# PhishEye: Live Monitoring of Sandboxed Phishing Kits

Xiao Han
Orange Labs and Eurecom
xiao.han@orange.com

Nizar Kheir
Orange Labs
nizar.kheir@orange.com

Davide Balzarotti
Eurecom
davide.balzarotti@eurecom.fr

## ABSTRACT

Phishing is a form of online identity theft that deceives unaware users into disclosing their confidential information. While significant effort has been devoted to the mitigation of phishing attacks, much less is known about the entire life-cycle of these attacks in the wild, which constitutes, however, a main step toward devising comprehensive anti-phishing techniques. In this paper, we present a novel approach to sandbox live phishing kits that completely protects the privacy of victims. By using this technique, we perform a comprehensive real-world assessment of phishing attacks, their mechanisms, and the behavior of the criminals, their victims, and the security community involved in the process – based on data collected over a period of five months.

Our infrastructure allowed us to draw the first comprehensive picture of a phishing attack, from the time in which the attacker installs and tests the phishing pages on a compromised host, until the last interaction with real victims and with security researchers. Our study presents accurate measurements of the duration and effectiveness of this popular threat, and discusses many new and interesting aspects we observed by monitoring hundreds of phishing campaigns.

## 1. INTRODUCTION

Despite the large effort and the numerous solutions proposed by the security community, phishing attacks remain today one of the main threats on the Internet [1]. They usually aim at deceiving users into visiting fake web pages that mimic the graphic appearance of real and authentic websites [18]. The main goal of an attacker, also known as phisher, is to collect sensitive user data such as login credentials, banking information, or credit cards numbers. The stolen data can then be monetized by leveraging hijacked accounts and performing fraudulent online transactions, or indirectly through the resale of the stolen information to other cyber-criminals, mostly on the Internet black market [16].

Phishing attacks constitute a major challenge for Internet Service Providers (ISPs), as well as for email providers, browser vendors, registrars, cloud service providers, and law enforcement agencies. To mitigate these attacks, a broad set of solutions have been proposed, tackling each of the three different stages that constitute a phishing attack [16]. At the first stage, they try to prevent phishing emails from reaching the end users by applying email filters [12], or by detecting [41, 45, 35, 43, 28], blocking, or taking down phishing web sites [29]. At a second stage, existing solutions focus on providing better user interfaces, such as browser plugins, that inform users about the reputation of a target web site, and notify users as soon as they are redirected towards a potentially malicious page [42, 11]. Finally, the last line of defense relies on proper education to help users recognize phishing sites [22, 23]. Despite this considerable effort, the phishing problem is far from being solved and a recent report by the Anti-Phishing Working Group (APWG) shows that the number of unique phishing sites was still increasing in 2015 [3], and that the number of phishing reports the APWG receives has almost doubled between 2014 and 2015.

In order to discern phishing attacks, diverse efforts have been made by security researchers, and that involve different actors in the phishing ecosystem. However, previous studies focused mainly on the technical aspects of the problem, i.e., how phishers compromise vulnerable servers [40], and how phishing kits work [8]. Researchers also remotely analyzed existing phishing pages, based on ground truth datasets such as spamtraps [33] and phishing blacklists [29], in order to provide a real world assessment of the number of victims and to measure the efficiency and extent of take-down operations. Recently, Bursztein et al. [4] advanced our understanding about phishing by studying criminals incentives and the way they monetize stolen user credentials.

Unfortunately, previous studies have been confronted to two main dilemmas. First, most phishing kits were monitored only *after* they had been detected by public or private anti-phishing services. This drastically limits the extent of these studies, since an important part of the phishing life cycle (preceding any detection) has mostly remained unknown. Second, researchers have never observed the way real victims interact with phishing kits because of obvious ethical reasons. In this paper, we try to fill the gaps in the understanding of the phishing ecosystem by analyzing the attackers behavior and the way the potential victims interact with phishing kits in the wild.

As discussed in [32], *a natural tension exists between conducting accurate, reproducible research and reducing the harm caused by the content that is being removed.* Two are the main challenges that affect research on phishing attacks:



*CCS'16, October 24 - 28, 2016, Vienna, Austria*



ISBN 978-1-4503-4139-4/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2976749.2978330

Should researchers notify affected parties in order to expedite take-down of phishing sites? Should researchers intervene to assist victims?

Bearing these issues in mind, this paper proposes a new approach that lifts the barriers imposed by these ethical considerations in order to provide a first comprehensive real world assessment of phishing attacks, their mechanisms, and the behavior of all the actors involved in the process. Our approach leverages a web honeypot to attract real attackers into installing phishing kits in a compromised web application. This is inspired by the fact that, according to the APWG's Global Phishing Survey, 71.4% of the domains that hosted phishing pages were compromised domains [2]. We then present a novel sandbox technology designed to neutralize a phishing kit while maintaining it functional for a long period of time. Our approach is designed to strictly preserve the victim privacy, without interfering with the attack process in order to make sure that attackers can compromise the honeypot, install phishing kits, and conduct functional tests without being alerted about the sandbox configuration.

Preserving the user privacy is a very challenging task. Most phishing kits instantly exfiltrate the newly collected victim credentials to the attacker, leaving no time to remove these credentials from our servers. Even worse, almost 97% of malicious phishing pages are accessible via unprotected HTTP connections [2], which may expose the cleartext compromised credentials to eavesdropping over the network. Finally, as discussed in section 6, some phishing pages re-route the HTTP requests to a different server directly controlled by the attacker [40], using the compromised application only to host the page but not to collect the data.

Our sandbox proposes a comprehensive solution to these problems, allowing us to collect real world data about the behavior of both attackers and victims, and to perform the first thorough investigation regarding phishing attacks. To the best of our knowledge, no previous work was able to monitor, in a white-box fashion, the lifecycle of a phishing kit.

Based on these elements, we discuss a number of interesting findings. For example, our experiments show that phishing kits are only active for less than 10 days since their installation and over this time most of them collect a limited number of user credentials (fewer than reported in past experiments). Therefore, attackers rely on compromised websites to install a large number of phishing kits in a sort of shot-and-forget approach, rapidly moving to new phishing pages as the old ones get blacklisted. We also confirm that Google Safe Browsing (GSB) and Phishtank are very effective tools to protect end users. However, our experiments show that both services tend to blacklist phishing URLs between 10 and 20 days after their first appearance, and this is often too late as most of the victims already connected to the page. Finally, we observed a considerable *flash crowd* effect once phishing URLs appear in public blacklists. If not properly modeled, this phenomenon can completely skew the analysis results, by confusing security researchers with potential victims.

To summarize, we make the following contributions:

- We present a novel approach to sandbox live phishing kits that completely protects the privacy of end users.
- We observed the interaction of attackers, victims, and security researchers with the phishing pages, reconstructing for the first time the entire lifecycle of a phishing kit.
- For the first time, we measure the impact of blacklisting services on phishing pages from the time in which they are first installed (and not from the time they are reported or discovered by security companies). We also discuss new techniques to use the collected data to promptly identify the email address used by criminals to retrieve the stolen information.

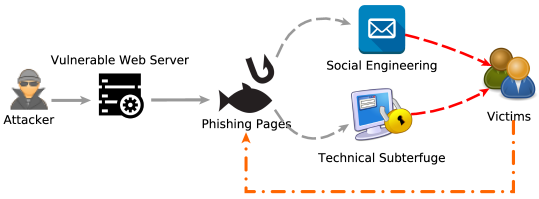



Figure 1: Typical Phishing Attack

Beyond these main findings, we also discovered two new phishing techniques that have never been reported before, and we conducted a thorough analysis of the modus-operandi of the corresponding campaigns.

