PhishLive: A View of Phishing and Malware Attacks from an Edge Router

Lianjie Cao[1], Thibaut Probst[2], and Ramana Kompella[1]

[1] Purdue University, West Lafayette, Indiana, USA
[2] INSA de Toulouse, Toulouse, France

Abstract. Malicious website attacks including phishing, malware, and drive-by downloads have become a huge security threat to today's Internet. Various studies have been focused on approaches to prevent users from being attacked by malicious websites. However, there exist few studies that focus on the prevalence and temporal characteristics of such attack traffic. In this paper, we developed the PhishLive system to study the behavior of malicious website attacks on users and hosts of the campus network of a large University by monitoring the HTTP connections for malicious accesses. During our experiment of one month, we analyzed over 1 billion URLs. Our analysis reveals several interesting findings.

1 Introduction

The rapid development of the Web over the recent few decades has made the Internet a hotbed for a wide range of criminal activities. Numerous types of attacks are hidden behind HTTP connections such as phishing, cross-site scripting, malware, and botnet attacks. The most commonly used solution to defend against such attacks is using blacklisting. A blacklist-based defense system contains a set of URLs that are identified as malicious or suspicious, either through a human-vetting process or other mechanisms. When users are trying to connect to such web pages, the browsers (e.g., Mozilla Firefox) pop out warnings or block the web page directly.

Literature is ripe with several studies that focused on documenting the effectiveness of such browser-based techniques in thwarting malicious website attacks. For example, [1] discusses the effectiveness of passive and active warnings to users. Similarly, [2] studies the efficacy of different anti-phishing tools. There also exist several papers (e.g., [3–6]) proposing different solutions for improving the attack detection and defense using enhanced blacklisting techniques. Other content-based techniques have also been proposed (e.g., [7–12]) for detecting malicious webpages, while [13] combines both URL-based and content-based methods.

Unfortunately, to date, there exists only a few studies that focus on understanding temporal characteristics of phishing or malware accesses in an edge network such as a campus or an enterprise network comprising of a few tens of thousand users. [14] analyzes the malware serving infrastructure of drive-by downloads. The paper indicates that the malware serving networks are composed of tree-like structure and malware are delivered through several redirections. However, they focus only on drive-by downloads ignoring other malicious attacks delivered through webpages and the data set

M. Roughan and R. Chang (Eds.) PAM 2013, LNCS 7799, pp. 239–249, 2013.

they use is collected generally not from a specific network. [15] assesses the issue of overt manifestation of three networks consisting of around 30,000 users in total. They study risky behaviors of users including security threats such as scanning, spamming, payload signature and botnet rendezvous. Nevertheless, they only study the probability of triggering malicious activities of users without concluding any possible pattern of vulnerable users and attackers.

There are however several questions that remain to be answered such as whether malicious sites are accessed just once, or a few times, or are repeatedly accessed over time across users, and whether other malicious website attacks are hidden between HTTP redirects. We believe a study to answer such questions is important for many reasons: First, it can help in sizing the resource requirements of security middleboxes that can be deployed to defend against them. Second, the temporal characteristics can help generate insights to inform future defense mechanisms. Thus, in this paper, we focus on studying and understanding the characteristics of HTTP accesses to malicious sites as seen by the edge router of a large campus network comprising upwards of 50,000 users.

A key requirement for our study is the ability to identify whether a given access is to a malicious site or not, for which, we leverage existing blacklisting tools such as the Google Safe Browsing (GSB) [16] back-end server. Thus, we do not invent any new mechanism for detecting malicious website attacks, but instead leverage existing techniques to continuously monitor the network for malicious accesses to phishing/malware websites. Our system called PhishLive monitors the HTTP traffic going through the gateway of the campus network and captures malicious URLs detected by GSB database in HTTP requests and redirect responses in real-time. It analyzes the statistical characteristics of dataset off-line including distribution of attacks over time, geolocation distribution of attacking IP addresses, attacking hostnames clustering and malicious redirect chain analysis.

