Tracking Phishing Attacks Over Time

Qian Cui
University of Ottawa
Ottawa, Canada
qcui@uottawa.ca

Guy-Vincent Jourdan
University of Ottawa
Ottawa, Canada
gjourdan@uottawa.ca

Gregor V. Bochmann
University of Ottawa
Ottawa, Canada
bochmann@eecs.uottawa.ca

Russell Couturier
CTO Forensics, IBM Security
U.S.A.
russ.couturier@us.ibm.com

Iosif-Viorel Onut
Principal R&D Strategist, IBM
Centre for Advanced Studies
Ottawa, Canada
vioonut@ca.ibm.com

ABSTRACT

The so-called "phishing" attacks are one of the important threats to individuals and corporations in today's Internet. Combatting phishing is thus a top-priority, and has been the focus of much work, both on the academic and on the industry sides. In this paper, we look at this problem from a new angle. We have monitored a total of 19,066 phishing attacks over a period of ten months and found that over 90% of these attacks were actually replicas or variations of other attacks in the database. This provides several opportunities and insights for the fight against phishing: first, quickly and efficiently detecting replicas is a very effective prevention tool. We detail one such tool in this paper. Second, the widely held belief that phishing attacks are dealt with promptly is but an illusion. We have recorded numerous attacks that stay active throughout our observation period. This shows that the current prevention techniques are ineffective and need to be overhauled. We provide some suggestions in this direction. Third, our observation give a new perspective into the modus operandi of attackers. In particular, some of our observations suggest that a small group of attackers could be behind a large part of the current attacks. Taking down that group could potentially have a large impact on the phishing attacks observed today.

Keywords

Phishing Detection; Clustering

1. INTRODUCTION

So-called "phishing" has been defined as "a type of computer attack that communicates socially engineered messages to humans via electronic communication channels in order to persuade them to perform certain actions for the attacker's benefit" [18]. Although this definition is quite general, the focus is usually put specifically on phishing web sites (e.g. [29]), which is our focus as well.

Phishing is an ongoing threat to individuals and to the community at large. According to a recent report produced by the FBI [9], from October 2013 to February 2016, phishing scam caused at least $2.3 billion in damages, involving 17,642 businesses in more than 79 countries. What is more, the Anti-Phishing Working Group (APWG, [3]) reports that in the first quarter of 2016, the number of phishing attacks increased by 250 percent when compared to the phishing attacks in the last quarter of 2015 [5]. Many companies have joined in the campaign against phishing. For instance, Google maintains a blacklist of phishing sites and has built a phish filter into its Chrome browser [29, 19]. Other companies have been developing a variety of anti-phishing tools, such as toolbars or browser extensions, to identify and block phishing sites [32]. Companies such as IBM do offer tools and services that include phishing prevention [16], and antivirus software such as Panda [24] and McAfee[22] now include some anti-phishing features.

Improving the accuracy and speed of phishing sites detection is very important. Academia has been working on this for the last several years, working from two main angles: the first approach is to find similarities between a phishing site and the legitimate site that it mimics. The second approach is to try to find intrinsic characteristics of phishing sites, such as the presence of specific types of web forms, or some unusual structures in URLs (see Section 5 for more details). In this paper, we are following a different approach: we have been monitoring a total of 19,066 confirmed phishing attacks reported on PhishTank[1] to try to find commonalities between these attacks. To that end, we have defined a feature vector based on the DOM of the web pages obtained from these attacks, and we have then clustered together vectors that are close to one another. We found that over 90% of our phishing database ends up in "flagged clusters", that is, cluster with more than one attack in it. Moreover, we only have 1,216 such flagged clusters. Testing our method with 24,800 legitimate sites only yields 20 false positive (0.08%). Our method is thus extremely effective at detecting an attack that happens to be a replica or a variation of another, already known attack. We say that these phishing attack **instances** belong to the same phishing **class**.

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052654

[1] https://www.phishtank.com/

A typical in-depth defense strategy against phishing consists of a first line of fast filtering, which will flag some sites as potential phishing sites, followed by a second line of much slower, in-depth analysis to confirm that the flagged sites are indeed attacks. Since we see that most of the new attacks are variations of known ones, it shows that detecting such replicas as an intermediary step is a sensible strategic move that will free up a considerable amount of much needed resources for the last step of in-depth analysis. Our results have been confirmed on proprietary production data owned by the security division of our partner IBM.

By clustering together all the known instances of an attack over several months, we can also learn about how these attacks are usually managed by the attackers. What we have found is that a single attack class can last for a very long time. We have examples of attack classes being live for the entire duration of our database, ten months, and they probably last much longer. We see that attackers are moving these attacks between different domains, different IP addresses and different URLs, but do not modify their actual attacks very much at all. This shows that the very short average lifetime of a given phishing site, about 10 hours according to [4], does not fairly reflect the reality of these attacks: if a given instance of a given attack is short-lived, the attack class itself seems to be easily maintained alive by the attacker. We have seen several attacks for which past instances were directly blocked by a browser such as Chrome, but newer instances of the same attack were not. It suggests that the current prevention methods wrongly focus on these instances of attacks and fail to address the core of the problem. The clustering method presented in this paper will help with this situation, by protecting against future instances of an attack once a single instance has been detected.

We also used our clusters to study possible connections between attack classes: as explained, attacks that are instances of a given class are moved across servers and across IP addresses by the attackers. We looked at the set of IP addresses used by each of our clusters, searching for addresses that would have been used at different times by different clusters. What we found was that 1,259 IPs are reused in at least one pair of clusters, and these 1,259 IPs represents directly about 50% of the recorded attacks. This suggests that these attacks, albeit targeting completely different sites, do share some common resources and are thus possibly operated by the same groups. Our study provides information that could help identifying these groups and thus possibly stop a large number of the current attacks.

The paper is organized as follows: In Section 2 we provide the definitions for our clustering approach. Then in Section 3, we present the basic results of our experiments. We discuss these results, and provide more analysis in Section 4. We provide an overview of the literature in Section 5 before concluding is Section 6.