## 2. BACKGROUND

In this section we provide a more detailed description of phishing attacks and their main actors. We then describe in more details the ethical issues we encountered during each phase of our experiments.

**Anatomy of Phishing Attacks:** A typical phishing attack, as depicted in Figure 1, consists of three main actors: a *phisher*, a set of potential *victims*, and possibly a number of *third party visitors* – such as researchers and security editors. Phishers mostly seek to compromise vulnerable web applications, install phishing kits that mimic victim web sites, and advertise the phishing URLs using, for example, spam emails and posts on social networks. The victims are the end users who receive these messages and connect to the phishing pages. Phishers usually seek limited interactions with their victims as their main goal is to hijack sensitive data without disclosing the real nature of their phishing pages. Therefore, they often redirect victims to the authentic website after they have provided their credentials, or they redirect them towards error pages to make them disconnect from the phishing site. As soon as a victim connects to a phishing page, she can either realize the real malicious nature of the site and disconnect, or she can be fooled by the legitimate-looking page appearance and attempt to login, thus providing her credentials. Finally, the third party category includes visitors from security companies, public crawlers, and independent researchers. They usually examine and monitor the phishing pages only after the presence of the phishing kit is included in popular blacklists (e.g., PhishTank), as discussed in details in section 6.2. The behavior of third party visitors can be very similar to the behavior of real victims, which makes it difficult to separate these two actors within the same experimental setup.

**Ethical Considerations:** Researchers have already proposed the use of honeypot systems as a tool to analyze phishing attacks [40]. However, conduct live phishing experiments

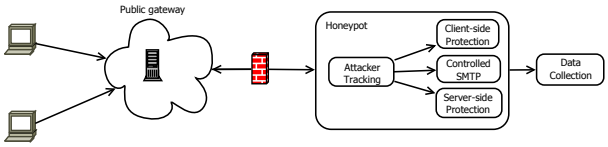



Figure 2: High Level System Overview

and evaluations inside honeypots has always been confronted with ethical considerations that have severely limited the scope of all previous research studies about phishing attacks. The result is that previous experiments were often limited to the analysis of how phishers compromise the honeypot, as well as the static analysis of the collected phishing kits. Unfortunately, honeypots have never been used to study the behavior of the victims as they connect and interact with the installed phishing pages. As soon as victims are being enrolled into a honeypot experimental setup, ethical considerations cover all aspects related to the protection and perfect secrecy of user identities, as well as the secrecy of their credentials in case where they may be exposed during the attack. In order to properly address this important ethical problem, one should have a better understanding about the key steps of a phishing attack where the user identity or credentials may be exposed. We divide these ethical issues into two main categories, depending on whether they may be addressed on the server or on the client-side.

On the server side, the phishing kits aim at collecting user submitted data. The stolen data may be either locally stored on the honeypot server until it is retrieved by the phisher, or it may be instantly sent to the phisher by email or direct HTTP connections. Therefore, it is important to prevent any user data to be locally stored on the server, or even processed by the phishing application. Moore et al. used public accessible data collected on vulnerable web servers, including statistics of the web page hits and credentials that the attackers had collected and forgotten to remove from the vulnerable server [29, 30]. Following the publication of their study, more ethical issues concerning the user credentials were exposed, most of which being still unanswered [32].

Apart from the above ethical problems on how to prevent information from being stored on the server, ethical issues also exist on the client-side. In fact, phishing kits may collect user credentials by posting them to other malicious servers that are under the direct control of the attacker. Moreover, 97% of the phishing pages are accessible via clear-text HTTP connections [2], thus exposing the victim's sensitive data to eavesdropping over the network. Therefore, and even though no sensitive data would be hosted on the remote server, there may still be a considerable risk because of the clear-text sharing of user credentials on the network.

Finally, it is important to note that our institution does not have an IRB, but we asked advice from the company legal department, and were granted permission to perform the research.

# 3. DATA COLLECTION

We leverage an existing honeypot infrastructure [5] as a basis to implement our system. As shown in Figure 2, our architecture consists of two main components: a public proxy gateway and a private backend server that implements the main honeypot applications. The gateway was hosted on a number of public hosting providers, including Amazon EC2, which previous research has revealed to be a popular target for attackers looking for machines to compromise and conduct malicious activities on the Internet [15]. The proxy server does not host any content and acts purely as a reverse proxy to forward all HTTP connections through a secured VPN channel towards the honeypot server hosted in our premises. For security reasons, outgoing traffic from the honeypot is dropped at the firewall – except for DNS queries and for a limited number of verified SMTPS connections (as explained in Section 4.3). On the honeypot server, we configured 18 vulnerable PHP pages that can be exploited by attackers and allow them to upload files and execute shell commands. Note that the honeypot is configured in such a way that attackers cannot modify these PHP pages.

The data collection module periodically retrieves the data collected on the honeypot server, including the server access logs and the uploaded files (such as web shells, phishing kits, defacement pages, exploit kits, and hacking tools).

**Elimination of Other Malicious Files:** The identification of phishing kits is performed through a number of heuristics, complemented by a manual classification. For instance, most phishing kits contain a large number of files and resources required to replicate the targeted website [8], and therefore isolated files are unlikely to be used for phishing. We also adopted a number of keywords to determine the content of the files and identify possible targets. However, it is very difficult (and outside the scope of this paper) to setup an accurate filter that can precisely separate phishing applications from other malicious files uploaded to the server (we refer the reader to existing work on this specific problem [45]). In our study we decided to adopt a conservative approach and we manually analyzed all the files that did not match our filters in order to be sure that *only* phishing pages were hosted on our honeypots. We removed on a daily basis other malicious files including web shells, exploit kits, and drive-by download from the honeypot. As part of this activity, we may have erroneously removed some small phishing kits, but we believe that it is better to be conservative and avoid exposing users to other dangerous threats.

**Data Exfiltration by Client-Side Side Channels:** We also verified that the phishing kits did not leverage other side-channels to capture and successfully exfiltrate the credentials from the victim browser. We found three PKs that had these functionality and used obfuscated JavaScript code disguised as a HTML `img` tag to send the user credentials directly to a remote server. However, our client-side protection described in Section 4 acts directly on the typed password, before the malicious JavaScript retrieves it.

Overall, through an accurate user inspection, we were able to confirm that no user credentials were disclosed during our experiments.

# 4. SANDBOX AND PK NEUTRALIZATION

This section presents an overview of our platform and describes the properties and design configuration that enable us to address the ethical issues outlined in section 2. It also illustrates the deployment scenario, including the system components and data management procedures. Finally, it describes the low-level implementation details and discusses the potential limitations of our approach.

## 4.1 Design Goals

In order to address the ethical issues and to enable a sanitized honeypot platform that preserves the privacy and security of any potential victim data, our system achieves the following design requirements:

**Client-Side Data Mangling:** To prevent sensitive data from being sent out of the user terminal, our system injects into all phishing pages a JavaScript component whose purpose is to replace any posted information with random data before it is sent over the network. The injected code protects all potential victims as long as they have not explicitly deactivated JavaScript in their browsers. For those users who may have deactivated JavaScript, our system also injects a HTML noscript tag that redirects user to an error page so that they would disconnect from the honeypot.