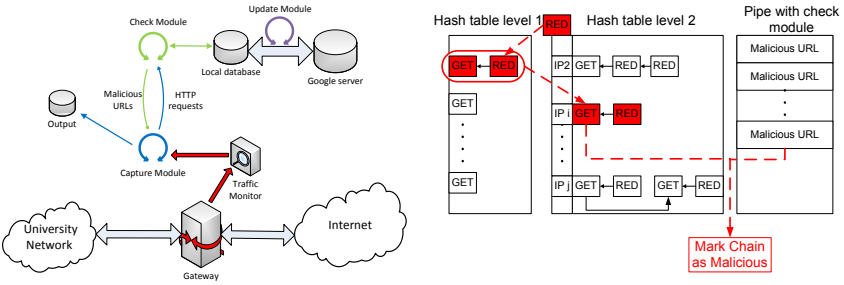
We deployed PhishLive on a large university gateway for a month during which we captured and analyzed about 1 billion URLs. Some of our key findings are:

- The fraction of URLs that are identified as belonging to phishing/malware sites is relatively small; in our data, it is less than 0.038% of all URLs.
- There is a relatively higher number of malicious URLs accessed during 11:00pm-5:00am compared to other times.
- Most domains (almost 50%) typically existed for less than 1 day. However, close to 10% of the domains were accessed for more than 15 days.
- An extremely small fraction of all HTTP redirection chains contain malicious URLs; in our data less than 2,000 URLs are part of redirection chains out of about 50 million redirection chains.

In the rest of the paper, we give an overview of the PhishLive system in Section 2 and outline our observations from a real deployment of PhishLive in Section 3.

2 System Overview

In this section, we describe the design of the PhishLive system that can continuously monitor HTTP traffic for phishing and malware attacks. We envision PhishLive to be



(a) System overview (b) Operations on two hash tables

Fig. 1. Internals of PhishLive system

deployed at an edge router, such as a campus gateway router, that can track the various HTTP requests issued by a bunch of users. We assume the presence of a standard high-speed capture device (e.g., Endace 10Gbps monitoring card) to collect each packet that is going through the gateway router to the outside world, from which we filter the HTTP traffic (port 80) and extract the URLs from HTTP GET requests. For verifying whether a given URL is malicious or not, each URL is cross checked with the Google Safe Browsing (GSB) database. Since HTTP redirects are also used to hide malicious content [17–19] in some attacks, the system is designed to also track and analyze HTTP redirect chains. We assume access to both directions of traffic for detecting redirects.

The PhishLive system comprises three components: a capture module, a check module and an update module (shown in Figure 1(a)); we describe their details next.

Capture Module. The capture module of our system utilizes the libpcap library to capture the HTTP requests from the hosts inside the university and redirect responses from the external hosts. As of now, the PhishLive system supports up to 5 types of HTTP requests: GET, HEAD, POST, PUT, DELETE, and 4 types of HTTP redirect responses: 301, 302, 303, 307. As of now, we only investigate URL redirection based on HTTP status code; other redirections based on HTML meta tag, Javascript and flash are not studied in this paper. The capture module also uses network libraries and regular expressions to extract the URLs from HTTP requests or the URLs in redirect responses, as well as source/destination IP addresses (SIP, DIP) and source/destination port numbers (SP, DP). For privacy reasons, all user-facing IP addresses are hashed.

As part of our analysis, we also wish to study the role of HTTP redirects in phishing and malware attacks. A redirect chain is a sequence of URLs starting from the first requested URL, ending with the last requested URL, that can be represented as follows: $GET(URL_1) \rightarrow REDIRECT(URL_2) \ldots \rightarrow GET(URL_n)$, where n is the number of different requested URLs in the chain. Although it appears simple, it is little tricky to track HTTP redirects in an online fashion, since it requires correlation across TCP connections (since each GET request is to a different hostname). Thus, in PhishLive, we build two hash tables (denoted as level-1 hash table and level-2 hash table in Figure 1(b)) to store HTTP redirects. The level-1 hash table holds related information of a HTTP request with a key of $\langle SIP, SP, DIP, DP \rangle$. Because the request and the redirect packet belong to the same TCP session, when a HTTP redirect response is captured, the

capture module checks if there is an existing record in the level-1 hash table with the same key. If a match is found, it means that this redirect is the response for the matched HTTP request. Then this pair (HTTP request and redirect response) is extracted from the level-1 hash table and inserted into the level-2 hash table with the SIP as key.

