] Torrent Sites. `http://www.torrentresource.com/`, 2013.

[122] S. J. Vaughan-Nichols. How the Syrian Electronic Army took out the New York Times and Twitter sites. `http://www.zdnet.com/how-the-syrian-electronic-army-took-out-the-new-york-times-and-twitter-sites-7000019989/`, August 2013.

[123] VirusTotal - Free Online Virus, Malware and URL Scanner. `https://www.virustotal.com/`.

[124] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites that Exploit Browser Vulnerabilities. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2006.

[125] webhosting.info. Country-wise top hosts. `http://www.webhosting.info/webhosts/tophosts/Country/`, 2012.

[126] Websense. 2013 Threat Report. `http://www.websense.com/assets/reports/websense-2013-threat-report.pdf`, 2013.

[127] Websense. Websense 2013 Threat Report. http://www.websense.com/content/websense-2013-threat-report.aspx?cmpid=prnr2.13.13, 2013.

[128] Wikipedia, the free encyclopedia. Anonymous (group). http://en.wikipedia.org/wiki/Anonymous_(group), September 2013.

[129] F. Wilcoxon. Individual comparisons by ranking methods. In S. Kotz and N. Johnson, editors, *Breakthroughs in Statistics*, Springer Series in Statistics, pages 196–202. Springer New York, 1992.

[130] G. Wondracek, T. Holz, C. Platzer, E. Kirda, and C. Kruegel. Is the internet for porn? An insight into the online adult industry. In *WEIS 2010, 9th Workshop on the Economics of Information Security, 7-8 June 2010, Boston, USA*, Boston, UNITED STATES, 06 2010.

[131] List of File Hosting and Sharing Websites. http://xboxpirate.eu/forums/topic/280-list-of-file-hosting-and-sharing-websites-137-entries/, 2013.

[132] H. Zhang. The Optimality of Naive Bayes. In *FLAIRS2004 conference*, 2004.

[133] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee. Arrow: Generating signatures to detect drive-by downloads. In *Proceedings of the International World Wide Web Conference (WWW)*, WWW '11, pages 187–196, New York, NY, USA, 2011. ACM.

[134] zx2c4. Linux Local Privilege Escalation via SUID /proc/pid/mem Write. http://blog.zx2c4.com/749, january 2012.