**Server-Side Data Randomization:** As an emergency backup in case JavaScript is enabled on the victim terminal but the injected code fails to replace the posted information, our honeypot server relies on a custom Apache PHP module that filters all incoming data before it reaches the phishing kit. In particular, our solution replaces on the fly all user data that reaches the server with random fake data, making sure that no sensitive information may ever reach the phishing kit, nor it can be locally stored on the honeypot.

**HTTP Redirection Disruption:** To make sure that our honeypot may not be used by the attacker as an elementary component of a broader malicious redirection chain [25], our system uses static analysis techniques to detect and disable any form of web redirection, and so to make sure that no users may be redirected from the honeypot towards any other malicious website under the control of the attacker.

**Concealed Instrumentation:** One of the main goals of our system is to protect the users while being perfectly transparent for the attacker. Therefore, the honeypot is designed to identify attackers and monitor their procedures and mechanisms without revealing the real nature of our experiment. In particular, we noticed that attackers usually test their kits by inserting fake credentials in the phishing page and verifying that such credentials are correctly exfiltrated to their preferred drop-zone. In order to ensure that these testing operations are successful, our system needs to selectively disable the client- and server-side randomizations, and to allow phishing kits to send emails (which is the most popular exfiltration technique implemented by the phishing kits [8]) when attackers decide to test their pages.

## 4.2 System Overview

As illustrated in figure 2, our honeypot implements five main functionalities, including *the attacker tracking module, the client-side protection module, the server-side protection module, the controlled SMTP module, and the data collection and processing module.*

As a core component of the system, the attacker tracking module is responsible for distinguishing attackers who connect to the honeypot from other benign users such as victims and third party visitors. This allows us to apply different access control policies whose purpose is on one hand to enable attackers to verify and test their phishing kits, and on the other hand to prevent any sensitive data posted by the victims from being captured by the attacker. This module assigns the role of the attacker to the user who installed the phishing kit on the compromised application. The attacker tracking module further keeps track of all attackers who connect to the honeypot using a history of attacker IP addresses and their associated user agents, and provides this information as input to the subsequent protection and data collection modules.

The client-side and server-side protection modules use the list of attackers provided by the tracking module as input in order to identify potential victims and to prevent their data from being exported or hijacked from the user terminal. If the visitor is not an attacker, the module inspects each outgoing HTTP response and injects in each page a HTML noscript tag and a static JavaScript file (more details in Section 4.3), which hooks the data submission and replaces the user's data with fake information.

The server-side protection module performs two functions: *backend data randomization* and *malicious redirection disruption.* In the first case, it acts as a second line of defense in addition to the client-side protection. It operates on the server side, and it replaces the incoming data with random values before it is passed to the web application. The server-side protection module also intercepts and blocks all HTTP, HTML, and JavaScript redirections that point to other remote websites in order to make sure that the honeypot server cannot be used as a stepping stone within a broader malicious redirection chain. It detects such redirections through performing static rule-based analysis over the content of each HTTP response provided by the server. Our system allows only redirections to other pages hosted in our honeypot, and automatically intercepts and drops any other destination.

Finally, as most phishing kits exfiltrate the stolen data by emails, and since our honeypot is configured to drop outgoing SMTP connections to prevent attackers to send spam, we also implemented an SMTP module that allows each attacker identified by the tracking module to send a configurable amount of messages (two in our experiments) to test a freshly-installed phishing kit.

## 4.3 Implementation

The experimental setup on the honeypot leverages two Apache HTTP modules, namely `mod_php` and `mod_security`. We configure and extend these two modules in order to incorporate our system functionalities. In the following, we describe in details our system implementation.

**mod_php** provides PHP support to the Apache HTTP server, and was extended to implement the attacker identification and server-side protection modules. In particular, we hook the PHP function `rfc1867_post_handler`, which is actually the form-based file upload handler, and we save the IP address of the attacker who has successfully uploaded a file in a privileged location. To implement our server-side protection module, we modify the functions `php_std_post_handler` and `php_default_treat_data`, which provide respectively the handler for HTTP POST and GET requests. These handlers whitelist the 18 pre-installed vulnerable pages to ensure that attackers can reach and use the honeypot. For requests toward other pages, the handlers perform the attacker verification and further replace the data with fake ones when the request is not originating from an attacker. The handlers identify specific types of information (such as login, email, and credit card fields) by using a pre-defined set of keywords and regular expressions, and overwrite these fields with randomly generated data that reproduces the same format. In

other words, emails are replaced with seemingly valid (but non-existent) email addresses, credit-card numbers by other fake numbers, and so on. If the system is unable to recognize the type of a field, its value is replaced with a random string of alphanumeric characters.

**mod_security** is a web application firewall that provides attack detection, traffic monitoring, logging and real-time analysis[1]. We extend the core functionalities of this module in order to implement the client-side data mangling, the malicious redirection disruption, and the data collection modules. The client-side data mangling behaves almost the same way as for the server-side data modification. The only difference is that it injects a HTML noscript tag and a static JavaScript code in the HTTP response in case the destination has not been identified by the attacker tracking module. The static JavaScript code, executed on the remote victim browser, modifies any data provided by the user before it is sent over the network. To achieve this, the injected JavaScript first replaces the native form submission function *submit()* with a custom handler that dispatches automatically a *submit* event on the page. Then it adds a *submit* event listener that hooks the form submission and modifies the submitted data. Note that the static JavaScript is prepended to the HTML page so that it is always executed before other JavaScript. During our experiment, we did not observe any PK that tried to detect this kind of hooking. If JavaScript is deactivated on the victim browser, the HTML noscript tag redirects the victim to an error page to prevent him from disclosing his credentials. Moreover, in order to prevent potential victims from being redirected towards other remote malicious destinations, we configure `mod_security` to drop all HTTP responses that contain redirections. Note that malicious redirections are only disabled for potential victims that cannot be identified by the attacker tracking module. The HTTP redirection can be detected by combining the response status code and the *Location* field in the HTTP header. To identify HTML and JavaScript redirections, our module parses the content of the response body and analyzes its HTML and JavaScript code. The last configuration option that we have added to the `mod_security` module is the ability to collect files uploaded to the honeypot. We configure the `SecUploadDir` and `SecUploadKeepFiles` functions in order to save a copy of the uploaded files into a protected location that is invisible to the attackers.

**Controlled SMTP:** The honeypot is configured to send all emails with a free email service provider through sSMTP[2]. To prevent attackers from sending spam emails from our honeypot, we limit to 2 the number of emails that each attacker can send. The original binary of sSMTP is replaced with a modified version that checks the presence of a specific flag in the arguments passed by the caller. Only the PHP mail handler is configured to call the sSMTP with the desired specific flag. Moreover, we modify the PHP mail handler so that it keeps only the first recipient when the destination contains multiple recipients.

**Experiment Limitations:** To mitigate denial of service attacks or botnet scans, we rate-limited to 5 the number of concurrent connections from the same IP address. We opted for this configuration because we detected multiple scan attempts to our honeypot using a common proxy server, and that most of these attempts did not lead to any attack. Moreover, we believe that it is highly unlikely to have more than five victims connecting simultaneously from the same IP address and it is equally unlikely to have more than five simultaneous attackers interacting with our honeypot using the same proxy IP address. Therefore, we believe that this rate-limiting has a marginal impact on our experiment.

Our strict attacker identification process would inevitably prevent attackers from submitting fake credentials to the site in case they use a combination of IP address and user agent that is different to the one used to upload the PK. However, we believe that this solution offers the best trade-off to achieve a complete protection of the user privacy. Nonetheless, based on the number and diversity of the collected kits and the potential victims, we believe that our experiment is sufficiently representative of existing phishing attacks in the wild, as discussed in section 5.4.