2. CONCEPTS AND DEFINITIONS

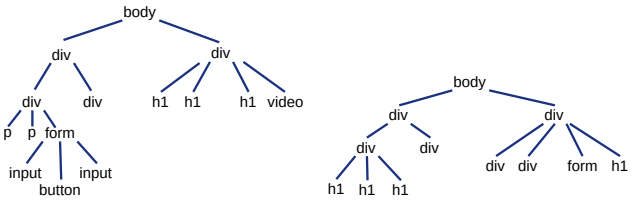
In this section, we introduce and discuss the various mathematical concepts that we have used in our analysis. It should be noted that many different DOM comparison techniques could have been used with probably similar results. The one presented here was chosen because it is fast to compute and yield good results.

2.1 Tag Vectors and Proportional Distance

Our goal is to be able to find attack instances that are fairly similar to one another, with some level of flexibility in the definition of similarity. In [31], the authors explain that they have used a "Hash-based Near-Duplicate Phish Detection" method that, in their case, finds that 72.67% of their phishing site database are replicas from at least one other site in that same database. This "hash-based" method takes the HTML of the phishing page and removes all the spaces from it. It also removes all the default values in all INPUT fields, replacing these values by empty strings. The resulting HTML is then hashed using SHA-1 [23].

We do not want to use a hash-based method, which are very strict and fail to match pages that are very small variations of each other. Instead, we want to compare the structure of the DOMs, using criteria that are fast to compute. Our idea is to compute what we have called the "tag vector" of the phishing page.

We first define a corpus of HTML tags. We used the complete set of HTML elements provided by the World Wide Web Consortium [30], from which we have removed some of the more common tags such as <body>, <head> and <html>. We end-up with a list of 107 tags, which can be fetched from `http://ssrg.site.uottawa.ca/phishingdata`. We then define an arbitrary but fixed ordering of the tags in the corpus, and we define a "vector" of integers of the size of the corpus. For each DOM, we compute the corresponding vector by counting how many times each tag of the corpus appears in the "body" part of the DOM. As an example, consider Figure 1, where two simple DOMs are provided. If the corpus consists of the html tags <form> <b> <p> <h1> <button> <video> <input> <iframe> and , in that order, then the tag vector for the page p_1 is <1, 0 ,2 ,3, 1, 1, 2, 0, 4>, while the tag vector for the page p_2 is <1, 0 ,0 ,4, 0, 0, 0, 0, 6>.



(a) DOM of the page p_1 (b) DOM of the page p_2

Figure 1: Tag vectors

In order to compare two pages, we use what we call the **proportional distance** between the tag vectors of the two pages. Here again, we will use a fairly simple definition, in which we will count the number of tags that appear a different number of times in the two pages (in other words, the number of indexes of the tag vectors that have a different value). This simple count would have the tendency to generate greater differences for pages that are relatively complex and that use a large variety of tags from the corpus, while pages that only use a few of the tags would have smaller differences. In order to alleviate this tendency, we will divide the number of differences by the number of tags that appear in at least one of the two pages.

Formally, for all integers x and y, we define the Hamming Distance $D(x, y) = 1$ if $x \neq y$ and $D(x, y) = 0$ otherwise. We define $L(x, y) = 1$ if $x \neq 0$ OR $y \neq 0$ and $L(x, y) = 0$

otherwise. Let t_1 and t_2 be two non-null tag vectors over the same corpus of size n. We define the **proportional distance**[2] $PD(t_1, t_2)$ between t_1 and t_2 as follows:

$$PD(t_1, t_2) = \frac{\sum_{i=1}^{n} D(t_1[i], t_2[i])}{\sum_{i=1}^{n} L(t_1[i], t_2[i])} \quad (1)$$

As an example, consider the two vectors from Figure 1: seven of the nine possible tags of the corpus are used by at least one of the two vectors, and only the first one, <form>, appears the same number of times in both vectors, so the $PD(p_1, p_2) = 6/7 = 0.85$, which indicates a large difference between the two pages.

2.2 Clustering Algorithm

The goal of our clustering is to group together in the same cluster the pages that have relatively similar tag vectors, since they are deemed to be similar to one another. We believe that a given site can be slightly modified a great number of times and re-published after each modification. It is thus unlikely that there is some kind of a "master" site that is used as the source for each iteration. Instead, we are probably looking at long sequences of modifications, in which each new instance is fairly similar to the instance it is based upon, but after a while we may end-up with instances that are fairly far apart. We thus avoided to use a centroid-based clustering algorithm. Instead, we use a hierarchical clustering approach to group together vectors that have at least one other vector which is relatively similar in the same cluster. This has the advantage of always producing the same output for the same input, regardless of the order in which the vectors are added to the model. This is a significant advantage in our context, since the database is going to evolve overtime, as new phishing sites are being discovered. These new sites can be added to the model in a greedy fashion, without having to recompute the model. Note that when a new site is added to the set, two clusters can end-up being merged together. A straightforward greedy algorithm was used in our experiments, with an overall complexity of $O(n^2)$ for n vectors. A more efficient method using "prototypes" for the clusters [27] was implemented for production, but the details are out of the scope of this paper.

As usual with clustering algorithms, our method is based on some threshold H. A common and intuitive definition of a "good" clustering threshold is a threshold that yields clusters that are both compact and far away from each other, so this is what we also aim for here. We define a **quality of clustering** using the average proportional distance of vectors inside a cluster. We average this value, and divides it by the smallest proportional distance between two vectors in any two clusters. The numerator evaluates the compactness of the clusters (the more compact the clusters, the smaller the value) while the denominator evaluates how far apart clusters are: the further apart they mutually are, the larger the value. Thus, a smaller value for our quality of clustering is better.

Formally, given $C_k \subseteq C$ a subset of the set of clusters C having more than one element, we define the quality of clustering as follows: for a given cluster $C_i \in C_k$ having $n_i > 1$ elements, we define the compactness of the cluster C_i as $Comp(C_i) = \frac{2}{n_i(n_i-1)} \sum_{x \in C_i, y \in C_i, x \neq y} PD(x, y)$, and the compactness of the set of clusters C_k as $Comp(C_k) = \frac{1}{|C_k|} \sum_{C_i \in C_k} (Comp(C_i))$. We also define the minimal distance between two clusters C_i and C_j as $Min(C_i, C_j) = min_{x \in C_i, y \in C_j} PD(x, y)$ and the minimal distance in a set of cluster C as $Min(C) = min_{C_i, C_j \in C, C_i \neq C_j} Min(C_i, C_j)$ The **Quality of Clustering** $QC(C)$ of C is defined as

$$QC(C) = \frac{Comp(C_k)}{Min(C)} \quad (2)$$

3. EXPERIMENT

We are now reporting our basic results, starting with a description of our database.