The level-2 hash table keeps record of all HTTP redirect chains observed in the form of a linked list. Each slot in the level-2 hash table corresponds to a user-facing IP address (inside the network). When inserting a pair (HTTP request and redirect response) into the level-2 hash table, it checks if the URL in the HTTP request matches the URL in the last record with the same key. A match indicates current request is the HTTP request of the URL in the last redirect response, and the redirect response is attached to the end of the chain. If not, a new redirect chain is built. Therefore, one slot in the level-2 hash table may contain more than one redirect chain. For instance, slot n in the level-2 hash table includes redirect chains a→b→c and x→y. When a new pair of HTTP redirect y→z is inserted, it searches for the first linked list entry a→b→c and finds that y does not match c. Then it moves to next linked list x→y, which matches and the new pair y→z is attached to the linked list resulting in x→y→z. The operations of the two hash tables are illustrated in Figure 1(b) .

The capture module also receives feedback which includes the malicious URL and the victim's IP address from the check module. It then compares the malicious URL with the records in the level-2 hash table. If the URL is found in the level-2 hash table, it will be marked as malicious and dumped to a file later on. The implementation of the capture module constitutes of three threads: one thread captures and extracts URLs from HTTP packets and feeds them to check module; another thread receives results from check module and scans level-2 hash table for a match; the last thread refreshes the two hash tables periodically to prevent them from growing too large and a fatal memory drop-off in a long-term running.

Check and Update Modules. The check module is based on the PHP API of Google Safe Browsing database provided by Google. The check module maintains a local database of malicious URLs verified by GSB server and interacts with the capture module through two pipes. Once it receives a URL from capture module, it checks the URL against the local database and feeds it back to capture module through a pipe if the URL is identified as malicious. The check module also produces general real-time statistics. The update module updates the local database with Google server periodically to ensure that the content of the local database is up-to-date.

3 Experimental Results

We deployed the PhishLive system at the edge router of a large university network over 30 days from March 19, 2012 to April 18, 2012, during which the system analyzed more than 1 billion HTTP requests (as summarized in Table 1). Out of the 1 billion URLs, only about 0.0381% of all HTTP requests were classified as requests to malicious webpages. We also observed about 50 million HTTP redirect chains out of which only about 7,500 included malicious URLs.

Since PhishLive system only captures the HTTP requests from hosts and HTTP redirect responses from servers and verifies the URL by querying GSB database, the

Table 1. Statistics of the Experiment

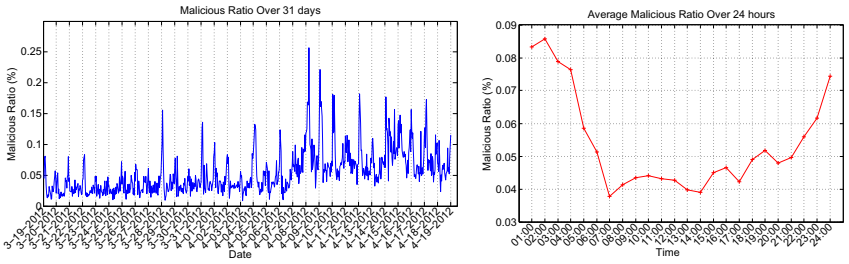
Experiment Duration	Mar 19, 2012 - Apr 18, 2012
# of HTTP Requests	1,038,803,540
# of Redirect Chains	50,204,174
# of Malicious URLs	395,671 (0.0381%)
# of Unique Malicious URLs	118,615
# of Malicious Redirect Chains	7,497 (0.0149%)