# 5. PHISHING ATTACK GLOBAL PICTURE

We collected 643 unique phishing kits, which were uploaded on our honeypot over a period of five months from September 2015 to the end of January 2016. Out of this initial dataset, 474 kits (74% of our initial dataset) have been correctly installed by 471 distinct attackers. The remaining kits were likely automatically uploaded by exploitation bots but never unpacked nor configured by the attackers. The installed phishing kits targeted 36 distinct organizations, mostly online banks, but also social networks and e-commerce portals. The five most frequent targets were Paypal (375), Apple (26), Google (10), Facebook (9) and the French online tax payment system (6).

This section presents an overview of our main findings, including the way attackers setup and operate their phishing attacks, the behavior of victims as they interact with the phishing kits, and our assessment for the lifetime of live phishing kits on the Internet. We initially focus on aggregated statistics and on understanding the big picture, and then we discuss the technical aspects related to how we identify victims, and how we separate them from attackers and third party visitors.

Figure 3 illustrates an aggregated timeline of all phishing attacks we observed in our study. On the graph, the Y-axis shows the major phases of an attack, and the X-axis indicates the time elapsed since the phishing kit was first uploaded to the honeypot. During our experiment, and as shown in Figure 3, we split the lifecycle of an attack in five separate phases: installation, testing, interaction with the first and last victims, and detection by popular blacklists.

The *installation* phase includes the actions performed by the phishers to unpack, install, and set up the phishing kit on the compromised machine. The *testing* phase describes how phishers test and verify the correct behavior of their newly installed kits. The *first* and *last victims* capture the time elapsed until we observe respectively the first and last connections from a victim. Lastly, the *detection* phase covers the time at which the phishing URL is added to public phishing blacklists (Google Safe Browsing and/or PhishTank in our experiment). All results are illustrated using a box plot that captures the distribution of the data spanning from the first quartile to the third quartile, with the red line showing the median value.

[1]https://www.modsecurity.org/
[2]http://linux.die.net/man/8/ssmtp

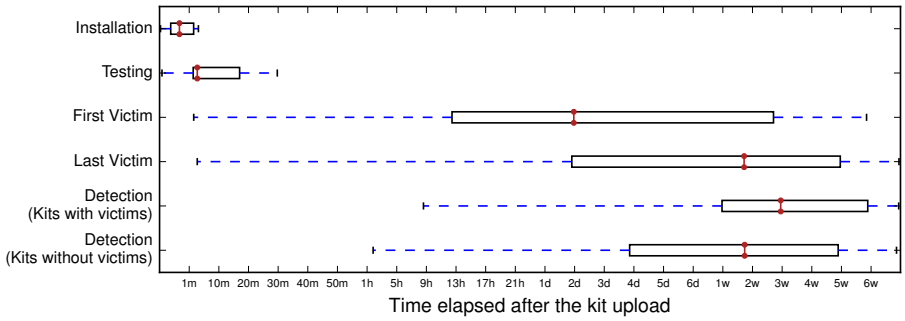



Figure 3: Phishing Attack Timeline

## 5.1 Attackers Behavior

Based on the referer field, 29% of attackers located our honeypot using search engine queries, mostly on Google (28%), and Yahoo (0.8%). Interestingly, over 40% of the attacks came with a Facebook-related referer, which seems to indicate that attackers are increasingly using social networks to share data (e.g., the location of web shells) or simply forge the referer field to make their traffic look more legitimate.

While most of the attackers installed only one copy of the phishing kit, some (26%) have installed their phishing kits on multiple sub-directories. After the installation was completed, 70% of the attackers visited the phishing pages and 58% of them submitted fake credentials to verify whether the kit was operating correctly. This confirms our initial hypothesis that a phishing sandbox needs to protect the victims but let the attacker freely play and test the installed pages, as described in Section 4. Without this feature, almost 60% of the uploaded kits would have been abandoned by the attackers.

As illustrated in Figure 3, phishers installed the uploaded kits very quickly on the compromised web server, just few minutes after the kit was uploaded. The testing phase was also performed immediately after the installation. However, the few minutes of delay and the irregular patterns suggest that this was performed manually and not through automated bots. Quite interestingly, after the testing phase was completed, we never observed any connection to the phishing pages from the same IP address.

In the second stage of a phishing attack, attackers would make use of standard social engineering techniques in order to drive potential victims into connecting to their phishing pages. During the entire period of our study, we observed only one attacker who has tried to use the compromised machine to also send the phishing emails (all messages were blocked by our firewall). This seems to suggest that attackers have decoupled the process of compromising public servers and installing phishing kits, from the process of sending phishing messages. A possible explanation is that using different infrastructures is more robust and decreases the probability that the compromised server is detected by security solutions.

## 5.2 Victims Behavior

To study the behavior of victims as they connect to the phishing pages, we first need to separate them in our dataset from other actors such as public crawlers and third party visitors. We leverage multiple heuristics and empirical observations that we describe as follows. First of all, users who were assigned the role *attacker* by the attacker tracking module do not belong to the victims category. We also discard public crawlers by looking at the user-agent header field in incoming HTTP requests – since it is reasonable to assume that a victim would not spoof its browser user-agent to mimic a search engine.

The most challenging part of our study was to separate victims from other third party visitors, such as researchers and security editors who may find the phishing URLs on public blacklists and connect to the honeypot to verify the content of the phishing pages. First of all, the source IP address may reveal valuable information that enabled us to classify users as either potential victims or third party visitors. For example, after verification in the whois database, we found that a large number of connections to our honeypot originated from university researchers (e.g. IP ranges belonging to Boston University, University of Pennsylvania, Carnegie Mellon, and Massachusetts Institute of Technology) and security companies such as Kaspersky, Symantec, Bluecoat, and Fortinet. To be conservative, we consider all these users as third party visitors (even though this can misclassify students who fell victim of the phishing attacks) and we remove them from the victims' category. This approach does not allow us to identify other third party visitors, such as independent researchers and curious individuals who found the URL on public blacklists or hacking forums. To identify this specific category of users, we leverage a number of additional heuristics, such as the fact that researchers may connect at regular intervals to verify whether the phishing pages are still available, or they spend a considerable amount of time investigating different sub-pages or other resources used by the phishing kit. Based on these observations, we filter out a large number of users from the victims category. We believe that our cleaning approach was very conservative and had potentially over-estimated the number of third-party researchers and reduced the number of victims. However, as we explain in more details later, in our experiments we noticed that a large amount of incoming HTTP requests were not from real victims and therefore would inflate the results if included in our analysis.

After our aggressive filtering, we counted a total number of 2,468 victims who have connected to 127 distinct phishing kits. Although the total number of victims seems to be relatively small compared to previous work that measured

the impact of phishing attacks, our study has the merit of providing the first fine-grained assessment of the number of victims for a phishing attack. In fact we addressed two main limitations that have contributed in the past to overestimate the number of victims for a phishing attack.

First of all, we observed a spike in the number of (third party) visitors just after a phishing page first appears on public phishing blacklists (details in section 6.2). While such crowd phenomenon is a natural consequence of blacklisting a phishing page, previous studies did not separate such connections from the set of real victims, which could have contributed to largely overestimating the real number of victims. Using our sandbox configuration we were able to identify and clearly separate this phenomenon, whose impact is further explained in details in Section 6.