3.1 Legitimate- and Phishing-Sites Database

We have used the community-driven portal PhishTank to compile our database of phishing attacks instances URLs. We have collected a raw list of 21,303 "verified" phishing sites by downloading PhishTank's archive daily between January 1st and October 5th, 2016[3]. We also recorded the submission date of each attack instance. For each of these URLs, we automatically attempted to fetch the DOM by sending a Web crawler to the URL. We also gathered the web server's IP address. Note that in some cases, the initial URL returned a redirection to another URL, in which case we recursively followed redirections until reaching the actual alleged phishing site. The data reported is the data collected after following these redirections.

We were unable to reach 2,237 of the 21,303 URLs fetched from PhishTank, for a variety of reasons including 350 URLs which returned a 400-level HTTP error code when accessed, 59 phishing sites that were identified has having been taken down, usually (but not always) by the host provider of these sites, and 1,828 attacks instances there were empty or failed to load in our crawler. We thus ended up with 19,066 valid URLs.

Detecting double entries in the dataset in not straightforward. They are due to the same attack instance being reported more that once to PhishTank. The corresponding URLs might have some variations to them (e.g. they may contain the email address of the targeted victim), and each time the page is loaded, it may also have some variations (e.g. an embedded tracking token). In order to detect such duplicates, and following the suggestions provided in [31], we have looked at the DOM of each alleged phishing attack instance, and removed all the spaces from these DOMs, as well as all the defaults values in every INPUT fields, replacing these values by empty strings. We then computed the SHA-1 hash of the resulting DOMs, to find phishing site with the same hash that were hosted on the same IP address. We call these "hash-duplicates". We need to distinguish a single attack reported twice from an attack relaunched by the attacker, and reported again. To attempt to distinguish between the two situations, we considered that hash-duplicates that were reported within a given number of days of each other were true hash duplicates (same attack instance reported twice to PhishTank), while the hash duplicates reported far enough apart from each other (and that were indeed live at the time of the report) were instance of attacks being relaunched by the attackers. We

[2]The proof that this is a mathematical distance can be found on `http://ssrg.site.uottawa.ca/phishingdata`.

[3]See `https://www.phishtank.com/stats.php` for PhishTank's latest stats.

	Number of URLs
Initial phishing list	21,303
Unreached phishing	2,237
Actual phishing list	19,066
Hash-duplicate	6,207
Phishing, no hash-duplicates	12,859
Legitimate sites list	24,800

Table 1: Phishing and legitimate databases.

chose 14 days for that period, which seemed to give enough time for the community to report an attack. This gave us 6,207 hash-duplicates in our database. A longer period did not significantly impact this results (20 days yields 6,441 hash-duplicates, and 30 days 6,650 hash-duplicates). In the following, we report our finding both with hash-duplicates included (19,066 URLs) and excluded (12,859 URLs).

In order to compare the result of our clustering algorithm on phishing sites and on legitimate sites, we have also gathered a database of non-phishing websites. For that purpose, we used the portal Alexa[4], which monitors web traffic and provides lists of the most popular websites. We fetched 24,800 legitimate sites, including 9,737 URLs from a series of Alexa free "top 500" most popular site by countries [2] and another 15,063 URLs randomly selected from the Alexa's top 100,000 to 460,697 web sites.

All of our datasets, URLs and DOMs for phishing and valid sites have been made available on `http://ssrg.site.uottawa.ca/phishingdata`. The data is recapped in Table 1.

3.2 Optimal Threshold

To choose the value of the clustering threshold, we have computed the threshold that will provide the minimal (i.e. the best) quality of clustering as defined in Section 2.2. In order to evaluate how stable this value would be overtime, we have decided to compute it over 4 different, increasing periods, using our phishing site database gathered from January 1st to March 1st, then from January 1st to May 1st, then from January 1st to July 1st and finally using the whole dataset, from January 1st to October 5th. The results are provided in Figure 2. In all four cases, we find the exact same threshold, 0.32, was minimizing the quality of clustering[5], which is at least an indication that this optimal threshold value can be relatively stable overtime.

3.3 Clustering Results

Our main results are shown in the Table 2: we find that our dataset of 19,066 phishing URLs only generates 3,796 different vectors, which in turn yield 2,831 clusters overall (that is, 2,831 attack classes). 1,216 of these clusters contain more than one URLs (we call these "flagged clusters" in the table). Of course, the set of 12,859 URLs with hash-duplicates removed generates exactly the same vectors and clusters by definition. In terms of hashes as defined in [31], we obtain 8,467 unique hashes, which means a collision rate of 34.2% with – or 55.6% without – hash duplicates, significantly lower than the 72.7% rate reported in 2011 [31]. The most important result is that 17,451 (resp.

[4] `http://www.alexa.com/`

[5] When several thresholds minimized QC, we selected the the smallest one.

