Phishing Attacks Modifications and Evolutions

Qian Cui[1(✉)], Guy-Vincent Jourdan[1], Gregor V. Bochmann[1], Iosif-Viorel Onut[2], and Jason Flood[3]

[1] Faculty of Engineering, University of Ottawa, Ottawa, Canada
{qcui,GuyVincent.Jourdan,Bochmann}@uottawa.ca
[2] IBM Centre for Advanced Studies, Ottawa, Canada
vioonut@ca.ibm.com
[3] IBM Security Data Matrices, Dublin, Ireland
FLOODJAS@ie.ibm.com

Abstract. So-called "phishing attacks" are attacks in which phishing sites are disguised as legitimate websites in order to steal sensitive information.

Our previous research [1] showed that phishing attacks tend to be relaunched many times, after sometimes small modifications. In this paper, we look into the details of these modifications and their evolution over time. We propose a model called the "*Semi-Complete Linkage*" (*SCL*) graph to perform our evaluation, and we show that unlike usual software, phishing attacks tend to be derived from a small set of master versions, and even the most active attacks in our database only go through a couple of iterations on average over their lifespan.

We also show that phishing attacks tend to evolve independently from one another, without much cross-coordination.

Keywords: Phishing attacks · Attacks modifications
Evolution graph

1 Introduction

In 2016, the number of phishing attacks reached an all-time high, with at least 255,000 unique attack instances [2]. Unfortunately, the trend only worsened, and there are already over 580,000 unique attack instances reported up to the *3rd Quarter* of 2017 [3,4]. This growth occurred despite the public's increasing awareness and widespread tools that are used to combat these attacks. For example, browsers such as Google Chrome, FireFox, Opera and Safari all use *Google Safe Browsing*[1] to provide to their users some level of built-in protection from phishing attacks. Microsoft Internet Explorer and Edge browsers also include a similar built-in defence mechanism, called *SmartScreen*[2].

[1] https://safebrowsing.google.com/.
[2] https://support.microsoft.com/en-us/help/17443/windows-internet-explorer-smartscreen-filter-faq.

The majority of the literature on phishing attacks focuses on detection, e.g. by using machine learning to train a detection model, or by using the reputation of the domains hosting the attacks, or by performing visual comparisons between the phishing site and its target. However, Phishing is still very active; for instance, a FBI report estimates that there were over 25,000 victims in 2017 for a total loss of almost 30 millions US dollars in the USA alone [5]. Our inability to stop the onslaught of attacks shows that we need to go beyond merely detecting attacks. We need to better understand why phishing attacks are growing so fast and how phishers achieve this.

In our previous research [1], we showed that most phishing attacks are not created from scratch, and they are actually duplicates or small variations of previous attacks. Our experiments showed that over 90% of the phishing attacks in our database were close duplicates of other attacks in the same database. This created *clusters* of similar attacks.

In this paper, we explore the variations that are seen in these phishing attack clusters, when the attacks are not the exact replica of another attack, and some small modifications were performed over time. We try to answer the following questions: (1) What reasons push phishers to create variations instead of simply reusing exact replicas? (2) How are the attack typically modified when variations are created? (3) Can we see common trends behind these modifications across seemingly unrelated phishing attacks, or are the modifications specific to each attack cluster? Our ability to answer these questions will further enhance our understanding of the phishing ecosystem and it will help with combating the problem more effectively.

In order to answer these questions, we are using a database of over 54,000 verified phishing attack instances collected between January 2016 and October 2017. This represents a small sampling of the total number of attacks (for instance, the Anti-Phishing Working Group reports about 2 million attacks during that same period[3]). Moreover, our dataset is mostly made of attacks occurring in North America and Europe. However, the model and analysis we proposed could be applied to a larger dataset. In order to explore the evolution of phishing attacks modifications over time, we propose a new cluster structure based on what we call a *semi-complete linkage graph* (SCL). We find that most attacks are derived from a small set of master versions, with few consecutive updates and long shelf life. Moreover, we find that new variations created from a given attack usually uses patterns specific to that attack. All of the data used in this research is publicly available at http://ssrg.site.uottawa.ca/phishing_variation/.

The paper is organized as follows: In Sect. 2, we introduce various mathematical concepts that we use in our analysis. Then in Sect. 3, we present the basic results of our experiments. We discuss these results and provide a detailed analysis in Sect. 4. We provide an overview of the literature in Sect. 5 before the conclusion in Sect. 6.

[3] https://www.antiphishing.org/resources/apwg-reports/.

2 Phishing Attacks Clustering

In order to analyze phishing attack modifications over time, we must first group together attacks that are related and share similar features. In this section, we introduce and discuss the mathematical concepts and algorithms that we used to cluster these phishing attacks.

2.1 DOM Similarity Between Phishing Attacks

The Document Object Model (DOM) is a tree structure in which each node represents one HTML element of a web page. In previous research, a variety of techniques have been used to compare the similarity of DOMs [6]. The *Tree Edit Distance* (TED) is one of the most popular metrics for measuring the structural similarity between two DOMs. It represents the minimal number of operations (adding, removing and replacing) to convert one document into the other. However, the complexity of the best TED algorithm to date, *AP-TED* [7], is still $O(n^3)$, where n is the number of nodes in the DOM. To reduce the complexity of computing TED, some approaches based on fuzzy hash [8] or information retrieval [9,10] have been proposed. These methods are however limited and cannot be used to find out the specific differences between the trees. Therefore, they cannot be used to perform an analysis of the modifications between the trees. Our previous research [1] proposed a trade-off method, introducing *tag vectors* to compare the similarity of the DOM of phishing attacks with complexity $O(n)$. A *tag vector* is based on an ordered list of 107 possible HTML tags. The *tag vector* of a given DOM is a vector of size 107, and each element of the vector is the number of occurrences of the corresponding HTML tag in the DOM. This method does not capture the structure of the DOM, which may lead to the grouping of DOMs that have different structures but have a similar number of each type of HTML tags. However, we have looked at the trees of DOMs that have the same tag vectors in our database. We found that only 521 of these DOMs (or 0.95% of our phishing attack database) have the same tag vector but a different DOM tree. It is thus safe to use tag vectors in our case.