Second, our honeypot configuration also enabled us for the first time to observe the victims submitting their credentials to the phishing page. This is made possible as we analyze the behavior of victims on a per-user basis, and so we can verify whether each victim has performed any HTTP POST request, which suggests that the user has submitted data to the phishing page. Over the period of our study, we found that 215 users (9% of the total) have indeed posted their credentials to the phishing page. Note that since our system replaces automatically the submitted data with fake values, we are however unable to estimate the number of victims who send fake credentials to the phishing page.

Finally, the geolocation of the victims did not provide any particular insight, except for confirming that PKs are often targeted to a particular audience (e.g., French citizens for the French tax payment system), and in fact many PKs received the majority of their victims from a single country.

## 5.3 PK Lifetime

We measure the effective lifetime of a phishing kit as an indicator of the time interval during which a phishing kit remains operational on the Internet. We consider a phishing kit to be operational as long as new victims connect to the phishing page. Note that even though the phishing kits are kept online on the honeypot server throughout the duration of our study, we did not observe new victims after the phishing URLs were blacklisted by a large enough number of anti-phishing services and browser plugins.

A main challenge when measuring the effective lifetime of a phishing kit is that it would be very difficult to capture the exact time for the last victim who connect to the phishing page. In particular, we may still observe few users connecting to a phishing kit days after the phishing kit has been abandoned by the attacker, thus causing us to overestimate the lifetime of a phishing kit. To address this challenge, and so to eliminate such outlier observations, we cut the tails of the distribution and only consider the time interval during which we observed 90% of victims connections. As shown in Figure 3, the *first victim* captures the time elapsed until we observed 5% of victims connecting to a given phishing page, and the *last victim* captures the time elapsed until we have observed 95% of victims. Using this definition, we observed during our study that the first victims connect to a phishing page in average two days after the page was installed by the attacker. The last victims (at the 95% threshold) connect in average to the phishing page after 10 days. This gives us an estimated lifetime of eight days.

## 5.4 Effectiveness of Phishing Blacklist

To evaluate the reaction of the security community against phishing attacks, we leverage two anti-phishing services: GSB and PhishTank. We chose these two actors first because GSB is integrated by default into Chrome and Firefox, which together account for 87.1% of the market of Internet browsers[3], and second because PhishTank has been extensively used as a source feed in many previous research studies [44, 29, 26]. Our main goal in this section is to measure the time it takes for an anti-phishing service to blacklist phishing URLs since the corresponding kits were first uploaded to the honeypot. To do so, and throughout the duration of our experiment, we periodically check, multiple times per day, both blacklists for all phishing URLs that were installed in our honeypot.

While almost all phishing pages that were installed on the honeypot (98% of them) were correctly detected and blacklisted by the two phishing blacklists, we observed an average detection latency of 12 days. More precisely, we split this value in two separate categories: phishing kits for which we observed victims were blacklisted in average 20 days after installation, while kits with no victims were blacklisted in average 10 days after their installation. As illustrated in Figure 3, the detection latency varies quite a lot depending on the phishing pages and the evasion techniques used by the attackers, as further discussed in Section 6.2. More importantly, 62% of the phishing kits were blacklisted only after 75% of victims had already connected to the corresponding phishing page. While these observations seem to indicate that the anti-phishing services were not effective against a certain category of phishing attacks, they were indeed very proactive against another 27% of the phishing kits during our study, as they blacklisted these URLs even before 25% of victims have yet connected to the phishing page.

Another interesting aspect is the fact that GSB initially blacklisted only individual phishing kits. However, after many phishing pages had been reported for the same domain name, GSB changed its policy and started blacklisting entire directories subtrees in those domains. This approach may have proactively blacklisted other kits that were installed within these same directories.

## 5.5 Measurement Bias

Clayton et al. [7] draw the attention to the potential measurement bias in this type of studies. For instance, PhishTank only contains 40% of all phishing URLs, but 100% of all PayPal phish. Therefore, if the composition of the PhishTank dataset is not understood, the focus on attacking PayPal will be overestimated. In our study, we evaluate the effectiveness of the PhishTank dataset against the 471 phishing kits installed in our honeypot. Even though an important amount (375, i.e., about 80%) of these PKs involved PayPal, only a small portion (about 1%) of them have been reported and thus included in the PhishTank dataset. This in return corroborates to some extent the diversity and the representativeness of our dataset. We thus argue that our measurement is relatively representative.

Another concern may be related to the large number of PKs that received no victims. It is hard to know the real reason, but we do not believe that the fact that attackers could have recognized the honeypot is the main explanation

[3]http://www.w3schools.com/browsers/browsers_stats.asp

| Drop Techniques | Live Kits |
|---|---|
| Email | 443 |
| File | 30 |
| POST | 2 |
| MAILTO | 1 |

Table 1: Drop mechanisms of the live phishing kits

behind this phenomenon (even though in some cases it could certainly be the case).

In 34% of the PKs that did not receive any victims, the system did not receive any follow-up connection after the PK was uploaded. These PKs are mostly uploaded by automated exploitation bots without any human activity. In this case, it is possible that the attackers had many available targets that were compromised by their bot, and simply did not choose to use our system. In the remaining 66% of the kits with no victims, the attacker connected to the PK and explicitly tested it by submitting some credentials. Our logs did not report anything anomalous and the PK correctly sent the email with the testing credentials to the attacker, but then received no victims. Even though we do not know the reason, it is possible that no victims clicked on the phishing link, or that the phishing emails were largely stopped by antispam solutions. Overall, since we observed that PKs are often fire-and-forget pages uploaded in large numbers and used only for few days, it can be normal that some of them are never used and some have zero-success rate.

# 6. CASE STUDIES

In this section we provide details about four interesting cases we observed in our experiments including a new phishing kit dropping technique, a blacklist evasion technique, the time distribution of victims as they connected to the phishing pages, and we discuss a possible way to use our system to detect drop email addresses from live phishing kits.

## 6.1 Dropping Techniques

Most of the phishing kits contain the complete phishing web sites in a ready-to-deploy package. One of the main function of these kits is to automatically send the collected information to the attackers. In order to assess the technical evolution of the phishing kits since a previous study [8] conducted in 2008, we analyze the mechanisms that the phishing kits use to exfiltrate information.

As illustrated in Table 1, the vast majority of kits use email accounts to send data to the attackers. Few kits save directly the collected information on the server, and only two send it to a remote server using a HTTP POST request. Interestingly, we observed during our study that attackers experimented a new drop technique that aimed at sending the information in a HTML form through an email directly from the end user terminal. In order for this to work, attackers configured the HTML form action to `mailto` the values directly to their email address. Even though this technique is already well documented[4], this is the first time that phishers were observed to implement it inside a phishing kit. However, note that this method is very unreliable and only works with a particular combination of Internet Explorer and Outlook Express.

[4]http://www.html-form-guide.com/email-form/email-form-mailto.html

To further discover new techniques adopted by phishers, we analyze all the external links included in the kits uploaded to our honeypot. We found that 10 kits made use of resources directly fetched from the content distribution network (CDN) of the target organization. More interestingly, we also found that another 98 kits contained code to contact other computers likely under control of the phishers. These were either carefully disguised backdoors to exfiltrate the stolen credentials on a second channel (which was blocked by our firewall), or ways to retrieve information provided by the attackers, typically blacklists of endpoints or user-agents to block from viewing the page.