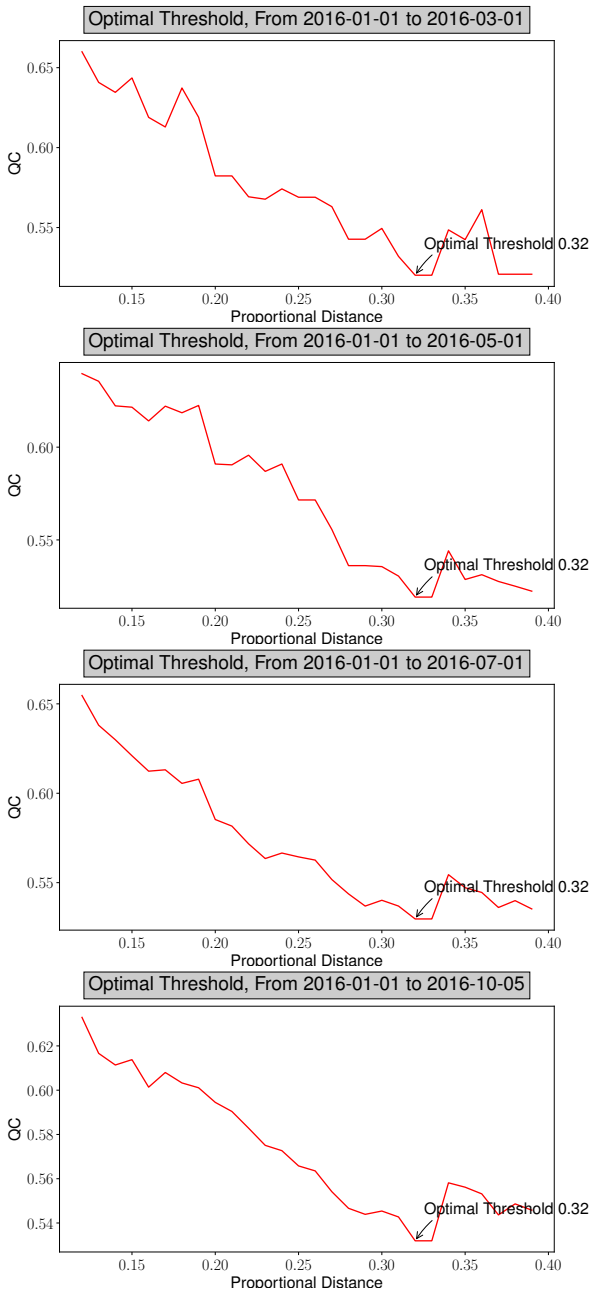


Figure 2: Optimal threshold computation over different time periods.

Phishing Sites Database	Size
# of phishing attack instances	19,066
# of phishing attack instances, no hash-duplicates	12,859
# of unique hash	8,467
# of unique vector	3,796
# of clusters (phishing attack classes)	2,831
# of flagged clusters	1,216

Legitimate Sites Database	Size
# of legitimate	24,800
# of unique hash	24,770
# of unique vector	24,655

Clustering Results	Size	Percentage
# of phishing in flagged clusters	17,451	91.53%
# of phishing, no hash-duplicates in flagged clusters	11,244	87.44%
# of legitimate in phishing clusters	20	0.08%

Table 2: Databases details (top) and clustering results (bottom).

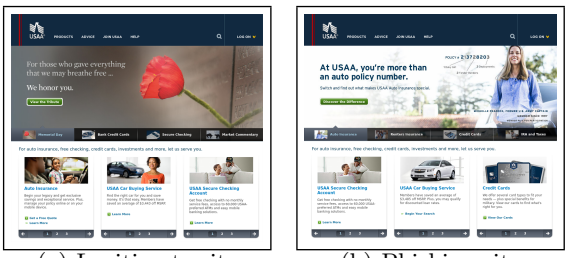
Targeted Brand	# of Phishing Attack Instances
Gmail	871
Dropbox	373
Dropbox	367
PayPal	323
Multiple mail login	272
Yahoo	261
Generic mail login	258
Dropbox	252
AOL	212
Gmail	179

Table 3: Targets of top 10 clusters

11,244) of the 19,066 (resp. 12,859) phishing URLs end up in a flagged cluster, which means that 91.53% (resp. 87.44%) of our entire phishing database is made of repeated attacks over our observation period. Figure 4 shows how this evolves as the database grows: a small window of only one month of recorded phishing attacks already yield good results, with over 75% of repeat attacks, and this threshold grows throughout the eight months duration of our experiment, so this suggests that the percentage will eventually be even higher. In Table 3, we show the targeted brands for the top 10 clusters after removing hash-duplicates. All but two have a single, well identified target. The fifth cluster targets most major email providers at once, while the seventh is a generic email login with no specific brand specified. Note that the same target appears more than once, because different attacks are targeting these same sites.

In our database of 24,800 legitimate sites, we do find a handful of hash collisions (0.12%) as well as some vector collisions (0.58%) which are due to having several country-specific versions of the same site in our database. Only 20 of these sites end up in one of the 2,831 phishing clusters, a false positive rate of only 0.08%. The main reason for these false positives is that some phishing sites are a copy of the legitimate site (See for example Figure 3 (a) and (b)). It is actually remarkable how infrequently this situation occurs. It shows that phishers do not tend to copy directly the site they are attacking. The reason for this is probably because it makes it particularly easy to detect and filter the attacks [8, 26]. In practice, we could easily reduce this false positive rate even further, e.g. by identifying the legitimate sites being targeted [20] and white-listing them, but this is out of the scope of this paper.

These results show that our clustering method is quite effective at finding clusters of phishing sites that are related to one another, and that our method yields a very low and sustainable false positive rate. This also shows that a simple, hash-based method such as the one suggested in [31] would in fact miss many of these replicas. We will discuss in detail the implications of these results in terms of phishing protection in Section 4.1, but it is important to understand that the fact that our method detected about 87.44% of the



(a) Legitimate site (b) Phishing site

Figure 3: Example of a false positive.

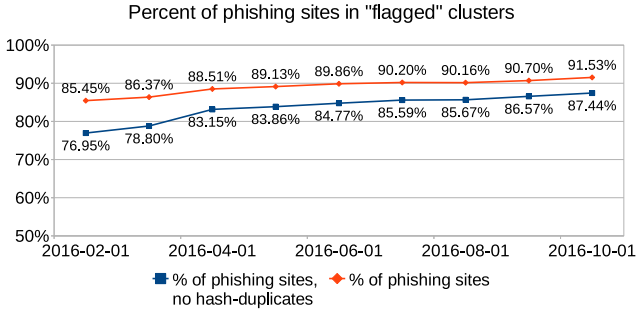


Figure 4: Percentage of duplicates over time.

phishing database does not mean at all a false negative rate of 12.56%. We are detecting replicas, and it is perfectly reasonable to assume that phishing sites in unflagged clusters do not have replicas (at least not in our database). The goal is not to detect a new phishing site, but a new variation of a known one. As we do not know precisely how many of these sites do have replicas in our database, we unfortunately cannot report the number of false negatives in our experiment, but this number is probably much lower than 12.56%.