To compare the distance between tag vectors, in [1] we proposed to use the *Proportional Distance* (PD), which divides the Hamming Distance of the vectors by the number of tags that appear in at least one of the two DOMs. Formally, given two non-null tag vectors t_1 and t_2, the proportional distance between t_1 and t_2 is given as:

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

$$\text{where } D(x,y) = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{otherwise} \end{cases} \text{ and } L(x,y) = \begin{cases} 1 & \text{if } x \neq 0 \text{ OR } y \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

The proportional distance PD as defined in [1] does not emphasize on the "amount" of differences between each HTML tag, and simply focuses on whether the number of tags is the same. For example, the vector $t_1 = \{1, 2, 5, 6\}$ and $t_2 = \{109, 2, 5, 6\}$ both have the same distance to the vector $t_3 = \{2, 2, 5, 6\}$, that is, $PD(t_1, t_3) = PD(t_2, t_3)$. For our study, we would like to capture the fact that t_2 is more different from t_3 than t_1 is. Therefore, we define a new distance, called the *Weighted Proportional Distance* (WPD)[4] to compare the similarity of phishing attack instances. Instead of using the Hamming Distance as the numerator, we use the sum of the *Weighted Differences* (WD), defined by the following formula:

$$WD(t_1, t_2) = \sum_{i=1}^{n} \frac{|t_1[i] - t_2[i]|}{max(t_1[i], t_2[i])}$$

whereas the value of D for a given tag was boolean (0 or 1), for tags that are used in both vectors, WD will be in the range $[0, 1)$. The larger the difference between the number of tag, the larger WD.

We define S as follows:

$$S(t_1, t_2) = \sum_{i=1}^{n} EQU(t_1[i], t_2[i])$$

$$\text{where } EQU(t_1[i], t_2[i]) = \begin{cases} 1 & \text{if } t_1[i] = t_2[i] \text{ AND } t_1[i] \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Finally, the *Weighted Proportional Distance* (WPD) is defined as follows:

$$WPD(t_1, t_2) = \frac{WD(t_1, t_2)}{WD(t_1, t_2) + S(t_1, t_2)}$$

In the rest of the paper, we use WPD as the distance between our tag vectors. It should be noted that other distance metrics could be used with probably similar results. We used WPD because it is fast to compute and works well for our goal.

2.2 Optimal Threshold

In order to create clusters of similar attacks, we need to find out a good threshold for grouping vectors together. If the distance between two vectors is less than this threshold, they are considered similar and grouped into the same cluster. Otherwise, they are separated into different clusters. The *optimal* threshold is one that yields clusters that are fairly compact inside while the distance between clusters is large. Before computing this optimal threshold for our database, we first must define how vectors are connected inside each cluster.

[4] For consistency with the name PD, we call this value the "Weighted" PD. However, it should be noted that WPD is *not* a distance in the mathematical sense of it.

2.3 Intra-cluster Vectors Connections

There are at least two common models that are widely used when it comes to intra-cluster connections: (1) Single-linkage, where each node inside the cluster is connected to at most one parent, creating a minimal spanning tree over the elements of the cluster, or (2) Complete-linkage, where a complete graph is created between all the elements of the clusters. However, neither of these two models can accurately capture what we are trying to do here, that is, capture the evolution of the elements inside a cluster. A good model should keep a connection between the elements of a series of modifications done to a given attack (and some of these elements may end up being fairly far apart after a long series of modifications), but it should also capture the fact that some elements are at a very small distance from each other within the cluster. This idea is illustrated on Fig. 1. Vectors a, b, c and d are close to one another, meaning that there is little variation between these four vectors. On the other hand, Vector e, while still part of the same cluster, is actually relatively "far" from these first four vectors, and is only linked to them through a long series of small variations.

To capture these series of modifications done to the phishing attacks inside a cluster, we proposed to use a *Semi-Complete Linkage* (SCL) model. Specifically, for any pair of tag vectors t_i and t_j in the same cluster, where $i \neq j$, we have an edge $E(t_i, t_j) \in SCL$ if and only if $WPD(t_1, t_2) \leqslant OPT$, where OPT is the optimal threshold for tag vector clusters defined in Sect. 2.4. A simple way to see this model is that inside a cluster, vectors that are "similar" are linked together. This model is an intermediate model between the spanning tree and the complete graph.

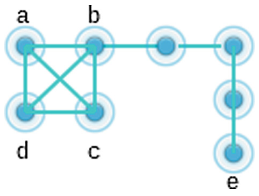


Fig. 1. An illustration of a *Semi-Complete Linkage* graph.

2.4 Quality of Clustering

We now explain how we define the *quality of clustering* and how we will compute the optimal threshold. We define $Min(C_i, C_j)$ to be the minimal distance between two clusters, which is defined as the minimum distance that can be found between two vectors, one in C_i and one in C_j. That is:

$$Min(C_i, C_j) = min(\{WPD(x, y) | \forall x \in C_i, \forall y \in C_j\})$$

As discussed in Sect. 2.3, we use the SCL model to capture the connections inside tag vector clusters. Thus, we define the **quality** of vector clusters with the following formula, which computes the total distance inside the clusters, and divides it by the distance between clusters. We will experimentally try different threshold to find one that minimizes this formula. The formula, which only includes the clusters that have more than one element, is as follows:

$$\frac{\frac{1}{k}\sum_{i=1}^{k}\frac{1}{|E_i|}\sum_{j=1}^{|E_i|}\{WPD(x,y)|E_j(x,y)\in SCL_i\}}{min\{Min(C_i,C_j)|i\neq j, 1\leqslant i,j\leqslant k\}}$$

where k is the number of clusters having more than one element, $E(x,y)$ is the edge between x and y in the SCL graph, C_i is the i^{th} cluster with more than one element, SCL_i is the SCL for C_i and $|E_i|$ is the number of edges in SCL_i.