## 6.2 Blacklist Evasion

During our experiments we experienced an unexpected service outage caused by disk space exhaustion. Further investigation revealed that the exhaustion was caused by a phishing kit whose entry page, namely `index.php`, automatically creates a random subdirectory, copies the content of the entire kit inside it, and then redirects the visitor to the newly generated random location. The following PHP code presents an example of such behavior:

```
$random=rand(0,100000000000);
$md5=md5("$random");
$base=base64_encode($md5);
$dst=md5("$base");
$src="New Folder";
recurse_copy( $src, $dst );
header("location:$dst");
```

After the PK was installed, the phisher visited the entry page, which generated a copy of the kit at a random location. Interestingly, the phisher could distribute either the newly generated link to conceal the original phishing page location, or the link to the entry page so that each visitor would be automatically redirected to a different location. To understand the intended function of this phishing kit, we leverage the server access log to observe the first victims of this kit. We found that the phisher conducted a phishing campaign with a link to the entry page of the phishing kit, as the first victims landed directly to the entry page. This approach may seem controversial at a first glance, as the phishers exposed the real link to the phishing kit instead of hiding it from being blacklisted. In order to understand the reason behind this, we examine how PhishTank and Google Safe Browsing react to such phishing kit.

**PhishTank** publishes phishing reports submitted by different users, which allows us to retrieve the reported links and further compare them with the original link to the entry page. Unfortunately, most of the users had been fooled by this technique, and reported the randomly generated locations instead of the entry page, as illustrated by the following anonymized web server access log:

```
[12/Nov/2015:18:57:41] 14.xx.xxx.198
GET /kit/ 302
User-Agent: curl/7.25.0
[12/Nov/2015:19:01:35] 213.xx.xxx.100
GET /kit/8c5fcf4518e94a9f272d60ee75c309a7 301
User-Agent: Mozilla/4.0
[12/Nov/2015:19:20:45] 213.xx.xxx.100
GET /kit/8c5fcf4518e94a9f272d60ee75c309a7/redirection.php 200
User-Agent: Mozilla/4.0
```

The user first leveraged a command line tool (curl) to visit the link distributed by the phisher, which referenced the entry page of the phishing kit. As a consequence, a new phishing link has been generated, which however has not yet been

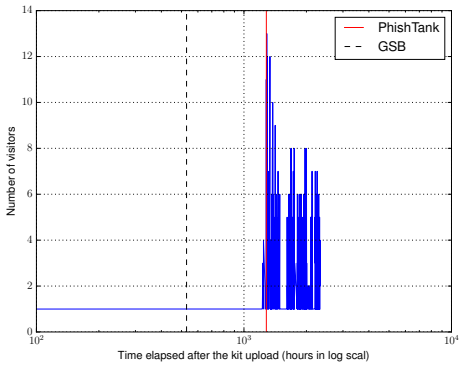



Figure 4: Visitor time distribution of the kit with blacklist evasion technique

visited until the reporter verified it a few minutes later with an IP address different to his first connection. The reporter was then redirected to the phishing page under a random location, which was also the link reported to PhishTank.

This PK confirms the existence of the crowd effect described in the previous section. In fact, while real victims would visit different randomly generated URLs, we observed hundreds of incoming connections toward the same link blacklisted by PhishTank. In Figure 4, the X-axis presents the number of hours in log scale that have elapsed after the kit was uploaded to the honeypot. The Y-axis describes the number of visitors that have connected to the phishing kit. The vertical lines correspond to the time upon which Google Safe Browsing or PhishTank has detected the given phishing kit. During the first few days the phisher was able to attract few victims, who connected directly to the entry page. After the random link was published by PhishTank we received many connections from researchers and security companies, which connected to the reported link instead of the entry page of the phishing kit. However, our technique was able to correctly remove these visits from the victim set.

**Google Safe Browsing** publishes only the MD5 digest of the phishing links, which are represented as host-suffix/path-prefix expressions. These expressions can match arbitrary URLs as long as they have the required host suffix and path prefix. This approach helps to protect against sites where the attacker uses randomly generated sub-paths to evade blacklists[5].

## 6.3 Victim Time Distribution

To illustrate and further confirm our measurement of the effective lifetime of phishing kits discussed in Section 5.3, we plot the victim time distribution from four of the most significant phishing kits, as presented by Figure 5.

In general, we observe two different distribution shapes: (1) a skewed right distribution where the majority of victims connected to the phishing kit within a short period of time after the kit was uploaded on the honeypot and (2) a bimodal distribution where two groups of victims connected to the phishing kit during two different periods. Kits 1-3, as presented in Figures 5a, 5b and 5c belong to the first category, having a single group of victims within a short period, followed by a long tail of very few victims. The phishers of the three kits performed probably only one phishing campaign after which they abandoned these kits. Different to the previous kits, Kit 4 in Figure 5d belongs to the second category, with two distinct groups of victims. The first group connected to the phishing kit within 24 hours, while the second burst arrived after exactly 70 hours (about 3 days). This is probably a consequence of two different phishing campaigns.

As shown in Figure 5, even if phishing kits remain online for a long period of time, they are only active for a short duration after the kit is uploaded to a compromised server. Besides, this effective lifetime is largely unaffected by the time at which Google Safe Browsing (GSB) or PhishTank blacklist the phishing pages, since these services usually become effective only after most victims have already visited the phishing pages. This victim time distribution further supports the way we measure the effective lifetime of the phishing kits described in Section 5.3.

## 6.4 Real-time Email Detection

Our results show that GSB and PhishTank are not fast enough to blacklist new phishing kits, which leaves victims on their own to identify and protect against phishing attacks. We believe that honeypot systems, as the one described in this paper, could be deployed to provide an early detection mechanism to promptly identify drop email addresses used by the attackers. These addresses can be further used by the email service providers to disable the email accounts to prevent phishers from retrieving the stolen credentials.

Our custom PHP interpreter records constantly the attempts to send an email even when the attacker makes use of the error control operators (@)[6] to silence all error messages. Each attempt is logged along with the file path of the script that tried to send an email and the destination address. Our system can detect the drop email as soon as an email is sent to the phisher, which may be either triggered by the phisher who verifies the good behavior of the phishing kit or by the first victim who submits some data.

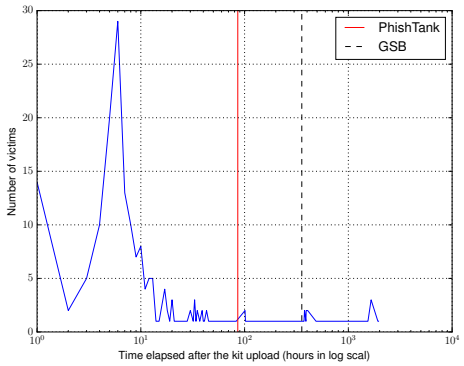
In order to assess the effectiveness of this approach, we manually check the emails sent to the attackers to estimate how many attacker email accounts could have been detected during our study. In total, we have found 68 distinct drop email addresses in the sent box – so each email address has been used in average by two different phishing kits. Unfortunately, only four of these addresses were disabled or were unreachable at the moment when the phishing kits attempted to send the first stolen credentials. This shows that email providers already disable drop email accounts. However, this process can greatly benefit from using a more systematic infrastructure (like the one presented in this paper) to collect more addresses in an efficient manner.

# 7. RELATED WORK

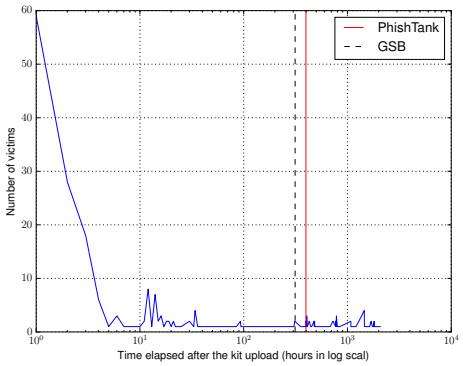
The literature includes a large number of papers related to phishing attacks. We classify these papers into the following three categories: anatomy of phishing, anti-phishing techniques, and evaluation of anti-phishing techniques.