accuracy of the dataset drawn from the experiment largely depends on the accuracy of the GSB database. Previous studies [13] indicate that GSB database has a false negative rate of less than 10%; so we believe that the results are more or less accurate. However, to improve the credibility of the results, we manually verified a small sample of the URLs. Since there could be many malicious URLs with same hostname, we used a stratified sampling approach to select samples across different hostnames while ensuring larger number of samples for high volume hostnames. Specifically, we assign a weight $\frac{\#\ of\ malicious\ webpages\ of\ the\ cluster}{\#\ of\ malicious\ webpages\ overall}$ for each cluster of URLs that share the same hostname. We chose 400 samples to manually check whether they are malicious. Since total number of hostnames is less than this number, we picked at least one URL from each cluster. The remaining were sampled using the weight calculated above. We found that 93 webpages belonged to a group of fast flux hostnames (discussed in section 3.4), that we are quite certain that they are malicious. Out of the rest, we found that more than 87% of the webpages are actually malicious.

3.1 Temporal Analysis

The PhishLive system computes the number of HTTP requests and number of malicious URLs observed per hour. Host users behave very differently at different times of a day, that could mean different fraction of malicious website accesses depending on the time of day. To investigate this, we focus on the ratio of malicious website visits as a proportion of the total number of HTTP requests (which we call *malicious ratio* in our paper). From the perspective of an IDS, a higher malicious ratio at a certain time means that a HTTP connection observed at this time has higher probability to be malicious website attack.

Figure 2(a) shows the malicious ratio which varies from 0.008% to 0.257% over the one month period. Users usually make more HTTP requests during daytime than night, which could mean more malicious website accesses during daytime than night. However, from Figure 2(b) which displays the average malicious ratio for 24 hours, we observe that the malicious ratio is significantly higher (almost double) from 11 PM to 5 AM compared to daytime. Upon further investigation, we found that the absolute number of attacks is almost the same during daytime and night, with a slight edge during night (average 11,460 during daytime compared with 12,098 during nighttime). We believe the reason for this stems from the fact that these accesses could be automatically initiated by malware already present on infected computers rather than via human accesses. As the study [9] on malware infections indicates, about 58.6% of all



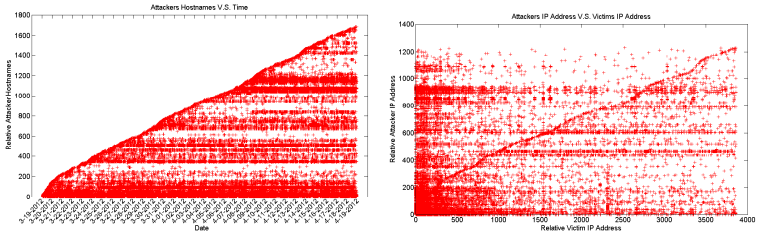
(a) Malicious ratio (b) Time of day

Fig. 2. Prevalence and location of malicious websites in our dataset. Subplot (a) shows the ratio of malicious URLs to benign over the month. (b) shows per-hour average malicious ratio.

malware make HTTP accesses, which partly explains our observations. In addition, we found a non-negligible numbers of visits to pornographic websites at nights compared to daytime, which are often associated with many types of malware.

3.2 Access Characteristics of Victims

We now analyze the malicious URL access characteristics. Specifically, we focus on the timing of the user accesses to attacker domains (IP addresses) and the relationship between victims (IP addresses) and attacker domains. Figure 3(a) displays the timing characteristics of when the attacker domains have been accessed by users. The y-axis represents distinct hostnames of attackers we observed in our data, while the x-axis is the date, with a resolution of one day. We can see that there are roughly three types of attacker domains. Type I attacker domains are those that users access frequently over a long period of time; such domains are appear as a horizontal line in the figure. Attacker domains of type II are those that may be intermittently accessed by users. They appear as dashed horizontal line in the figure. Type III attackers scatter attacks infrequently, and mostly appear as sparse points in the figure. Figure 3(b) shows the scatter plot between victims and attacker domains. From the figure, we can see that a significant number of victims seem to have contacted a few popular attacker IP addresses. Similarly, vertically, many users seem to have contacted many different attacker IP addresses as well.