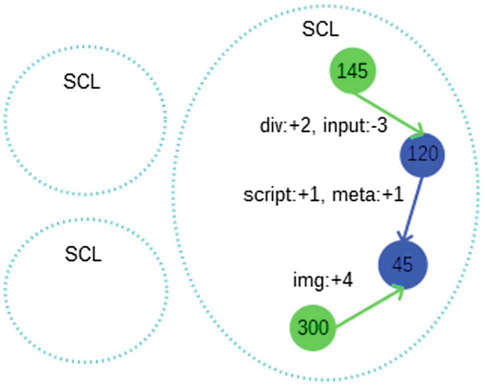


Fig. 2. Example of phishing attacks modifications graph

2.5 Phishing Attacks Modifications Graph

To analyze the evolution of phishing attacks, we computed the SCL model for each tag vector cluster, as illustrated in Fig. 2. Each node represents a unique tag vector, and the nodes label shows the number of phishing attack instances using this vector. The directed edge $E(x,y)$ captures an evolution from vector x to vector y, that is, a modification made to the corresponding attack, which transforms the original attack (which has vector x) into a slightly different attack (which has vector y). The text on the edge provides the details of the modifications. For example, an edge with the label "div:+2, input:-3" should be interpreted as meaning that two *div* tags where added to the attack and three *input* tags where removed in the creation of the new variation of the attack. The direction of the edge is determined by the reported date of the two connected vectors; the edge flows from the earlier attack to the later attack. Since several attacks will have the same vector, we consider that the "reported date" of a vector is the date at which we learned of the first attack that produced this vector.

As a consequence of this definition, a source node of the graph, that is, a node that has an in-degree of zero, is the earliest reported attack instance in our data source from this series of modifications. We color these nodes in green. Node that are variations of previously reported attacks have a positive in-degree and are shown in blue in our graph.

3 Experiments

3.1 Phishing Sites Database

We have compiled our phishing database by collecting the URLs of phishing-attack instances from the community-driven portal PhishTank[5] and the enterprise security analysis platform IBM X-Force[6]. A total of 54,575 "verified" phishing sites were collected by fetching the daily archive from PhishTank between January 1st, 2016 and October 31st, 2017 and from IBM X-Force between June 12th, 2017 and October 31st, 2017. For each phishing site, we fetched the DOM, the first URL (the reported one), the final URL (which is different from the first URL only when redirection has been used by the attacker), and a screenshot of the final page. To compare the performance of our new model with the model proposed in [1], we used a database of 24,800 legitimate sites found on Alexa[7], made of 9,737 URLs coming from the lists of "top 500" most popular sites by countries [11] and another 15,063 URLs randomly selected from the Alexa's top 100,000 to 460,697 web sites. The list of URL is available on http://ssrg.site.uottawa.ca/phishingdata/.

3.2 Vectors and Clustering Results

To compute the set of tag vectors, as was done in [1], we used the complete set of HTML elements provided by the World Wide Web Consortium [12], and removed the common tags <body>, <head> and <html>. That gave us a corpus of 107 unique tags. We then counted the number of occurrences of each tag in each DOM and used these number to create integer vectors of 107 features. We obtained 8,397 *unique* tag vectors out of the DOMs of our 54,575 phishing attack instances.

In order to compare the performance of our model to the one proposed in [1], we first trained both models with the same phishing database and computed the phishing attacks clusters and related optimal threshold. We then used our database of legitimate sites to see how many false positives each model yields.

As shown in Table 1, the SCL model has a smaller optimal threshold, but captures many more attacks than our previous models (only 3,869 undetected attacks, compared to 4,351 with the previous model). There was however a slight increase in the false positive rate, which remains very low at 0.26%. This shows that the model proposed here is more efficient than the one proposed in [1] if the aim is to detect phishing attack replicas. Similar to [1], the false negative rate is unknown since we don't know how many of the 3,869 unflagged attacks have a replica in our database.

[5] https://www.phishtank.com/.
[6] https://exchange.xforce.ibmcloud.com/.
[7] https://www.alexa.com/.

Table 1. Vector and clustering results for both models. "Flagged" cluster have more than one element, and the corresponding attacks are detected.

	SCL Model	Model of [1]
Optimal threshold	0.24	0.33
# of vectors	8,400	8,400
# of multiple-element clusters ("*flagged*")	941	908
# of single-element clusters	3,869	4351
# of phishing sites in flagged clusters	50,706 (92.9%)	50,224 (92.03%)
# of "similar" legitimate sites (false positive)	65 (0.26%)	58 (0.23%)

4 Analysis of the Modifications Seen in Phishing Attacks

4.1 Who Made Modifications, Phishers or Hosts?

One possible explanation for the modifications we see on different instances of the same attack is that the attack was not actually modified by the attacker, but by the hosting server, which is automatically injecting some html into the pages, e.g. some Google Analytics tracking links, some WordPress plugins or some other Javascript libraries. Since a given attack will be hosted on a range of servers, these modifications would be misinterpreted as modifications to the attack itself.

To verify this, we compared the DOM of the phishing attacks to the DOM of homepages of the server hosting these attacks. We removed all the "blanks" (including \t \r \n \f \v) from both DOMs, and we then extracted the content that was common between the two DOMs. This content could have been coming from the hosting server, and not from the attack itself. We did this for all the attack instances in our database for which the host homepage could be reached and had a different tag vector from the attack[8].

We were able to collect the DOMs of 14,584 such homepages[9]. Of these, 2,566 had some common content with the hosted attacks. A closer look at the tags involved in these common contents showed that the tag <**meta**> was involved in 2,280 of these cases, which is not surprising since <**meta**> is used for information such as encoding, page size etc., information usually set by the hosting server. The tag <**script**> was a very distant second present in only 96 cases. This shows that the tag <**meta**> is the only tag for which the hosting server can really impact our results. Therefore, we decided to remove that tag altogether from our tag vectors. Redoing the experiment of Sect. 3.2 without that tag, we find the same optimal threshold (0.24), and end up with 8,290 tag vectors distributed across 913 flagged clusters (cluster with at least 2 vectors) and 3,912 single-vector clusters. The false positive rate drops to 0.25%, as a couple of

[8] This excludes attacks that are located right at the homepage of the hosting server.
[9] Many hosting servers were not reachable anymore by the time we did this experiment.

legitimate sites are now correctly flagged. Out of an abundance of caution, we used that updated model in the analysis presented in the next sections.