Note that we ran similar experiments with proprietary live data from our partner IBM and we obtained the same results. This database is, however, confidential, so we cannot share it as we have done with our own database.

4. ANALYSIS AND DISCUSSION

4.1 Phishing Sites Detection Tool

To understand why our tool will play an important role in a phishing protection strategy despite the fact that it does not detect new phishing sites but rather new iterations of known ones, one has to consider how a typical realistic **corporate** in-depth anti-phishing strategy works. In such a context, a steady stream of tens of thousands, if not millions of daily URLs has to be processed to assess and blacklist phishing attacks in near-real-time. This is normally done in two steps: a first level of fast filtering is applied to remove the vast majority of the URLs, which are benign. The resulting, much-smaller set of potential phishing sites is then analyzed using time-consuming methods such as the ones reviewed in Section 5. This second step is slow, often requires an actual in-browser rendering of the web sites, and sometimes even requires human intervention. It takes seconds to minutes to process a site at that stage. The number of sites reaching this second step is thus of importance: for the defense to be effective, that number must be as low as possible.

The results of Section 3.3 show that today, about at least 90% of the attacks are repeats of previous attacks. It means that at the second step above, the vast majority of the time is spent re-confirming a known attack, whereas a technique such as ours can be used to automatically find many, if not all of these repeats, with almost no false positives. What is more, our clustering method can assess a site quickly: with a non-optimized code, on our database of almost 20,000 URLs we process a site in an average of 44 milliseconds.

Our tool is thus an effective and important intermediary step in the defense strategy: the list of potential attacks that have been flagged by the first step is then analyzed by our method. Based on our results, as it stands about 90% of the actual attacks in that list will automatically be removed (and the corresponding vectors immediately added to our clusters). A much reduced list is then passed on the next and final step, where new phishing attacks are evaluated using slower methods. When a new attack is confirmed at this last step, this information is immediately feed back to our system, and the corresponding cluster is created. From that point on, all the incoming variations of that new attack will be automatically removed and will not reach the last step. Adding that intermediary step can be the difference between a system that can cope with the kind of flow of incoming attacks that we see today and a system that cannot do it.

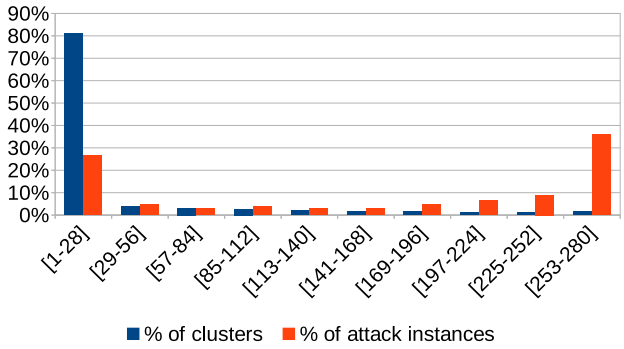


Figure 5: Lifespan of clusters and attack instance, in days.

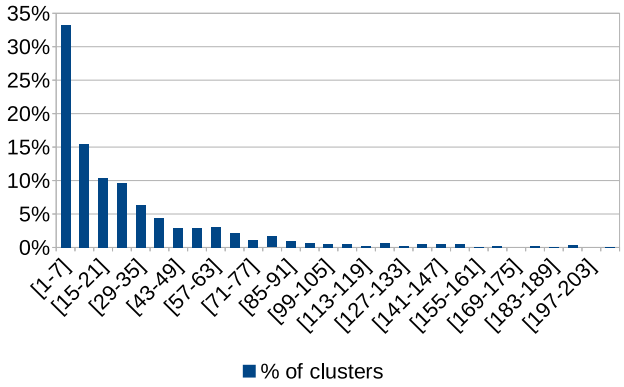


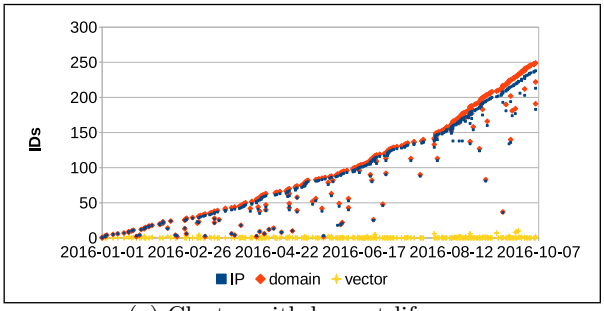
Figure 6: Average time between attack instances, in days.

Attackers can attempt to counter our tool by making it more difficult to detect re-publishing of an attack. Our current approach is not difficult to defeat in that regard, although it currently works well. However, the actual method used to detect resubmission can be adapted, and detecting similarities between DOMs is an active field of research, outside the scope of this paper. Even if the percentage of caught repeat attacks decreases, the ones detected will still be a net gain for the defense, and this detection step should therefore be part of the strategy as long as attackers repeat attacks. The ultimate goal of that step is to prevent attackers from relying on existing attacks, and force them to create new ones every time, driving up their cost and reducing their efficiency.

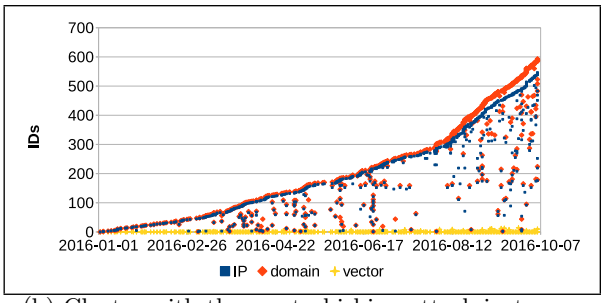
4.2 Lifespan of Phishing Sites

One cluster represents one phishing attack class – loosely speaking, one phishing site –, which is made of several attack instances (actual mailing campaigns trying to drive victims to the attack URL). We investigate here the lifespan of these attacks in our database, using the dataset without hash-duplicates. The lifespan is defined as the duration between the first and the last observed attack instance[6]. The main results are shown in Figure 5: the vast majority of the attack classes, around 80% of the clusters, are active for less than one month. The long-lasting attack classes, however, can stay active anywhere between 2 months to the entire

[6]Keep in mind that we cannot really tell when the attack observed at the beginning of our observation period actually started, and similarly at the end of the ten months.