[5]https://code.google.com/p/google-safe-browsing/wiki/SafeBrowsingDesign

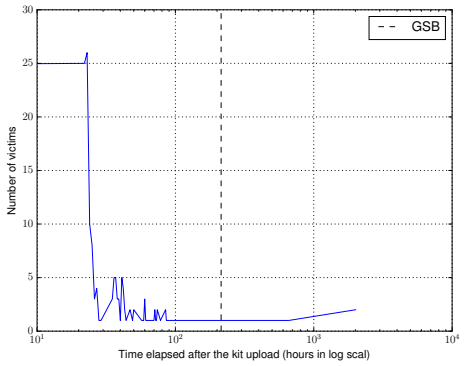
[6]http://php.net/manual/en/language.operators.errorcontrol.php



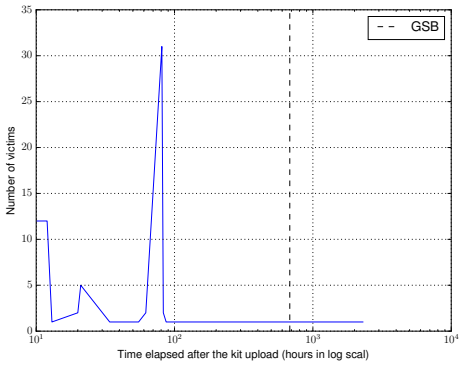

(a) Kit 1 that is detected by GSB and PhishTank




(b) Kit 2 that is detected by GSB and PhishTank




(c) Kit 3 that is detected by GSB




(d) Kit 4 that is detected by GSB

Figure 5: Victims time distribution for the most significant phishing kits

### *Anatomy of Phishing*

The work most closely related to our study is from Waston et al., who described two phishing incidents [40] that were discovered by the Honeynet Project [38]. Authors describe how phishers behave and the techniques used to set up the phishing sites. One of the two phishing kits received 256 inbound HTTP requests, but apparently no personal data was submitted by the visitors. Yet, authors had to shut down the honeypot because they did not have any system that can avoid user data from being stolen. Our work adopts a similar honeypot-based approach but focuses on providing an ethical system to study how real-world phishing attacks are structured. McGrath et al. [27] analyze the modus operandi of phishers, the characteristics of phishing URLs, the domains, and their hosting infrastructure. The authors also estimate the lifetime of phishing domain names by using periodically collected DNS records. Moore et al. [30] present the evidence that miscreants use search engine ("Google Hacking") to compromise and re-compromise machines, which are further used to host phishing sites. In another work, Moore et al. [33] studied the temporal correlations between spam and phishing websites in order to understand the attackers behavior, and to evaluate the effectiveness of phishing site take-down. Sheng et al. [36] conducted a demographic analysis of victims' susceptibility to phishing attacks and discussed the effectiveness of educational materials.

A few studies have focused on estimating the success rate of phishing emails. Jagatic et al. first report a baseline success rate for individual phishing emails [17], while Jakobsson et al. propose ethical phishing experiments on a popular online auction website [19], in order to measure the success rate of simulated phishing emails. The reported success rate is respectively about 15% and 11% (compared to 9% we found in our study). However, these works measured the success rate based only on *simulated* phishing attacks.

Multiple studies measured the impact of phishing on potential victims. Moore et al. have empirically measured the lifetime of phishing sites and the number of user responses [29]. Authors retrieved confirmed reports from PhishTank, and then relied on the records generated by Webalizer, a free web server log analysis tool, in case where it was already installed on the compromised websites. However, this tool saves merely the number of hits for a given web page instead of the unique number of visits, which makes the reported results a very rough estimation. The authors were also able to estimate the number of victims for 20 phishing sites, based on the assumption that only users who fall victim of a phishing would be redirected to a confirmation page after they have provided their credentials. Trusteer measured the effectiveness of phishing attacks based on the statistics gathered by a browser plugin over a period of three months [39]. The authors found that in 2009, 45% of bank customers who were redirected to a phishing site divulged

their personal credentials. However, their study provides only a partial view of a phishing attack.

Cova et al. in [8] studied the phishing kits distributed for free and those obtained by crawling live phishing sites. They analyzed the target organizations, the techniques used to exfiltrate data, and the obfuscation methods implemented in the phishing kits.

### *Anti-Phishing Techniques*

Phishing countermeasures have attracted a lot of interest from the research community. The proposed countermeasures can be grouped into three categories: phishing page detection, blocking, and user training.

Most phishing detection techniques identify phishing pages by building a classifier using different heuristics based on URL features [13, 24] or on the web page content [35, 45, 41]. Some studies aim at detecting and blocking phishing attacks at different stages. For instance, Fette et al. [12] use machine learning to identify and block phishing emails. Several studies propose a browser plugin to protect users from phishing [6, 9, 20]. Another popular approach to block phishing is to compile and distribute blacklists, such as Google Safe Browsing and PhishTank. A number of studies have focused on education against phishing attacks and how to train users to identify phishing [22, 21, 23, 34]. Finally, only one study focused on identifying drop email addresses through leveraging the list of known phishing websites from PhishTank and metadata maintained by email providers [31]. However, this method introduces a significant latency compared to our approach since public blacklists may not be efficient enough to promptly detect live phishing kits.

### *Evaluation of Anti-Phishing Techniques*

In 2006, a number of studies concluded that anti-phishing solutions [44], security indicators [10], and browsers toolbars [42] were ineffective in detecting phishing sites and protecting users.

In 2007, Ludl et al. [26] and Sheng et al. [37] specifically focused on the effectiveness of blacklists to prevent phishing, reaching different results. In the first study, the authors collected online phishing URLs from PhishTank, and tested them against Google Safe Browsing along with the Phish Filter of Microsoft Internet Explorer. This study concluded that the blacklist approach is efficient in protecting users, especially Google which correctly recognized almost 90% of the malicious URLs [26]. In the second study, Sheng et al. evaluated five blacklists (including the ones tested by Ludl et al.) with phishing URLs less than 30 minutes old, collected from the University of Alabama Phishing Team's email data repository. The paper found that those blacklists were ineffective as most of them caught less than 20% of phishing pages at hour zero [37].

Egelman et al. conducted an empirical study to evaluate the effectiveness of web browser phishing warning and provided some recommendations to improve security indicators [11]. In 2014, Gupta et al. [14] measured the effectiveness of educating pages [21] to see if they helped users not to fall for phishing.

Most existing studies evaluated the effectiveness of anti-phishing techniques only *after* the phishing page was reported publicly or privately. Our study assesses instead the effectiveness of the blacklist approach right from the beginning of the phishing attacks.

## 8. CONCLUSIONS

In this paper we present the design and implementation of a honeypot system especially designed to analyze and disarm phishing kits. Using this infrastructure, we conducted a five-month experiment to understand and measure the entire life cycle of this type of attack.

Different from previous works, our approach is able to measure the effective lifetime of phishing kits, which starts immediately after the kit installation. We are also the first to clearly distinguish the victims from the attackers and the other third party visitors. Our results show that less victims divulge their credentials compared to previous studies conducted in 2009 [39], maybe due to an increased user education in the past seven years against this threat.