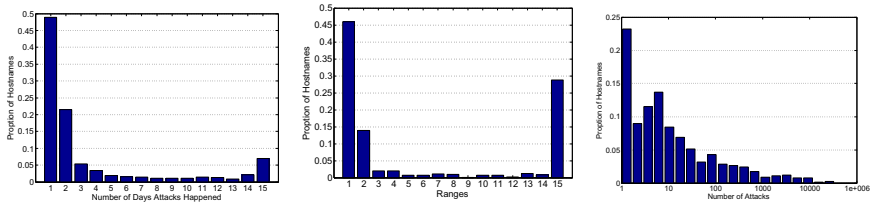


(a) Attacker hostnames (b) Victim Attacker scatter plot

Fig. 3. Behavior of attacker's hostnames and IP addresses over time

3.3 Persistence of Hostnames

The previous graphs used IP addresses; since a single IP address could host many different hostnames, we now switch to understanding the persistence characteristics of the various hostnames we observed. Since we do not exactly know the lifetime of a hostname, we only focus on duration between the first and last times a URL was observed from a given hostname as an estimate of the lifetime of a given hostname. In Figure 4(a), we plot a histogram of the number of days a particular hostname was observed in our dataset. The x-axis is the number of days an attack hostname was observed, while the y-axis shows the fraction of attacks. Since we have collected only one month's trace, we restrict ourselves to the hostnames collected in the first 15 days, so each hostname has the same chance of appearing in the next 15 days from the first time a hostname appears to eliminate the fringe bias in our data. In Figure 4(a), we found that almost 50% of the attacker hostnames were present for less than 1 day, which confirms the fast-flux like behavior observed in previous studies [20, 21]. But there are a non-trivial number of hostnames that were accessed persistently for a number of days. Close to 10% of the hostnames were accessed for all 15 days; due to lack of data beyond, we cannot conclusively determine how long these campaigns actually persisted.



(a) # of Days (b) Range of Days (c) # of HTTP accesses

Fig. 4. Temporal characteristics of malicious accesses. Subplot (a) shows distribution of hostnames in terms of number of days a hostname has been accessed. (b) shows the distribution of range, and (c) shows distribution of number of accesses.

A similar behavior can be observed in Figure 4(b), where we plot the range of the days between which we observed a given hostname. While this plot (in Figure 4(b) has similar characteristics as the previous one (Figure 4(a)), the bar for 15 days, which includes all hostnames that were active for greater than or equal to 15 days, is significantly taller (almost 30% compared to only 5%). But, the number of hostnames that were active for only 15 days is small ($<$10%), leaving a significant number of hostnames (the remaining 20%) active for $>$15 days.

Finally, we plot the distribution of the number of HTTP accesses to particular hostnames in Figure 4(c). From this figure, we can observe that most hostnames are accessed relatively infrequently. Almost 70% of hostnames were seen only less than 10 times, and 90% were seen less than 100 times in our dataset. A small number of hostnames seem to have been accessed a lot of times, greater than 33,000 times. We manually checked the 43 hostnames that were accessed more than a 1000 times. Those 43 hostnames belong to 27 different websites. Among those websites we found 1 job hunting website, 1 local

forum, 2 file sharing websites, 3 porn websites, 3 news websites and the rest of them are general websites with many different types of content.

3.4 Lexical Similarity of Domains

We now study the lexical similarity in attack domains we have observed in our data set. We decompose a malicious URL into three components: hostname, file path and query string, out of which our main interest is just the hostname. We lexically group together hostnames that share some similarity in the form of a tree. All the 395,671 malicious URLs we observed belong to 1686 distinct hostnames, 37 of which are IP addresses that we exclude in the clustering process. We first split each hostname to several tokens which are strings between two dots. For example, we break pann.nate.com into pann, nate and com tokens. Then we start to build the tree in the reverse order, starting from the top level domain name (e.g., com, nate and then pann).

