This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

PhishNet: Predictive Blacklisting to Detect Phishing Attacks

Pawan Prakash, Manish Kumar, Ramana Rao Kompella, Minaxi Gupta†
Purdue University, †Indiana University

***Abstract*—Phishing has been easy and effective way for trickery and deception on the Internet. While solutions such as URL blacklisting have been effective to some degree, their reliance on exact match with the blacklisted entries makes it easy for attackers to evade. We start with the observation that attackers often employ simple modifications (*e.g.*, changing top level domain) to URLs. Our system, PhishNet, exploits this observation using two components. In the first component, we propose five heuristics to enumerate simple combinations of known phishing sites to discover new phishing URLs. The second component consists of an approximate matching algorithm that dissects a URL into multiple components that are matched individually against entries in the blacklist. In our evaluation with real-time blacklist feeds, we discovered around 18,000 new phishing URLs from a set of 6,000 new blacklist entries. We also show that our approximate matching algorithm leads to very few false positives (3%) and negatives (5%).**

I. Introduction

The simplicity and ubiquity of the Web has fueled the revolution of electronic commerce, but has also attracted several miscreants into committing fraud by setting up fake web sites mimicking real businesses, in order to lure innocent users into revealing sensitive information such as bank account numbers, credit cards, and passwords. Such *phishing attacks* are extremely common today and are increasing by the day [1]. One popular solution to address this problem is to add additional security features within an Internet browser that warns users whenever a phishing site is being accessed. Such browser security is often provided by a mechanism known as 'blacklisting', which matches a given URL with a list of URLs belonging to a *blacklist*. Most blacklists are generated by a combination of procedures that involve automatic mechanisms and humans.

Although blacklists provide simplicity in design and ease of implementation by browsers and many other application, a major problem with blacklists is *incompleteness*. The reason is that today's cyber-criminals are extremely savvy; they employ many sophisticated techniques to evade blacklists. At some level, the incompleteness problem cannot be solved easily since malicious URLs cannot be known before a certain amount of prevalence in the wild. Despite the inherent difficulty in exhaustive prediction, we *observe* that malicious URLs do often tend to occur in groups that are close to each other either syntactically (*e.g.*, www1.rogue.com, www2.rogue.com) or semantically (*e.g.*, two URLs with hostnames resolving to the same IP address). There are two direct implications of this simple observation. First, if we can exploit this observation to systematically discover new sources of maliciousness in and around the original blacklist entries and add them to the blacklist, that would significantly increase its resilience to evasion. Second, we can deviate from the exact match implementation of a blacklist to an approximate match that is aware of several of the legal mutations that often exist within these URLs. In this paper, we describe the architecture of PhishNet, a system that combines these two ideas to improve the resilience and efficiency of blacklists significantly.

PhishNet comprises two major components: 1) a *URL prediction* component (Section II) that works in an offline fashion, examines current blacklists and systematically generates new URLs by employing various heuristics (*e.g.*, changing the top-level domains). Further, it tests whether the new URLs generated are indeed malicious with the help of DNS queries and content matching techniques in an *automated* fashion, thus ensuring minimal human effort. 2) an *approximate URL matching* component (Section III) which performs an approximate match of a new URL with the existing blacklist. It uses novel data structures to perform approximate matches with an incoming URL based on regular expressions and hash maps to catch syntactic and semantic variations.

To evaluate the URL prediction component, we collected live feeds from PhishTank [2] over 24 days and generated about 1.5 million combinations from 6,000 URLs that resulted in about 18,000 new URLs after the vetting process. Using malicious URLs from PhishTank and SpamScatter [3], and benign URLs from DMOZ and Yahoo, we found that our system enjoys low false negatives (less than 3%) and false positives (less than 5%). In comparison with the Google Safe Browsing alternative, PhishNet is significantly faster even though it employs approximate matching.