## 9. ACKNOWLEDGMENTS

We would like to thank the reviewers for their valuable comments that allowed us to improve the quality of this paper. This research was partly funded by the French Ministry of education and research under Cifre grant given to Xiao Han, and by the European Union's Horizon 2020 project SUPERCLOUD under grant agreement 643964.

## 10. REFERENCES

[1] Kaspersky: Top 7 Cyberthreats to Watch Out for in 2015-2016. http://usa.kaspersky.com/internet-security-center/threats/top-7-cyberthreats.

[2] G. Aaron and R. Manning. APWG global phishing report 2014. http://apwg.org/download/document/245/APWG_Global_Phishing_Report_2H_2014.pdf.

[3] G. Aaron and R. Manning. APWG phishing activity trends report 2015. https://docs.apwg.org/reports/apwg_trends_report_q1-q3_2015.pdf.

[4] E. Bursztein, B. Benko, D. Margolis, T. Pietraszek, A. Archer, A. Aquino, A. Pitsillidis, and S. Savage. Handcrafted fraud and extortion: Manual account hijacking in the wild. In *Internet Measurement Conference (IMC)*, 2014.

[5] D. Canali and D. Balzarotti. Behind the scenes of online attacks: an analysis of exploitation behaviors on the web. In *Annual Network and Distributed System Security Symposium (NDSS)*, 2013.

[6] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell. Client-side defense against web-based identity theft. In *Annual Network and Distributed System Security Symposium (NDSS)*, 2004.

[7] R. Clayton, T. Moore, and N. Christin. Concentrating correctly on cybercrime concentration. In *Workshop on the Economics of Information Security*, 2015.

[8] M. Cova, C. Kruegel, and G. Vigna. There is no free phish: An analysis of "free" and live phishing kits. In *Workshop on Offensive Technologies (WOOT)*, 2008.

[9] R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic security skins. In *Symposium on Usable Privacy and Security*, 2005.

[10] R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *SIGCHI conference on Human Factors in computing systems*, 2006.

[11] S. Egelman, L. F. Cranor, and J. Hong. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In *SIGCHI Conference on Human Factors in Computing Systems*, 2008.

[12] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In *World Wide Web (WWW) Conference*, 2007.

[13] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *Workshop on Recurring malcode*, 2007.

[14] S. Gupta and P. Kumaraguru. Emerging phishing trends and effectiveness of the anti-phishing landing page. In *Electronic Crime Research (eCrime)*, 2014.

[15] X. Han, N. Kheir, and D. Balzarotti. The role of cloud services in malicious software: Trends and insights. In *Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2015.

[16] J. Hong. The state of phishing attacks. *Communications of the ACM*, 2012.

[17] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Communications of the ACM*, 2007.

[18] M. Jakobsson and S. Myers. Phishing and countermeasures: understanding the increasing problem of electronic identity theft. 2006.

[19] M. Jakobsson and J. Ratkiewicz. Designing ethical phishing experiments: a study of (rot13) ronl query features. In *World Wide Web (WWW) Conference*, 2006.

[20] Y. Joshi, S. Saklikar, D. Das, and S. Saha. Phishguard: a browser plug-in for protection from phishing. In *Internet Multimedia Services Architecture and Applications (IMSAA)*, 2008.

[21] P. Kumaraguru, L. F. Cranor, and L. Mather. Anti-phishing landing page: Turning a 404 into a teachable moment for end users. In *Conference on Email and Anti-Spam (CEAS)*, 2009.

[22] P. Kumaraguru, Y. Rhee, S. Sheng, S. Hasan, A. Acquisti, L. F. Cranor, and J. Hong. Getting users to pay attention to anti-phishing education: evaluation of retention and transfer. In *Anti-phishing working groups annual eCrime researchers summit*, 2007.

[23] P. Kumaraguru, S. Sheng, A. Acquisti, L. F. Cranor, and J. Hong. Teaching johnny not to fall for phish. *ACM Transactions on Internet Technology (TOIT)*, 2010.

[24] A. Le, A. Markopoulou, and M. Faloutsos. Phishdef: Url names say it all. In *Conference on Computer Communications (INFOCOM)*, 2011.

[25] Z. Li, S. Alrwais, Y. Xie, F. Yu, and X. Wang. Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures. In *Security and Privacy (S&P)*, 2013.

[26] C. Ludl, S. McAllister, E. Kirda, and C. Kruegel. On the effectiveness of techniques to detect phishing sites. In *Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2007.

[27] D. K. McGrath and M. Gupta. Behind phishing: An examination of phisher modi operandi. In *Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[28] E. Medvet, E. Kirda, and C. Kruegel. Visual-similarity-based phishing detection. In *Security and Privacy in Communication Netowrks Conference*, 2008.

[29] T. Moore and R. Clayton. Examining the impact of website take-down on phishing. In *Anti-phishing working groups annual eCrime researchers summit*, 2007.

[30] T. Moore and R. Clayton. Evil searching: Compromise and recompromise of internet hosts for phishing. In *Financial Cryptography and Data Security*. 2009.

[31] T. Moore and R. Clayton. Discovering phishing dropboxes using email metadata. In *eCrime Researchers Summit (eCrime)*, 2012.

[32] T. Moore and R. Clayton. Ethical dilemmas in take-down research. In *Financial Cryptography and Data Security*. 2012.

[33] T. Moore, R. Clayton, and H. Stern. Temporal correlations between spam and phishing websites. In *Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2009.

[34] K. Onarlioglu, U. O. Yilmaz, D. Balzarotti, and E. Kirda. Insights into user behavior in dealing with internet attacks. In *Annual Network and Distributed System Security Symposium (NDSS)*, 2012.

[35] Y. Pan and X. Ding. Anomaly based web phishing page detection. In *Annual Computer Security Applications Conference (ACSAC)*, 2006.

[36] S. Sheng, M. Holbrook, P. Kumaraguru, L. F. Cranor, and J. Downs. Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In *SIGCHI Conference on Human Factors in Computing Systems*, 2010.

[37] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang. An empirical analysis of phishing blacklists. In *Conference on Email and Anti-Spam (CEAS)*, 2009.

[38] L. Spitzner. The honeynet project: Trapping the hackers. *Security and Privacy (S&P)*, 2003.

[39] Trusteer. Measuring the Effectiveness of In-the-Wild Phishing Attacks. https://web.archive.org/web/20120324061250/http://www.trusteer.com/sites/default/files/Phishing-Statistics-Dec-2009-FIN.pdf.

[40] D. Watson, T. Holz, and S. Mueller. Know your enemy: Phishing. https://www.honeynet.org/papers/phishing, 2005.

[41] C. Whittaker, B. Ryner, and M. Nazif. Large-scale automatic classification of phishing pages. In *Annual Network and Distributed System Security Symposium (NDSS)*, 2010.

[42] M. Wu, R. C. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *SIGCHI conference on Human Factors in computing systems*, 2006.

[43] G. Xiang and J. I. Hong. A hybrid phish detection approach by identity discovery and keywords retrieval. In *World Wide Web (WWW) conference*, 2009.

[44] Y. Zhang, S. Egelman, L. Cranor, and J. Hong. Phinding phish: Evaluating anti-phishing tools. In *Annual Network and Distributed System Security Symposium (NDSS)*, 2007.

[45] Y. Zhang, J. I. Hong, and L. F. Cranor. Cantina: a content-based approach to detecting phishing web sites. In *World Wide Web (WWW) Conference*, 2007.