End nodes in this tree are the hostnames we detected in our experiment. Children sharing the same parent node contain the same upper-level domain names. By doing this, we are trying to put different hostnames into clusters indicating the degree of similarity. From the tree, we observed that some prefixes exhibited large subtrees. For example, in our data, we observed PassingGas.net shared by 42 hostnames that differed mainly in the third-level token, such as nealyxadxloa.PassingGas.net, hccayxadxloa.PassingGas.net and so on. We observe that their third-level domain names look like they have been generated randomly; this discovery is not surprising as we observe such occurrences even in public domain phishing blacklists such as PhishTank.

What was interesting, however, is that each unique hostname was accessed for no more than 2 days by hosts in our network; such information cannot be obtained by observing blacklists such as PhishTank alone. The old hostnames became invalid, leading to a page indicating that this website is using Sitelutions Redirection Engine and the URL is either entered incorrectly or has been removed by itself. Another feature of these hostnames is that they all share the same IP address (located in Herndon, VA) no matter how they change the third level domain name. Obviously, this indicates that the attackers are manipulating the DNS Resource Records dynamically, which is not surprising as attackers often try to evade detection this way [3].

3.5 Redirect Chain Analysis

Researchers in [19] reported that attackers may use long redirect chains to hide malicious content; we therefore study whether redirections are actively used in attacks today. There are many ways to implement HTTP redirections, such as server-side 3XX response, client-side scripting (javascript), META refresh tag and so on. Since, server-side redirection is prominent in both legitimate and spam redirection [22], and since content-based analysis has significant impact on the performance of a real-time system, we choose to only focus on server-side HTTP redirects.

In our dataset, we observed a total of 7,497 redirect chains that contained at least one malicious URL. However, in some of the redirect chains, the original HTTP requests and redirect response belonged to the same hostname. If a redirect chain is created by attackers intentionally, then: (i) the URL in the redirect response typically belongs to

Table 2. Statistics of Effective Malicious Redirect Chain

Type	Number
Total Redirect Chains	50,204,174
Malicious Redirect Chains	7,497
Effective Malicious Chains	1449
Average Number of Redirect	2.221
Number of Chains Longer Than 1	246
Max Number of Redirect	5
Start with Normal Request	988
301 Redirect	231
302 Redirect	1523
303 Redirect	6
307 Redirect	0

a different hostname; (ii) the last redirect is malicious; (iii) the redirect chain usually contains more than one redirect. Therefore, we define a redirect chain to be a *effective malicious redirect chain* if the URLs in the chain belong to at least two different hostnames, and the last redirect is malicious. However, we do not put any restriction on the length of the redirect chain, since it usually does not matter.

For example, the chain consisting of redirecting http://www.dwnews.com/images/news/blog.gif → http://www.dwnews.com/ is not effective malicious redirect chain since the two URLs belong to the same hostname. However, the chain that involves the following redirection, http://grannymovs.in/ → http://lotaz.in/MyTRAFF/apiLINK da.php → http://servantspywarekeep.info/755063395c4 a385d/ is an effective malicious redirect chain, since the last URL is malicious. We also noticed a special case that the redirect chains caused by the expiration of the hostnames mentioned before. Redirects related to the expiration of those hostnames occurred 2,065 times, which we removed from our set. Among the remaining, we identified 1,449 effective malicious redirect chains, the statistics of whom are summarized in Table 2.

Several conclusions can be derived from our analysis on malicious redirect chains: (1) The number of malicious redirect chains (7497) among all redirects ($\approx$ 50 million) is quite small ($<$0.0149%). (2) Only a small portion of malicious redirect chains (246 out of 7497) are effective malicious redirect chains and contain more than 1 redirect (about 3.29%) in our experiment. (3) Most of the effective malicious chains (about 988 out of 1449) start from a normal HTTP request and end up with a malicious URL as redirect response.

4 Conclusions

To date, no studies exist on how prevalent phishing/malware attacks are or on the temporal characteristics of malware accesses in edge networks. We designed the PhishLive system for long-term monitoring of HTTP traffic of a large campus network that enabled us to study various temporal characteristics of phishing/malware attacks.

Using a month-long deployment of the PhishLive system at the university gateway router, we observed many interesting characteristics of phishing attacks. For example, we found that malicious accesses are more common during 11:00-5:00pm than during day times. Similarly, we found that most domains appeared only for one day and redirection was not common among many of the malware URLs we detected.