II. Component 1: Predicting Malicious URLs

In this section, we describe the first component of the PhishNet, which predicts new malicious URLs from existing blacklist entries. We study *five* different heuristics that allow synthesizing new URLs from existing blacklist entries. Our heuristics are derived based on prior studies [1], [4] that have observed the prevalence of lexical similarities in URLs as well as our own observations on the PhishTank database [2]. The *basic idea* of our approach is to combine pieces of known phishing URLs (*parent*) from a blacklist to generate new URLs (*child*). We then test the existence of these child URLs using a verification process (section II-B).

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

A. Heuristics for generating new URLs

Typical blacklist URLs have the following structure: http://domain.TLD/directory/filename?query_string. The directory specifies the path with the file which is passed with a query string, together forming the *pathname* portion of the URL. Our heuristics (discussed below) involve interchanging these field values among URLs clustered together lexically or along some other dimension.

1) H1, Replacing TLDs: Our first prediction heuristic relies on proactively finding such variants of original blacklist entries obtained by changing the TLDs. We use $3,210$ effective top-level domains (TLDs)—longest portion of hostname that should be treated as top-level domain (*e.g.*, co.in)—obtained from [5]. Thus, for each new URL that enters a given blacklist, we replace the effective TLD of the URL with $3,209$ other effective TLDs that form the candidate child URLs that need to be validated.

2) H2, IP address equivalence: Separate phishing campaigns on the same infrastructure (same IP addresses) may share the directory or path structure among each other. In order to obtain new URLs using this heuristic, we maintain host equivalence classes in which phishing URLs having same IP addresses are grouped together into clusters. We then create new URLs by considering all combinations of hostnames and pathnames.

3) H3, Directory structure similarity: The basic intuition here is that, there is a good chance that two URLs sharing a common directory structure may incorporate similar set of file names. For example, if www.abc.com/online/signin/paypal.htm and www.xyz.com/online/signin/ebay.htm are two known phishing URLs then our heuristic predicts the existence of www.abc.com/online/signin/ebay.htm and www.xyz.com/online/signin/paypal.htm. We maintain a path equivalence class in which URLs with similar directory structure are grouped together. We build new URLs by exchanging the filenames among URLs belonging to the same group.

4) H4, Query string substitution: Query string is often a very simple way to inflicting subtle changes to the URL without changing the ultimate destination. In our analysis of the PhishTank database, we observed several URLs with exact same directory structure differing only in query string part of the URL. Thus, if we have two URLs, www.abc.com/online/signin/ebay?XYZ, and www.xyz.com/online/signin/paypal?ABC, we create two new URLs, www.abc.com/online/signin/ebay?ABC and www.xyz.com/online/signin/paypal?XYZ. In order to ensure we have finite combinations, we only consider existing combinations in the blacklist. As in $H3$ we create path equivalence class and build new URLs by exchanging the query strings among URLs.

5) H5, Brand name equivalence: The key observation behind our brand name heuristic is that, phishers often target multiple brand names using the same URL structure method. For example, RockPhish gang [6] uses this method in their fraud infrastructure. We, therefore, build new URLs by substituting brand names occurring in phishing URLs with other brand names, in essence treating all brands as an equivalence class. Thus, for n URLs that have a brand name embedded within them, we substitute each occurrence of a brand name with k other brand names. We use a list of 64 brand names most targeted by phishers.

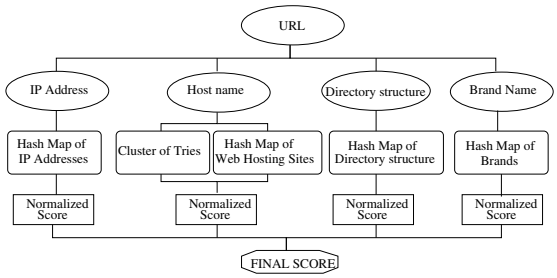

Fig. 1. Computing the score of a new URL in PhishNet. If the score is above a threshold, we flag the URL as a phishing site.