4.2 Clusters Sample Selection

We applied the SCL model discussed in Sect. 2.3 to our 913 flagged clusters. We observed that there are several clusters with very few edges in their SCL graph, meaning that for these clusters, our database does not contain many variations of the corresponding attacks. Table 2 shows a detailed distribution of sizes of the SCL graphs. As already pointed out in [1], a small minority of the clusters cover the vast majority of the attacks. In this case, only 46.88% of the clusters have a SCL graph with two or more edges, but they contain more than 75% of the phishing attack instances. For our study, we selected the clusters having a SCL graph with 30 or more edges because they capture the majority of the phishing attack instances (52%) and they contain enough variations of the attacks to study their evolution over time.

Table 2. Number of edges and pages distribution among clusters

# of edges in the cluster	# of clusters (%-tage of total)	# of pages covered (%-tage of total)	# of edges covered
≥2	428 (46.88%)	41,229 (75.55%)	18,636
≥3	394 (43.15%)	40,579 (74.35%)	18,568
≥4	258 (28.26%)	38,059 (69.74%)	18,160
≥5	243 (26.62%)	37,321 (68.38%)	18,100
≥10	150 (16.43%)	34,539 (63.29%)	17,504
≥15	107 (11.72%)	31,638 (57.97%)	17,043
≥20	88 (9.64%)	30,797 (56.43%)	16,732
≥30	62 (6.79%)	28,801 (52.77%)	16,113
≥40	47 (5.15%)	26,298 (48.19%)	15,591
≥50	42 (4.60%)	25,306 (46.37%)	15,381

4.3 Analysis of Master Vectors

As explained in Sect. 2.5, the orientation of the edges in the SCL graphs is determined by the reported date of the DOMs creating the tag vectors, from the earlier one to the later one. We call a tag vector of in-degree zero in the SCL graph a *master vector*. Master vector represents one of the initial versions of the attack in our database. Of course, each cluster contains at least one master vector (the earliest reported vector in that cluster), but they can have several ones when the distance between the vectors is too large for them to be connected in the SCL graph. Having several master vectors in a cluster means that some attacks have been substantially modified at once, or that we are missing to intermediate steps in our database. Each non-master vector can be reached from at least one of the master vectors in the cluster. Those master vectors provide a view of the initial attacks and the non-master vectors give a view of how they evolved over time. Figure 3 shows the SCL graphs of the two largest clusters in our database (master vectors in green, non-master vectors in blue). We can see that there are far fewer master vectors than non-master ones, indicating that the majority of attacks in these clusters evolved from the original vectors.

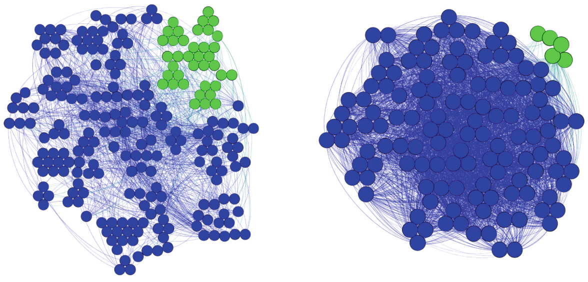

(a) SCL graph of cluster 0 (b) SCL graph of cluster 1

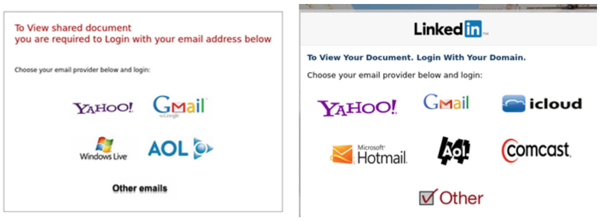
Fig. 3. Examples of SCL graphs

Table 3 provides an overview of the results for all 62 clusters: overall, there are 190 (10.47%) master vectors, covering around 35% of the attack instances. This shows that the master vectors are often reused to relaunch the attacks. Moreover, 34 clusters (54.84%) have two or more master vectors, suggesting several initial versions of the attack which were later merged through a series of updates.

Table 3. Overview of master/non-master vectors in the 62 largest clusters.

# of clusters	62
# of vectors	1814
# of attack instances	28,455
# of master vectors	190 (10.47%)
# of attack instances in master vectors	9,855 (34.22%)
# of clusters with two or more master vectors	34 (54.84%)
# of clusters with only one master vector	28 (45.16%)

By manually inspecting the DOMs of master vectors, we found that master vectors can be grouped into three categories: (1) Different initial versions of the attack by attackers, with enough changes to push the distance beyond the threshold. It could be the case that the target is modified or that several new features are released at once. Figure 4(a) shows such an example. (2) Different steps of the same attack. Some attacks go through several steps as they attempt to gather additional information from the victim. For example, in Fig. 4(b), a first step is used to capture login information, and if it is provided, a second step follows in which credit card details are requested. These different steps are recognized as belonging to the same attack, but the difference between them is too large for the threshold and there is no directed path between them in the *SCL* graph. (3) Copies of different versions of the target site. As shown in Fig. 4(c), sometimes the master vectors are essentially copies of the target sites taken at different times. The target site was modified, so the corresponding attack instances do not initially match. It is also possible that in some cases


To View shared document
you are required to Login with your email address below

Choose your email provider below and login:

YAHOO! Gmail Windows Live AOL

Other emails

LinkedIn

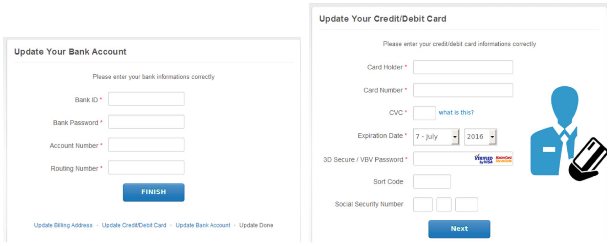
To View Your Document. Login With Your Domain.