Acknowledgements. We thank the anonymous reviewers and Changhyun Lee, our shepherd, for their comments that improved the paper significantly. We thank the Purdue University IT staff including Bill Harshbarger and Greg Hedrick for their support in collecting the traces. This work was supported in part by NSF grant 1017915 and a grant from Google.

References

1. Egelman, S., Cranor, L.F., Hong, J.: You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In: CHI, 1065–1074 (April 2008)
2. Zhang, Y., Egelman, S., Cranor, L., Hong, J.: Phinding phish: Evaluating Anti-Phishing tools. In: NDSS (February 2007)
3. Prakash, P., Kumar, M., Kompella, R., Gupta, M.: Phishnet: Predictive blacklisting to detect phishing attacks. In: INFOCOM, pp. 1–5 (March 2010)
4. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. In: KDD, pp. 1245–1254 (June 2009)
5. Ramachandran, A., Feamster, N., Vempala, S.: Filtering spam with behavioral blacklisting. In: CCS (October 2007)
6. Garera, S., Provos, N., Chew, M., Rubin, A.D.: A framework for detection and measurement of phishing attacks. In: WORM, 1–8 (2007)
7. Zhang, Y., Hong, J.I., Cranor, L.F.: Cantina: A content-based approach to detecting phishing web sites. In: WWW, pp. 639–648 (May 2007)
8. Bayer, U., Habibi, I., Balzarotti, D., Kirda, E., Kruegel, C.: A view on current malware behaviors. In: LEET, pp. 1–11 (April 2009)
9. Rossow, C., Dietrich, C.J., Bos, H., Cavallaro, L., et al.: Sandnet: network traffic analysis of malicious software. In: BADGERS (April 2011)
10. Gu, G., Zhang, J., Wenke, L.: BotSniffer: Detecting botnet command and control channels in network traffic. In: NDSS (February 2008)
11. Perdisci, R., Lee, W., Feamster, N.: Behavioral clustering of http-based malware and signature generation using malicious network traces. In: NSDI (April 2010)
12. Song, C., Zhuge, J., Han, X., Ye, Z.: Preventing drive-by download via inter-module communication monitoring. In: ASIACCS, pp. 124–134 (April 2010)
13. Whittaker, C., Ryner, B., Nazif, M.: Large-scale automatic classification of phishing pages. In: NDSS (February 2010)
14. Provos, N., Mavrommatis, P., Rajab, M.A., Monrose, F.: All your iframes point to us. In: IEEE S&P Conference (Oakland), pp. 1–15 (May 2008)
15. Maier, G., Feldmann, A., Paxson, V., Sommer, R., Vallentin, M.: An Assessment of Overt Malicious Activity Manifest in Residential Networks. In: Holz, T., Bos, H. (eds.) DIMVA 2011. LNCS, vol. 6739, pp. 144–163. Springer, Heidelberg (2011)
16. Google safe browsing API, `https://developers.google.com/safe-browsing/`

17. Webb, S., Caverlee, J., Pu, C.: Introducing the webb spam corpus: Using email spam to identify web spam automatically. In: CEAS (July 2006)
18. Webb, S., Caverlee, J., Pu, C.: Characterizing web spam using content and http session analysis. In: CEAS (July 2007)
19. Lee, S., Kim, J.: Warningbird: Detecting suspicious URLs in twitter stream. In: NDSS, pp. 1–13 (February 2012)
20. Konte, M., Feamster, N., Jung, J.: Dynamics of Online Scam Hosting Infrastructure. In: Moon, S.B., Teixeira, R., Uhlig, S. (eds.) PAM 2009. LNCS, vol. 5448, pp. 219–228. Springer, Heidelberg (2009)
21. Holz, T., Gorecki, C., Rieck, K., Freiling, F.: Measuring and detecting fast-flux service networks. In: NDSS (February 2008)
22. Bhargrava, K., Brewer, D., Li, K.: A study of URL redirection indicating spam. In: CEAS (July 2009)