B. Verification

Once we create the child URLs, we subject them to a validation process whereby we eliminate URLs that are either non-existent or are non-phishing sites. As a first step, we conduct a DNS lookup to filter out sites that cannot be resolved. For each of the resolved URLs, we try to establish a connection to the corresponding server. For each successful connection, we initiate a HTTP GET request to obtain content from the server. If the HTTP header from the server has status code $200/202$ (successful request), we perform a content similarity between the parent and the child URLs using a publicly available similarity detection tool [7]. If the URL's content has sharp resemblance (above say 90%) with the parent URL, we can conclude that the child URL is a legitimate bad site that needs to be added to the blacklist.

III. Component 2: Approximate Matching

Motivated by the success of heuristics to predict new URLs to add to the blacklists (as described in Section IV-A), we use the fundamental principles behind these heuristics to devise an approximate matching component within PhishNet that determines whether a given URL is a phishing site or not, even when the exact URL does not match any entry in the blacklist. It performs an approximate match of a given URL to the entries in the blacklist by first breaking the input URL into four different entities—IP address, hostname, directory structure and brand name (as shown in Figure 1)—and, scoring individual entities by matching them with the corresponding fragments of the original entries to generate one final score. If the score is greater than a pre-configured threshold, it flags the URL as a potential phishing site. We now describe these individual modules within the approximate matching component and our scoring technique in more detail next.

A. M1: Matching IP address

As we found in heuristic $H2$, there are many phishing URLs with different domains which resolve to the same IP address. In the IP address module of our component, we perform a direct match of the IP address of URL with the IP addresses

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

of the blacklist entries and assign a *normalized* score based on the number of blacklist entries that map to a given IP address. The score is formally computed as follows. If IP address IP_i is common to n_i URLs, the score assigned to the URL is:

$$\frac{n_i - min\{n_i\}}{max\{n_i\} - min\{n_i\} + 1} \quad (1)$$

where $min\{n_i\}$ ($max\{n_i\}$) corresponds to the minimum (maximum) of the number of phishing URLs hosted by blacklisted entries of IP addresses.

B. M2: Matching hostname

In the second module of this component, we focus on performing hostname match with those in the blacklist and assign a score based on the relative prevalence of the hostname among all the blacklist URLs. We note that a significant percentage of phishing URLs either have domains specifically registered for hosting phishing sites or are hosted on free/paid-for web-hosting services (WHS). They need to be considered differently since WHS-based sites typically have a different structure than those that are non-WHS sites. We identify whether an incoming URL consists of a WHS or not by matching the primary domain in a pre-computed list of hosting sites. This classification helps in developing a better approach to devise our data structures and to assign different scoring techniques. We describe these individually next.

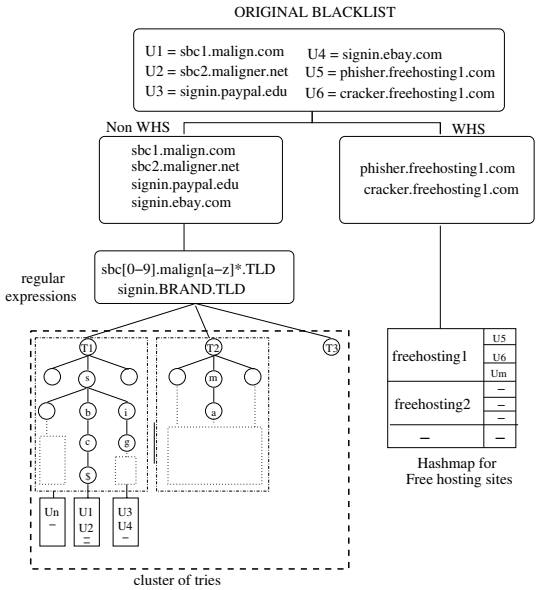

Fig. 2. Data structures used in hostname representation.

1) Matching WHSes: URLs from the blacklist database are clustered on the basis of their primary domains and stored in a *hash map* with the primary domain as the key. The value corresponding to a key in this hash map represents the number of URLs sharing the same WHS as their primary domain name. In Figure 2, freehosting1 is a key in the hashmap and the corresponding value is the hashtable consisting of URLs U5, U6, Um. The primary domain of an incoming URL is looked up in the hash map. If the match succeeds, a confidence score is computed using the same function as the expression in (1) on the number of URLs that have the same primary domain. In essence, the higher the number of malicious sites a particular WHS is known to have hosted before, the higher the score that is assigned to that WHS.