(a) Cluster with longest lifespan



(b) Cluster with the most phishing attack instances

Figure 7: Number of vectors, IP addresses and domains used in sample clusters over time.

10 months of our observation timeframe. In general, the average lifespan of a cluster in our database is 25 days. If we look at the actual attack instances on the other hand, we have a bimodal distribution: the 80% of clusters that are active for less than one month only account for about 30% of the attack instances, while about 35% of the attack instances are on the other end of the spectrum, in clusters lasting 32 weeks and more. This shows that, contrary to what was previously assumed, actual attacks can last for a long time through many short lived attack instances (attack instances are blocked after about 10 hours according to [4]), and the majority of the instances are part of long-lasting attack classes. Figure 6 shows how often attack instances are observed in the 856 clusters in our database that do last more than one day: we see that these long-lasting clusters are active throughout their lifespan, with 33% having on average more that one attack instance per week, and almost half having on average more than one attack instance per couple of weeks.

In order to understand how these attacks evolve and survive such a long time, we look at the distribution of vectors, IP addresses and second-level domain names[7] over time. Figure 7 shows this for two sample clusters: one of the longest lasting ones in our database and the one with the most attack instances (871 phishing attacks). In this figure, numbers are cumulative, and reuse of a vector/IP/domain is indicated by another dot at the level at which that vector/IP/domain was first observed. As can be easily seen here, attackers do tend to change domains most often, with IP addresses a close second, but the actual vectors do not change nearly as much. If we look at the complete set of clusters lasting more than one day, the ratio vectors/IPs is 0.45 and the ratio vectors/domains is 0.36. This proves that what attackers do when they relaunch an attack is to find a different domain for the attack, and very often a different host for it. But they seldom invest time to modify substantially the site itself. This suggests that it is costlier for them to modify the attack, let alone to create a new one, when compared with using a new domain or a new host.

One of the salient points of this analysis is that a given phishing attack class can have a long lifespan, through thousands of "instances" of this attack. In this situation, it is not surprising that trying to protect against actual instances of an attack is not a good, effective strategy. It is a losing game since attackers can simply launch new instances at very low cost. What might be costlier for the attackers is to create new attacks. It is thus unfortunate that most of the prevention techniques in use today are really based on instance-detection. It is also what creates the illusion that the prevention is effective, whereas it is only effective at removing ephemeral attack instances. A much more effective strategy is to target the actual attack itself. In that context, our clustering method is a step in the right direction: once an instance is detected, a cluster is created for the actual attack, and new forthcoming instances will be detected. This suggests that Web browsers should be enhanced with solutions such as ours to be more effective in phishing prevention.

4.3 Linking Attacks Together

We have seen that attack instances can be clustered in such a way that all the instances of the same attack are in the same cluster, an attack class, showing few variations of the resulting DOM, and more variations in terms of domain names and ultimately IP addresses of the machine serving the attack. The next question we investigate is to try to find common features across clusters, and specifically shared IP addresses. If two attack classes are totally independent, they have no particular reasons to be hosted by the same IP address at any time. On the other hand, if the same group is behind two different attack classes (that is, attacks targeting two different sites, and thus creating two separated clusters in our system), it is quite possible that the same IP address will be used at different points in both attack classes, because that IP address is one of the possible hosts for that group.

The first observation is that only 5,267 IP addresses have been used to host the 19,066 phishing instances in our database (resp. 12,859, without hash-duplicates), proving that reuse is indeed occurring. For 1,807 of these IP addresses, the reuse occurred for different instances of the same attack class (in other words, these IP addresses appeared multiple times in the same cluster). This type of reuse is probably to be expected, since whatever group is behind an attack class knowns the IP addresses that have been used in the past, and might use them again for new instances of the attack class. However, 1,259 of the 5,267 IP addresses in our database (about 24%) appear in more than one cluster, and thus are reused across different attack classes. These 1,259 IP addresses alone appear in 1,289 clusters – that is about 46% of the cluster – and represent 9,564 (resp. 6,328, without hash-duplicates) attack instances – about 50% (resp. 49%) of the recorded attack instances. If we consider the entire span of

[7]When a country-code domain name is used, we look at the third-level domain names, so `www.example.com` is "example" while `www.CountryExample.co.uk` is "CountryExample".

the clusters involved in these shared IP addresses, we find that 15,247 (resp. 8,333) of the attack instances are linked to this situation – that is 80% (resp. 65%) of the attacks instances. This does suggest some commonalities between these different attack classes. It is of course difficult to be sure that shared IP addresses means a common attacker. One other possible explanation would be that independent attackers use the same "service" to acquire vulnerable machines, for example a fast-flux network service [15]. Finding this out will require further research, although it can already be pointed out that if we are looking at independent attackers sharing a common service, then taking down that particular service will also significantly disrupt the landscape of attacks, at least for the short term.

5. RELATED WORKS

There exists a significant body of academic work focusing on phishing sites detection. Two main approaches seem to be followed: on the one hand, one can identify a phishing site by comparing it to the legitimate site being targeted. On the other hand, one can instead find some intrinsic characteristics of the phishing sites themselves. Most of the published literature falls into one of these two categories, with some papers advocating for a combined strategy which may borrow ideas from both approaches.