Choose your email provider below and login:

YAHOO! Gmail icloud Hotmail Aol comcast

Other

(a) Different versions developed by phishers


Update Your Bank Account

Please enter your bank informations correctly

Bank ID *
Bank Password *
Account Number *
Routing Number *

FINISH

Update Billing Address · Update Credit/Debit Card · Update Bank Account · Update Done

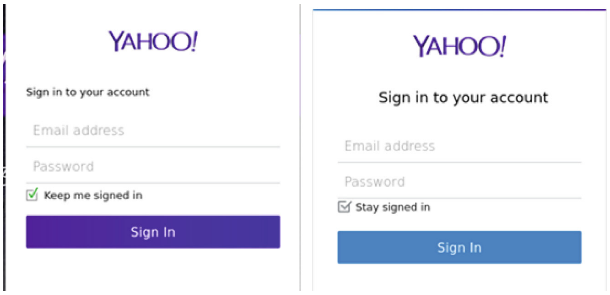
Update Your Credit/Debit Card

Please enter your credit/debit card informations correctly

Card Holder *
Card Number *
CVC * what is this?
Expiration Date * 7 - July 2016
3D Secure / VBV Password *
Sort Code
Social Security Number

Next

(b) Different steps of the same attacks


YAHOO!

Sign in to your account

Email address
Password
Keep me signed in

Sign In

YAHOO!

Sign in to your account

Email address
Password
Stay signed in

Sign In

(c) Different versions copied from legitimate sites (Yahoo login page, circa 2015 and 2016)

Fig. 4. Examples of master vectors

our database is missing an even earlier version of the attack that would yield an initial, sole master vector.

4.4 Analysis of Variation History

In order to analyze the evolution of the attacks in our database, we first introduce a few definitions. As explained before, every non-master vector v has at least one directed path in SCL from a master vector to v.

We call the *Evolution Path of* v (EP_v) the directed path from a master vector to v for which the sum of Weighted Proportional Distances of the edges along the path is minimal. In other words, EP_v is the directed path from one of the master vector to v for which the amount of transformation was the smallest.

For a non-master vector v and its evolution path $EP_v = [t_0, t_1, \ldots, t_{k-1}, t_k = v]$, we have the following definitions:

1. The *Path Distance* (PD_v) is the sum of the weighted proportional distance of the edges along the evolution path EP_v. It represents an evaluation of the "amount" of difference between v and its master vector.

$$PD_v = \sum_{i=0}^{k-1}(WPD(t_i, t_{i+1}))$$

2. The *Evolution Distance* (ED_v) is the average weighted proportional distance of edges along the evolution path EP_v. It represents the average "amount" of difference in each modification. Formally, $ED_v = PD_v/k$.
3. The *Variation Lifespan* (VL_v) is the time difference between the reported date of v and the reported date of its master vector. It represents the complete length of time during which this attack has been actively modified. If $T_{report}(t_i)$ is the reporting date of vector t_i, we have

$$VL_v = T_{report}(t_k) - T_{report}(t_0)$$

4. The *Update Interval* (UI_v), is the average of the time difference between consecutive vectors along the evolution path EP_v. It represents how often modifications are being deployed. Formally, $UI_v = VL_v/k$.

Table 4. Analysis of the evolution paths in our database.

# of evolution paths	1,230
Average *Path Distance*	0.1719
Average *Evolution Distance*	0.111
Average *Variation Lifespan*	267 days
Average *Update Interval*	186 days

Table 4 provides the average values of these attributes for all evolution paths in the selected 62 clusters. To compute these values, we have not included Evolution Paths that are included into other, longer evolution paths. The results show that in general, the attacks are only modified once every six months (186 days) and that the modifications are usually not drastic (the average WPD between these modifications is 0.111). We also see that average path distance is low, only 0.1719. Consequently, the average length of the evolution paths is only $0.1719/0.111 < 2$, less than two edges. This indicates that attackers usually do not maintain long evolution paths to create lots of variations over time. Instead, they tend to re-create new variations from the same master vectors over and over. We also find that each variation tends to stay active for a long time, around nine months (267 days).

In conclusion, we see that most phishing attack modifications are derived from a small set of master versions. Each of these modifications tend to be reused *as is* for an extended period of time. This behavior matches the "crimeware-as-a-service" model proposed by Sood et al. [13]: The underground producers build the crimewares and sell them to underground buyers who are the ones launching cyber-attacks.

4.5 Types of Modifications Seen on Phishing Attacks

In this section, we study the type of modifications that are found on our *Evolution Paths*, in order to find out if the modifications are geared toward specific attacks or if we see common trends across attacks. In the following, the analysis is done on the set of *Evolution Paths*, not on the whole SCL graphs. The *Evolution Paths* define a total of 1,624 edges. We will use the following two concepts:

1. The *Modified Tags* (MT) is the set of tags used anywhere on an edge of the set of the *Evolution Paths*. These are the tags that have been added or removed to modify attacks.
2. The *Modification Tags Subsets* (MTS) are all the subsets of the set of tags used on at least one edge of the set of the *Evolution Paths*. We exclude singletons from MTS, so we only consider subsets of at least two tags.

For example, if a SCL graph has only two edges, one labeled with {div:+1, a:+6} and the other one labeled with {input:+3, a:+5, h2:+1}, the set MT is {<div>, <a>, <input>, <h2>} and we have five subsets in MTS, namely {<div>, <a>}, {<input>, <a>, <h2>}, {<input>, <a>}, {<input>, <h2>}, and {<a>, <h2>}.