2) Matching non-WHSes: The URLs that are not based on WHSes, are clustered on the basis of syntactic similarity across labels (*i.e.*, words separated by dot in an URL). For example, the URLs http://chaseonline.chase.com.illifi.com.mx and http://chaseonline.chase.com.hhili.com.mx only differ the fourth label. To handle such URLs, we first dissect the hostname portion of such URLs into individual labels. Our approximate matching data structure comprises multiple tries—one for each of the label positions. We process the phishing URLs from our database by considering the sets of labels at each and every position and then form regular expressions representative of these groups of labels.

The regular expressions formed at each label are stored in a trie data structure. Corresponding to each leaf in the trie, we store the associated regular expression. The score of a given regular expression is computed similar to the expression in (1), with n_i referring to the number of URLs that match a given regular expression. An incoming hostname is broken down into its corresponding labels and is searched in corresponding tries. A regular expression match in any label returns the score associated with the leaf. We compute the average normalized score from the individual label matches as the overall score from this component.

Example. Figure 2 shows the construction of regular expressions and tries from an example set of 4 non-WHS URLs. The three unique labels—{sbc1, sbc2, signin}—are first converted into two regular expressions {sbc[0-9], signin}, both of which are then represented in the trie data structure (with a special $ sign indicating a wild card number match). Thus, this representation using regular expressions allows PhishNet to be resilient against simple subtle variations of labels. In Figure 2, U1 and U2 are the two URLs which have the regular expression sbc[0-9] in their first label. So when an input hostname contains a label which matches this regular expression, it will be assigned a normalized score computed using expression (1) with $n_i = 2$.

C. M3: Matching directory structure

This module consists of a hash map with directory structure as the key. Corresponding to each key in the hash map, the number of phishing URLs in our database that contain that directory structure is maintained. The philosophy of this design follows from the heuristics H3 (*directory structure similarity*) and H4 (*query string substitution*), both of which rely on combining phishing URLs that share the same directory structure with different filename portion or query strings. In effect, this ensures that we assign a high score to an incoming URL, if it has the same directory structure seen before in hosting many phishing URLs in the past. The calculation of the normalization score is done similarly as in (1), with n_i

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the Mini-Conference at IEEE INFOCOM 2010

representing the number of URLs corresponding to a directory structure in the hash map.

D. M4: Matching brand names

This component of our system checks for existence of brand names in pathname and query string of URLs. The brand names are assigned a frequency score which is normalized using (1), with n_i being the number of occurrences of the brand name. An incoming URL is checked against the brand names contained in our list. In case of a match, the normalized score corresponding to that brand is returned. The confidence scores returned from each test above is used to compute a final cumulative score. We have assigned different weights to different components of PhishNet. We selected the weights empirically, but loosely based on the yield of new URLs we obtained from different heuristics. For example, high yield of "directory structure similarity" heuristics shown in section IV-A motivated us to assign a higher weight to the path component. If w_1, w_2, w_3 and w_4 are the weights assigned to the different components and c_1, c_2, c_3 and c_4 are the confidence scores, then the final cumulative score is $\sum_{i=1}^{4} w_i \times c_i$.

IV. Evaluation

We now validate the efficacy of our heuristics in generating new phishing URLs, and analyze the performance of our approximate matching component.

A. COMPONENT 1: Predicting Malicious URLs

Our goal is to evaluate the efficacy of the heuristics outlined in Section II-A in identifying new sources of maliciousness. As earlier studies suggest that phishing domains are live for a very short period of time [4], we build a system that collects *live* URLs from PhishTank feeds every 6 hours, applies heuristics to generate new URLs and validates them. Using this methodology, we have collected URLs over a period of 24 days starting from 2^{nd} July 2009 to 25^{th} July 2009. Table I presents individual results of each of the heuristics.