In the category of the academic papers trying to detect phishing sites by comparing them with their legitimate target, Zhang et al. [33] propose a content-based method using a Term Frequency and Inverse Document Frequency (TF-IDF) analysis to identify the phishing target. The keywords extracted by the TF-IDF algorithm on a given page are submitted to search engines such as Google to find out the possible targets of the phishing attack. They report 89% true positive and 1% false positive on a database of 100 phishing sites and 100 legitimate sites. Rosiello et al. [28] propose a method based on the layout similarity between a phishing site and its target. Their analysis is based on the comparison of the DOM trees. Chen et al [7] also compare the visual similarity of the phishing site and its target, applying Gestalt theory on the image screenshot of the sites. Afroz et al. propose "PhishingZoo" which uses the profiles of the targeted sites appearances to detect phishing sites [1]. Wenyin et al. [20] propose a method using embedded links: a Web graph of links is built and the so-called "parasitic coefficient" is used to measure the relationship between the phishing sites and their targets. In this paper, a 99.67% true positive rate and a 0.5% false positive rate is reported on a database of 3,374 phishing sites and 1,200 legitimate sites. Ramesh et al. [26] also look at the hyperlink relationships between phishing sites and their targets. They report an accuracy rate of 98.95% and a false positive rate of 1.2% on a database of 10,005 phishing sites and 1,000 legitimate sites. Some authors also report good results while focusing on very specific characteristics of the sites, such as the favicon [12] or the site's logo [6].

In the category of the academic papers looking at identifying phishing site directly, Garera et al. [10] focus on the pattern sometimes found in the URLs of phishing sites. They were able to obtain a true positive rate of 95.8% and a false positive rate of 1.2% using a database of 1,245 phishing sites and 1,263 legitimate sites. Ludl et al. [21] created a decision tree based on 18 features taken from the URLs as well as from the site's HTML. They report a true positive rate of 83.09% and a false positive rate of 0.43%, using a database of 4,149 legitimate sites and 680 phishing sites. More recently, Xiang et al. [31] propose CANTINA+ based on a machine learning framework using a set of 15 features. They report a 99% true positive rate and 0.4% false positive rate, using a database of 8,118 phishing sites and 4,883 legitimate sites. Gastellier-Prevost et al. propose a method based on 20 heuristic tests on the site's URL and html [11]. Their experiments yield a true positive rate of 98.0% and a false positive rate of 2.0%, using 500 legitimate sites and 730 phishing sites. He et al. select 12 features to reach a 97.33% true positive rate and a 1.45% false positive rate with a database of 200 legitimate sites and 325 phishing sites [14]. Gowtham et al. do extend [14] to 15 features, and report a 99.8% true positive rate and a 0.4% false positive rate using 1,764 phishing sites and 700 legitimate ones [13].

Some researcher have also proposed an evaluation of popular anti-phishing tools. Zhang et al. [32] provided an analysis of ten popular anti-phishing tools using 516 legitimate sites and 200 phishing URLs, showing many weaknesses in these tools. More recently, Liang et al. [19] reverse-engineered the anti-phishing mechanism built into the Chrome browser and showed that they were able to evade detection after some minor modifications to the phishing sites. Prakash et al. use a blacklist of known phishing sites that is used as a basis to decide if a site is legitimate or not [25]. They report a true positive rate of more than 97% and a false negative rate of less than 5%, using a database of 32,000 phishing sites and 120,000 legitimate sites. Jain et al. propose a method using a white-list of legitimate sites accessed by individual users [17]. Their experiments show a true positive rate of 86.07 % and a false positive rate of 1.48%, with 1,120 phishing sites and 405 legitimate sites. A general survey on phishing protection is available in [18].

To the best of our knowledge, the only work similar to ours is the discussion provided in [31] regarding the duplicates in their database. As shown previously, our solution is very significantly more effective at flagging repeated publications of the same phishing attack.

6. CONCLUSION

In this paper, we have demonstrated that a common behavior of attackers using phishing sites is to repeatedly republish their attacks, using different domain names, different hosts and sometimes some minor modifications, probably as a way to get around the very short average lifetime of a given phishing site, about 10 hours according to [4]. This provides an opportunity to detect these "replicas" by comparing new attack instances with previously known ones. We have shown that methods based simply on hashes of the DOM, as suggested in [31], are not flexible enough and fail to detect many of the replicas. We have introduced a simple method which computes a vector counting the number of different HTML tags used in the DOM of the attacks. We have defined a distance between these vectors, and used that distance to cluster together sites with a distance lower than a given threshold. Our clustering method puts together vectors if their distance is closer than the threshold. These clusters are thus not centroid based, and accommodate long strings of successive modifications. Our method is deterministic, which means that the clusters can be iteratively built over time. Clusters coming from different phishing site databases can be combined and made publicly available.

Our tests show that a very large number of newly discovered phishing attack instances today would be caught by our method: about 90% of reported phishing sites are in fact replicas of already known phishing sites, and can be flagged immediately by our tool. What is more, the level of false positive it very low at 0.08%. This shows that the method presented here is an effective tool to add to a phishing defense strategy, as a preprocessing step to weed-out a large number of new phishing attack instances, dramatically reducing the load on the time-consuming back-end tools in charge of finding and confirming new attacks. This tool is meant as an addition to such a defense mechanism, and not a replacement of the existing tools. It is, however, a necessary improvement to prevent the time-consuming step of new attack detection from being overwhelmed by the sheer volume of new attack instances recorded today.

Although our clustering method is today quite effective, there is no doubt that as it gets deployed in corporate environments, attackers will attempt to defeat it by modifying their attack instances more thoroughly. The overall defense scheme will simply have to be adapted with more intelligent ways of detecting different instances of the same attack. The real novelty here is to compare an attack with other known attacks, instead of with the targeted legitimate site. It should be noted that the current body of work that detects a phishing attack by comparing it to its target could probably easily be adapted to compare attacks against other attacks, as we do here.

We have also shown that long-lasting attacks survive current prevention methods by re-launching very similar attacks on new domains and on new servers, possibly after some minor variation of the DOM. Current prevention methods aim at identifying and black-listing attack instances, instead of the attack classes. This is a losing strategy, easily escaped by the attackers, and a strategy that provides a false sense of efficiency. Instead, moving forward, anti-phishing defense strategies should focus on the actual attack classes behind the attack instances, forcing attackers to invest more between each iteration of their attacks. The method presented here is one such strategy. Once an attack instance is identified, the entire attack class can be neutralized.