First, we analyzed the common modification among clusters. The top 10 most common MTs, and the number of clusters in which they appear, are <script> (57), <div> (53), <img> (52), <a> (51), <input> (50), <br> (48), <link> (47), <span> (47), <p> (41), and <style> (40). The top 10 most common MTS among the selected 62 clusters are shown in Table 5. We found that beside the tags <span>, <div> and <br> that are used for spacing or containers, and the functional tags <script>, and <link> that are used for adding scripts

Table 5. The top 10 most common *MTS* in our database.

MTS	# of clusters	%	# of edges	%
{a, div}	45	72.58%	403	24.82%
{div, img}	44	70.97%	286	17.61%
{div, script}	44	70.97%	403	24.82%
{div, span}	40	64.52%	264	16.26%
{br, div}	39	62.90%	215	13.24%
{img, script}	39	62.90%	199	12.25%
{a, img}	37	59.68%	235	14.47%
{link, script}	37	59.68%	215	13.24%
{script, span}	35	56.45%	174	10.71%
{input, span}	35	56.45%	161	9.91%

and resources, phishers only use three tags in the top 10 *MTS*: <a>, <img> and <input>. Figure 5 shows two examples of substantial (visual) modifications were only one tag is actually updated. In Fig. 5(a), one <img> tag was added to change the target. In Fig. 5(b), an email credential phishing attack was converted into a tax return phishing page by changing the background images and adding 31 <input> tags.

We also note that despite the very small number of tags used to perform these modifications, none of the top *MTS* are used by more that 25% of the edges. In order to better understand how common or uncommon each combination of *MTS* is, we computed the *Jaccard Index*: for each pair of clusters, we computed the number of top 10 *MTS* (resp. top 10 *MT*) common to both clusters, divided by the number of top 10 *MTS* (resp. top 10 *MT*) included in either clusters. Figure 6 shows the distribution of the values thus obtained.

As shown in Fig. 6, the distribution of Jaccard Indexes for the pairs of top 10 *MT* covers a relatively wide range, from 0.1 to 0.7. This indicates that different clusters do use the same tags to create the variations, for example <div> or <input>. The distribution of Jaccard Indexes for the pairs of *MTS* on the other hand is very different: most indexes are less than 0.3 and the vast majority (almost 80%) are less than 0.1.

These results show that even through very few tags are actually used when the attacks are modified, the combination of tags used tends to be unique to the attack. In other words, attacks are evolving independently from one another, and the modifications are made for reasons that are specific to each attacks, and not as some sort of global update made across a range of attacks.

5 Related Work

5.1 Phishing Detection

The bulk of academic literature on phishing understandably focuses on the automatic detection of phishing sites. There are three main approaches that have been suggested.

The first one is to identify a phishing attack by comparing it with its target site to find similarities between the two. Rosiello et al. [14] propose a browser extension based on the comparison of the DOM tree, which records the mapping between sensitive information and the related information of legitimate sites (Table 6).

Several papers explore visual similarity comparison. Chen et al. [15] applied the Gestalt Theory to perform a comparison of visual similarity by using normalized compression distance (NCD) as the similarity metric. Sites logo [16] and favicon [17] comparison have also been suggested. Liu et al. [18] proposed a refined comparison method by using block level, layout and overall style similarity. A recent overview of these methods can be found in [19]. The drawback of these methods is that they require some initial knowledge of the targeted legitimate sites. Some authors have suggested to use search engines to acquire this knowledge automatically, for example Cantina [20] which attempts to find the current page on Google and warns if it is not found. Similarly, Huh et al. [21] suggested to search the site's URL in different search engines and use the number of returned pages as an indicator of phishing.

The second approach is to look for intrinsic characteristics of phishing attacks. Cantina+ [22] proposes a system using Bayesian Network mixing 15 features. Gowtham et al. [23] proposed a detection system using a Support Vector Machines (SVM) classifier and similar features to Cantina+. Their system achieved 99.65% true positive and 0.42% false positive. Daisuke et al. [24] conducted an evaluation of nine machine learning-based methods; in their study, AdaBoost provided the best performance. Some research also applies machine learning techniques for detecting phishing emails instead of the phishing site [29–31]. Danesh et al. [32] analyzed more than 380,000 phishing emails over a 15 months period. They found that some attacks keep similar messages over a long period of time, while the other attacks use different messages over time to avoid being detected by email filters.

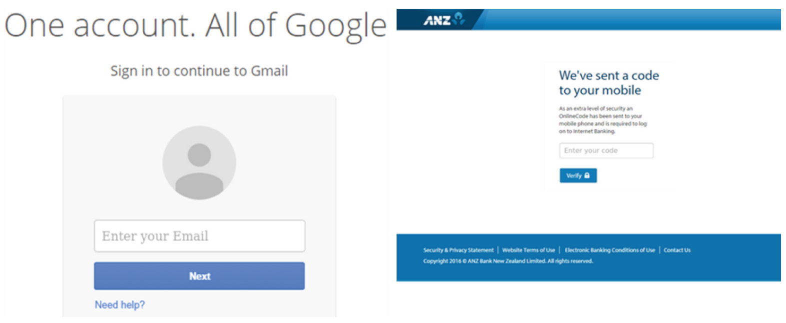
Finally, some new approaches have been proposed recently, in which a phishing attack is compared to known ones. Our previous paper [1] found that most phishing attacks are duplicates or variations of previously reported attacks. Thus, new attack instances can be detected using these similarities. Corona et al. [25] proposed a method to detect attacks hosted on compromised servers, which compares the page of the attack with the homepage that hosts it and the pages linked by it.