As a result of the five heuristics, we generated almost 1.55 million child URLs from the approximately 6,000 parent URLs. Out of these 1.55 million URLs, only about 14% URLs had an associated DNS entry. For about 80% of the URLs with DNS hits, we could connect to the server. About half (84,932 out of 172,449) of GET requests in our experiments received 404 reply which implies "page not found" message. For about 20% of the URLs, we could fetch the content (status code `200`), which we then, compared against the parent URL using the page similarity tool [7]. For all heuristics we find a bi-modal distribution with most URLs either having high or low similarity scores. In other words, several of the tested URLs seem to be conclusively similar or dissimilar; conclusively similar ones with greater than 90% similarity are reported as our new phishing URLs.

B. COMPONENT 2: Approximate Match

We evaluate the effectiveness of the approximate matching component in terms of the false positives and false negatives

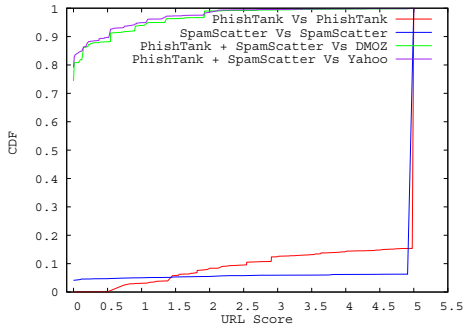

Fig. 3. CDF of scores with different training and testing data sets.

suffered in detecting phishing URLs and, the amount of time it takes to process a URL. In all the experiments, we use data from four sources. We use two sources of phishing URLs—PhishTank (consists of about $18,000$ URLs) and SpamScatter ($14,000$ URLs). In addition to these two, we have $100,000$ benign URLs from an open directory service called DMOZ [8], and almost $20,000$ benign URLs from Yahoo Random URL generator (YRUG) [9]. Our experimental methodology is similar to that in [10]. The experimental setup consists of two phases—training and testing. In the training phase, we create various data structures (Section III) using the phishing URLs. During testing, an input URL is flagged as a phishing or a genuine site. If a benign (malicious) site is (not) flagged as malicious, it is counted as a false positive (negative). For the purposes of our evaluation, we have used the following weight assignment to different normalized scores from individual modules: W(M1) = 1.0, W(M2) = 1.0, W(M3) = 1.5, and W(M4) = 1.5. The larger weight associated with M3 and M4 are because of the relatively larger number of new URLs found with heuristics H3 and H5. The final score computed lies between 0 and 5.

To obtain appropriate threshold, we compute the cumulative distribution (CDF) of score values output by our system when both trained and tested on either malign or benign URLs. We divide the data set into two halves by picking URLs at random and use one half for training and one half for testing. From Figure 3, we can observe a clear trade-off between false positives and negatives; a higher threshold would result in increased false negatives, while lowering the threshold would result in more false positives. Depending on what an administrator deems acceptable, one can pick different threshold values. We observe that a threshold of 1 ensures a good balance between false positives and negatives.

C. Comparison with Google's browser blacklist

Google Safe Browsing API [11] allows end user to examine the nature of a URL by comparing against Google's constantly-updated list of suspected phishing and malware pages. We start with 153 URLs which were present both in Google's blacklist and PhishTank. We apply our URL prediction heuristics on these parent URLs and checked the predicted ones within Google's blacklist through the safe browsing API. We found out that about 312 new URLs created

	H1 (TLD)	H2 (IP)	H3 (Directory)	H4 (Query)	H5 (Brand)	Total
URLs Gen	1,369,083	6,213	115,896	44,810	14,109	1,550,111
DNS Hits	46,443	5,267	114,891	41,673	7,612	215,886
Connected to Server	28,813	4,336	112,015	20,776	6,509	172,449
HTTP Status Code						
200	6,486	408	19,242	8,255	627	35,018
3XX	288	12	8,347	3,280	371	12,298
302	5,266	256	7,691	3,782	692	17,687
4XX	221	2,193	1,617	15	261	4,307
404	9,846	1,460	63,850	5,383	4,393	84,932
5XX	400	2	11,149	1	49	11,601
Similarity Value						
0-29	6,245	106	7,367	1,445	-	15,163
30-59	25	6	461	0	-	492
60-89	79	46	808	0	-	933
90-100	137	250	10,606	6,810	-	17,803

TABLE I
NEW URLS OBTAINED AS A RESULT OF APPLYING HEURISTICS H1-H5 ON ABOUT 6,000 URLS COLLECTED FROM THE PHISHTANK FEED.

by our heuristics were also present in the Google's blacklist, which indicates that our system produces good results.