Finally, we show that a large percentage of the phishing attacks recorded today do share some common resources. This suggest that it might be possible to significantly lower the number of attacks by preventing some active groups from operating, or by taking down some popular "services". Our data and our analyses can be used as a starting point to identify such groups or such services. Our entire dataset is freely available for further research on `http://ssrg.site.uottawa.ca/phishingdata`.

Acknowledgements

This work is supported by the IBM® Center for Advanced Studies and the Natural Sciences and Engineering Research Council of Canada (NSERC).

7. REFERENCES

[1] S. Afroz and R. Greenstadt. Phishzoo: Detecting phishing websites by looking at them. In *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*, pages 368–375. IEEE, 2011.

[2] Alexa. Top 500 Sites in Each Country. `http://www.alexa.com/topsites/countries`.

[3] Anti-Phishing Working Group. Anti-phishing working group home page. `http://www.antiphishing.org/`.

[4] Anti-Phishing Working Group. Global Phishing Report 2H 2014. `http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2H_2014.pdf`.

[5] Anti-Phishing Working Group. Phishing Activity Trends Report 1st Quarter in 2016. `http://docs.apwg.org/reports/apwg_trends_report_q1_2016.pdf`.

[6] E. H. Chang, K. L. Chiew, S. N. Sze, and W. K. Tiong. Phishing detection via identification of website identity. In *2013 International Conference on IT Convergence and Security, ICITCS 2013*, pages 1–4. IEEE, 2013.

[7] T.-C. Chen, S. Dick, and J. Miller. Detecting visually similar web pages: Application to phishing detection. *ACM Trans. Internet Technol.*, 10(2):5:1–5:38, June 2010.

[8] R. Dhamija and J. D. Tygar. The battle against phishing. In *Proceedings of the 2005 symposium on Usable privacy and security - SOUPS '05*, number July, pages 77–88, New York, NY, 2005. ACM.

[9] FBI. FBI Says $2.3 Billion Lost to Fake CEO Phishing Scams. `https://www.fbi.gov/phoenix/press-releases/2016/fbi-warns-of-dramatic-increase-in-business-e-mail-scams`.

[10] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring Malcode - WORM '07*, pages 1–8, New York, NY, 2007. ACM.

[11] S. Gastellier-Prevost, G. G. Granadillo, and M. Laurent. Decisive heuristics to differentiate legitimate from phishing sites. In *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, pages 1–9. IEEE, 2011.

[12] G.-G. Geng, X.-D. Lee, W. Wang, and S.-S. Tseng. Favicon - a clue to phishing sites detection. In *eCrime Researchers Summit (eCRS), 2013*, pages 1–10, Sept 2013.

[13] R. Gowtham and I. Krishnamurthi. A comprehensive and efficacious architecture for detecting phishing webpages. *Computers & Security*, 40:23–37, 2014.

[14] M. He, S.-J. Horng, P. Fan, M. K. Khan, R.-S. Run, J.-L. Lai, R.-J. Chen, and A. Sutanto. An efficient phishing webpage detector. *Expert Systems with Applications*, 38(10):12018–12027, 2011.

[15] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detecting fast-flux service networks. In *NDSS*, 2008.

[16] IBM. Trusteer is now part of ibm. `https://www-01.ibm.com/software/security/trusteer/`.

[17] A. K. Jain and B. Gupta. A novel approach to protect against phishing attacks at client side using auto-updated white-list. *EURASIP Journal on Information Security*, 2016(1):1–11, 2016.

[18] M. Khonji, Y. Iraqi, and A. Jones. Phishing detection: A literature survey. *IEEE Communications Surveys Tutorials*, 15(4):2091–2121, Fourth 2013.

[19] B. Liang, M. Su, W. You, W. Shi, and G. Yang. Cracking Classifiers for Evasion : A Case Study on the Google ' s Phishing Pages Filter. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*, pages 345–356, Montréal, QB, 2016.

[20] W. Liu, G. Liu, B. Qiu, and X. Quan. Antiphishing through Phishing Target Discovery. *IEEE Internet Computing*, 16(2):52–61, 2012.

[21] C. Ludl, S. McAllister, E. Kirda, and C. Kruegel. On the Effectiveness of Techniques to Detect Phishing Sites. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 20–39. Springer Berlin Heidelberg, 2007.

[22] McAfee. Phishing protection. `https://home.mcafee.com/advicecenter/?id=ad_phishing&ctst=1`.

[23] National Institute of Standards and Technology (NIST). Secure Hash Standard. Federal Information Processing Standards Publication 180-1., 1995.

[24] Panda Security. Phishing: personal data theft. `http://www.pandasecurity.com/canada-eng/homeusers/security-info/cybercrime/phishing/`.

[25] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta. Phishnet: predictive blacklisting to detect phishing attacks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5. IEEE, 2010.

[26] G. Ramesh, I. Krishnamurthi, and K. S. S. Kumar. An efficacious method for detecting phishing webpages through target domain identification. *Decision Support Systems*, 61(1):12–22, 2014.

[27] K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19(4):639–668, Dec 2011.

[28] A. P. E. Rosiello, E. Kirda, C. Kruegel, and F. Ferrandi. A layout-similarity-based approach for detecting phishing pages. In *Proceedings of the 3rd International Conference on Security and Privacy in Communication Networks, SecureComm*, pages 454–463, Nice, 2007.

[29] C. Whittaker, B. Ryner, and M. Nazif. Large-Scale Automatic Classification of Phishing Pages. In *In Proceedings of the Network & Distributed System Security Symposium (NDSS 2010)*, pages 1–14, San Diego, CA, 2010.

[30] WWW. HTML Tag Set. `https://www.w3.org/TR/html-markup/elements.html`.

[31] G. Xiang, J. Hong, C. P. Rose, and L. Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Trans. Inf. Syst. Secur.*, 14(2):21:1–21:28, Sept. 2011.

[32] Y. Zhang, S. Egelman, L. Cranor, and J. Hong. Phinding Phish: Evaluating Anti-Phishing Tools. In *In Proceedings of the Network & Distributed System Security Symposium (NDSS 2007)*, pages 1–16, San Diego, CA, 2007.

[33] Y. Zhang, J. Hong, and C. Lorrie. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th International Conference on World Wide Web*, pages 639–648, Banff, AB, 2007.