Table 6. A summary of related work for phishing detection and phishing kits

Category	Work	Brief description
Comparison to target	Roiello et al. [14]	Compare the layout similarity to identify phishing attacks
	Chen et al. [15]	Applies Gestalt Theory to perform visual similarity comparison
	Chang et al. [16] and Geng et al. [17]	Identify phishing sites by comparing logos and favicons used on target sites
	Liu et al. [18]	A refined visual similarity comparison including block level, page layout and style
	Jain et al. [19]	An overview of phishing detection methods based on visual similarity comparison
Use of search engines	Cantina [20]	Query search engines with the keyword extracted from suspicious sites
	Huh et al. [21]	Feed search engines with suspicious URL, and then use the number of returned pages as the indicator of phishing
Machine learning based methods	Cantina+ [22]	Detect phishing sites using a Bayesian Network
	Gowtham et al. [23]	A SVM classifier is used to identify phishing attacks by using features similar to Cantina+
	Daisuke et al. [24]	An evaluation of nine machine learning methods
Similarity comparison to known attacks	Cui et al. [1]	Identify phishing attacks by comparing the similarity with known attacks
Similarity comparison with homepage	Corona et al. [25]	Compute the similarity score between suspicious pages and the homepage of the same site to detect inconsistencies
Analysis of phishing kits	Cova et al. [26] and Mccalley et al. [27]	Analysis of phishing kits and their obfuscation techniques
	Han et al. [28]	Analysis of phishing attacks and phishing kits collected using a honeypot

5.2 Phishing Kits

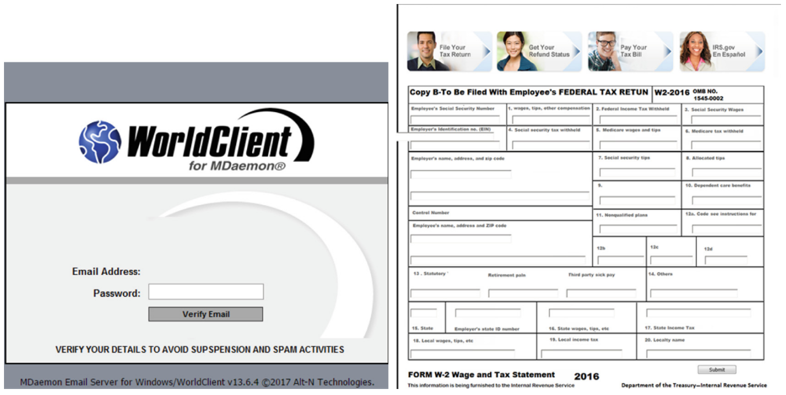
Some of the literature looks at the server side of phishing. Cova et al. [26] collected 584 “phishing kits”. They analyzed the structure of the source code as


One account. All of Google
Sign in to continue to Gmail
Enter your Email
Next
Need help?
ANZ
We've sent a code to your mobile
Enter your code
Verify
Security & Privacy Statement | Website Terms of Use | Electronic Banking Conditions of Use | Contact Us

As an extra level of security an OnlineCode has been sent to your mobile phone and is required to log on to Internet Banking.

Copyright 2016 © ANZ Bank New Zealand Limited. All rights reserved.

(a) One <img> tag was added between the left and the right attack



VERIFY YOUR DETAILS TO AVOID SUPSPENSION AND SPAM ACTIVITIES

MDaemon Email Server for Windows/WorldClient v13.6.4 ©2017 Alt-N Technologies.

Copy B-To Be Filed With Employee's FEDERAL TAX RETUN W2-2016 OMB NO. 1545-0002

FORM W-2 Wage and Tax Statement 2016

This information is being furnished to the Internal Revenue Service

Department of the Treasury—Internal Revenue Service

(b) Between the left and the right, 31 <img> tags are added, and the background image is changed

Fig. 5. Modification of attacks by changing one tag

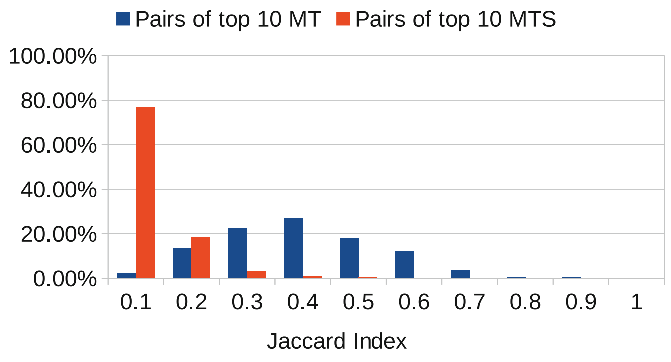


Fig. 6. Histogram of Jaccard index for top 10 *MT* and *MTS*.

well as the obfuscation techniques used. Mccalley et al. [27] did a similar and more detailed analysis of these obfuscation technique. Han et al. [28] collected phishing kits using a honeypot on which 643 unique phishing kits were uploaded. They analyzed the kits' lifespans, victims' behaviors and attackers' behaviors.

To the best of our knowledge, the only work comparable to ours is the research conducted in [32] regarding the evolution of phishing emails. This paper is the first one that gives a good picture of the evolution of phishing sites. Our study provides a detailed analysis of how attackers modify and improve their attacks, and what can motivate these modifications.

6 Conclusion and Future Work

In this paper, we have proposed a new cluster model, the *Semi-Complete Linkage* graph (SCL), to analyze similar phishing attack instances. This model gives us an opportunity to track the evolution of these attacks over time. We discovered that the two main reasons for attackers to update their attacks are aiming at new target and adding new features, e.g. collecting additional information or improving the interface.

Our analysis shows that most attack instances are derived from a small set of "master" attacks, with only a couple of successive versions being deployed. This shows that attackers do not tend to update and improve a baseline of their attacks, and instead keep reworking from the same base version. This suggests that the phishing ecosystem follows a producers-buyers economic model: the producers build and adapt crimewares and sell them to buyers who launch cyber-attacks but barely update them.

Finally, we have also shown that each attack tends to be modified on its own, independently from other attacks; each cluster of attacks uses its own page template and is improved without a general plan across attacks. This could be because a different attacker is beyond each attack, or more likely because attackers follow poor software engineering standards.

Our database comes from Phishtank and X-force, and it has some bias towards some brands [33] and some part of the world (in particular, it lacks data from China and Russia). Therefore, we plan to redo the experiment using a more comprehensive database in the future.