We also compare the amount of time taken by Google's API and by PhishNet's approximate matching component. We find that, on an average, our system performs 80 times faster than Google's API (1ms lookup time compared to about 80ms for Google). This excludes the time taken by an end user to download the Google blacklists locally and also excludes the time taken by PhishNet for DNS resolution of the URLs.

V. RELATED WORK

While several prior techniques have been proposed for detecting phishing attacks, to the best of our knowledge, we are the first to design a system that involves systematic generation of new URLs. Our work derives important observations about basic differences between phishing and non-phishing URLs from [4]. They point out subtle features about the anatomy of phishing URLs and how these phishing campaigns are launched using web services like URL aliasing and Web hosting. The Anti Phishing Work Group [1] also regularly publishes facts and figures about phishing such as list of TLDs and brand names targeted, trends in phishing URL structure, etc., that can be used to derive more heuristics.

In [12], the authors propose highly predictive blacklists that use a page rank-style algorithm to estimate the likelihood that an IP address be malign. Ma *et al.* consider the problem of matching URLs with blacklist entries in [10]. They rely on tens of thousands of features based on extra information from outside sources such as WhoIS, registrar information, etc., to train a machine-learning based classifier. In contrast, our algorithm operates with very few lexical features in the URLs, yet achieves similar false positive and negative rates. CANTINA also looks at features related to content, lexical and WHOIS to classify the phishing URLs [13].

VI. CONCLUSIONS

Blacklisting is the most common technique to defend against phishing attacks. In this paper, we proposed PhishNet to address major problems associated with blacklists. PhishNet has two major components. The first grows blacklists by generating new URL variations from the original ones but after vetting them through DNS and content matching. The second component consists of an approximate matching data structure that assigns a score to each URL based on piece-wise similarity with existing URLs. PhishNet suffers from low false positives and is remarkably effective at flagging new URLs that were not part of the original blacklist. While we have outlined several heuristics, there could be more. Exhaustively identifying new heuristics and evaluating their efficacy is a problem we will address in our future work.

VII. ACKNOWLEDGMENTS

This work was supported in part by NSF Award CNS 08316547 and a grant from Cisco Systems.

REFERENCES

[1] APWG, "Anti phishing work group," `http://www.antiphishing.org/`.
[2] OpenDNS, "Phishtank," `http://www.phishtank.com`.
[3] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker, "Spamscatter: Characterizing internet scam hosting infrastructure," in *USENIX Security*, 2007.
[4] D. K. McGrath and M. Gupta, "Behind phishing: An examination of phisher modi operandi," in *In Proc. of the USENIX Workshop on Large-Scale Exploits and Emergent Threats(LEET), San Francisco, CA*, 2008.
[5] M. Foundation, "Public suffix list." `http://publicsuffix.org`.
[6] T. Moore and R. Clayton, "Examining the Impact of Website Take-down on Phishing." in *Second APWG eCrime Researcher's Summit, Pittsburgh, PA,USA*, 2007.
[7] "Similarity page checker," `http://www.webconfs.com`.
[8] NetScape, "Dmoz open directory project." `http:www.dmoz.org`.
[9] YURG, "Dmoz open directory project." `http://random.yahoo.com/bin/ryl`.
[10] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *KDD*. ACM, 2009, pp. 1245–1254.
[11] "Google safe browsing api," http://code.google.com/apis/safebrowsing/developers_guide.html.
[12] J. Zhang, P. A. Porras, and J. Ullrich, "Highly predictive blacklisting." in *USENIX Security Symposium*, P. C. van Oorschot, Ed. USENIX Association, 2008, pp. 107–122.
[13] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *WWW*, 2007, pp. 639–648.