References

1. Cui, Q., Jourdan, G.V., Bochmann, G.V., Couturier, R., Onut, I.V.: Tracking phishing attacks over time. In: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, pp. 667–676 (2017)
2. Anti-Phishing Working Group: Global Phishing Survey: Trends and Domain Name Use in 2016 (2017). http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2015-2016.pdf

3. Anti-Phishing Working Group: Phishing Activity Trends Report 1st Half 2017 (2017). http://docs.apwg.org/reports/apwg_trends_report_h1_2017.pdf
4. Anti-Phishing Working Group: Phishing Activity Trends Report 3rd Quarter 2017 (2017). http://docs.apwg.org/reports/apwg_trends_report_q3_2017.pdf
5. FBI: 2017 Internet Crime Report. https://pdf.ic3.gov/2017_IC3Report.pdf
6. Tekli, J., Chbeir, R., Yetongnon, K.: An overview on XML similarity: background, current trends and future directions. Comput. Sci. Rev. **3**(3), 151–173 (2009)
7. Pawlik, M., Augsten, N.: Tree edit distance: robust and memory-efficient. Inf. Syst. **56**, 157–173 (2016)
8. Manku, G.S., Jain, A., Das Sarma, A.: Detecting near-duplicates for web crawling. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, New York, NY, USA, pp. 141–150 (2007)
9. Fuhr, N., Großjohann, K.: XIRQL: a query language for information retrieval in XML documents. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 172–180. ACM (2001)
10. Grabs, T.: Generating vector spaces on-thefly for flexible xml retrieval. In: [1, Citeseer] (2002)
11. Alexa: Top 500 Sites in Each Country. http://www.alexa.com/topsites/countries
12. WWW: HTML Tag Set. https://www.w3.org/TR/html-markup/elements.html
13. Sood, A.K., Enbody, R.J.: Crimeware-as-a-service-a survey of commoditized crimeware in the underground market. Int. J. Crit. Infrastruct. Prot. **6**(1), 28–38 (2013)
14. Rosiello, A.P.E., Kirda, E., Kruegel, C., Ferrandi, F.: A layout-similarity-based approach for detecting phishing pages. In: Proceedings of the 3rd International Conference on Security and Privacy in Communication Networks, SecureComm, Nice, pp. 454–463 (2007)
15. Chen, T.C., Dick, S., Miller, J.: Detecting visually similar web pages: application to phishing detection. ACM Trans. Internet Technol. **10**(2), 5:1–5:38 (2010)
16. Chang, E.H., Chiew, K.L., Sze, S.N., Tiong, W.K.: Phishing detection via identification of website identity. In: 2013 International Conference on IT Convergence and Security, ICITCS 2013, pp. 1–4. IEEE (2013)
17. Geng, G.G., Lee, X.D., Wang, W., Tseng, S.S.: Favicon - a clue to phishing sites detection. In: eCrime Researchers Summit (eCRS), pp. 1–10, September 2013
18. Liu, W., Huang, G., Xiaoyue, L., Min, Z., Deng, X.: Detection of phishing webpages based on visual similarity. In: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web - WWW 2005, pp. 1060–1061 (2005)
19. Jain, A.K., Gupta, B.B.: Phishing detection: analysis of visual similarity based approaches. Secur. Commun. Netw. **2017**, 20 (2017)
20. Zhang, Y., Hong, J., Lorrie, C.: Cantina: a content-based approach to detecting phishing web sites. In: Proceedings of the 16th International Conference on World Wide Web, Banff, AB, pp. 639–648 (2007)
21. Huh, J.H., Kim, H.: Phishing detection with popular search engines: simple and effective. In: Garcia-Alfaro, J., Lafourcade, P. (eds.) FPS 2011. LNCS, vol. 6888, pp. 194–207. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27901-0_15
22. Xiang, G., Hong, J., Rose, C.P., Cranor, L.: Cantina+: a feature-rich machine learning framework for detecting phishing web sites. ACM Trans. Inf. Syst. Secur. **14**(2), 21:1–21:28 (2011)
23. Gowtham, R., Krishnamurthi, I.: A comprehensive and efficacious architecture for detecting phishing webpages. Comput. Secur. **40**, 23–37 (2014)

24. Miyamoto, D., Hazeyama, H., Kadobayashi, Y.: An evaluation of machine learning-based methods for detection of phishing sites. In: Köppen, M., Kasabov, N., Coghill, G. (eds.) ICONIP 2008. LNCS, vol. 5506, pp. 539–546. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02490-0_66
25. Corona, I., et al.: DeltaPhish: detecting phishing webpages in compromised websites. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) ESORICS 2017. LNCS, vol. 10492, pp. 370–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66402-6_22
26. Cova, M., Kruegel, C., Vigna, G.: There is no free phish: an analysis of "Free" and Live phishing kits. In: 2nd Conference on USENIX Workshop on Offensive Technologies (WOOT), San Jose, CA , vol. 8, pp. 1–8 (2008)
27. McCalley, H., Wardman, B., Warner, G.: Analysis of back-doored phishing kits. In: Peterson, G., Shenoi, S. (eds.) DigitalForensics 2011. IAICT, vol. 361, pp. 155–168. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24212-0_12
28. Han, X., Kheir, N., Balzarotti, D.: Phisheye: live monitoring of sandboxed phishing kits. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1402–1413. ACM (2016)
29. Moradpoor, N., Clavie, B., Buchanan, B.: Employing machine learning techniques for detection and classification of phishing emails. In: IEEE Computing Conference, pp. 149–156 (2017)
30. Akinyelu, A.A., Adewumi, A.O.: Classification of phishing email using random forest machine learning technique. J. Appl. Math. **2014**, 6 p. (2014)
31. Smadi, S., Aslam, N., Zhang, L., Alasem, R., Hossain, M.: Detection of phishing emails using data mining algorithms. In: 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), pp. 1–8. IEEE (2015)
32. Irani, D., Webb, S., Giffin, J., Pu, C.: Evolutionary study of phishing. In: ECrime Researchers Summit, pp. 1–10. IEEE (2008)
33. Clayton, R., Moore, T., Christin, N.: Concentrating correctly on cybercrime concentration. In: WEIS (2015)