Annual ADFSL Conference on Digital Forensics, Security and Law

2015 Proceedings

May 21st, 9:10 AM

Phishing Intelligence Using the Simple Set Comparison Tool

Jason Britt
University of Alabama at Birmingham, Computer and Information Sciences, jrbritt@uab.edu

Alan Sprague
University of Alabama at Birmingham, Computer and Information Sciences

Gary Warner
University of Alabama at Birmingham, Computer and Information Sciences

PHISHING INTELLIGENCE USING THE SIMPLE SET COMPARISON TOOL

Jason Britt, Dr. Alan Sprague, Gary Warner
University of Alabama at Birmingham
Computer and Information Sciences
Birmingham, AL 35233
jrbritt@uab.edu

ABSTRACT

Phishing websites, phish, attempt to deceive users into exposing their passwords, user IDs, and other sensitive information by imitating legitimate websites, such as banks, product vendors, and service providers. Phishing investigators need fast automated tools to analyze the volume of phishing attacks seen today. In this paper, we present the Simple Set Comparison tool. The Simple Set Comparison tool is a fast automated tool that groups phish by imitated brand allowing phishing investigators to quickly identify and focus on phish targeting a particular brand. The Simple Set Comparison tool is evaluated against a traditional clustering algorithm over a month's worth of phishing data, 19,825 confirmed phish. The results show clusters of comparable quality, but created more than 37 times faster than the traditional clustering algorithm.

Keywords: phishing, phish kits, phishing investigation, data mining, parallel processing

1. INTRODUCTION

Phishing websites, phish, attempt to deceive users into exposing their passwords, user IDs, and other sensitive information by imitating legitimate websites, such as banks, product vendors, and service providers. Phish widely range in quality from simple html files to complex replicas indistinguishable from the actual website.

Phishing has been a problem for years and organizations such as the Anti-phishing Working Group (APWG) founded in 2003 and PhishTank founded in 2005 have been fighting phishing for years [1, 2]. Today, phishing is still a problem. A 2013 Kaspersky lab report places phishing attacks as one of the three most prevalent external threats facing corporations [3]. Between April and June of 2014, APWG reported observing 128,378 new phishing attacks [4]. This is the second highest phishing attack volume observed in a three month period by APWG [4]. Phishing attack volumes are large and have increased over the years.

Quickly identifying similar phish imitating a particular brand can be useful for corporate and law enforcement phishing investigators. An entity that can quickly identify itself as the target of a phishing attack can take timely and appropriate responses. It can also tailor its response to the phishing attack that is targeting it. During an investigation, law enforcement gains the ability to focus on phishing attacks against a particular brand. Law enforcement is also able to quickly identify and aggregate all of the phishing attacks against a particular brand.

The currently observed phishing attack volumes make manual phish analysis uneconomical. Fast and scalable automated methods need to be used to assist corporate and law enforcement phishing investigators. Clustering algorithms, which are algorithms used to sort items into groups of similar items, can be used to generate groups or clusters of similar phishing websites. Clustering unbranded phish with branded phish can be used to apply a brand label to the unbranded phish. If a phish cluster contains a

branded phish the same brand can be applied to the unbranded phish in the cluster with some amount of confidence. Traditional clustering algorithms such as k-means, SLINK, and DBSCAN can be used but are relatively slow when operating on large data sets [5].

This paper presents the Simple Set Comparison Tool to cluster large phishing data sets faster than traditional clustering algorithms. The Simple Set Comparison Tool quickly sorts phishing sites into groups of similar phish by brand. The Simple Set Comparison Tool uses a divide and conquer approach to quickly cluster large chronological data sets. The large chronological data set is subdivided or partitioned into many smaller datasets by date and time. Because the dataset is partitioned the most computationally expensive clustering work can be performed on these partitions in parallel by multiple machines. After the computationally expensive work has been performed the smaller partitions are rejoined to form a clustering of the original large dataset, but taking less runtime than traditional methods. Another key feature of the Simple Set Comparison Tool is its adaptability. It can make use of most methods to compare phishing websites and adapt to use most clustering algorithms with very few restrictions.

The tool is evaluated using manually reviewed real world phishing data consisting of 19,825 phish covering 245 brands collected from September 1st 2014 to September 30th 2014. The real world phish have been reviewed by a security company and assigned a brand label representing the brand the phish is imitating.

The tool is evaluated using the brand labels as a ground truth and using several common clustering evaluation metrics. The Simple Set Comparison Tool's clustering quality and runtime are compared to a traditional clustering algorithm's runtime and clustering quality over the same dataset. The results show the Simple Set Comparison Tool produces a similar high quality clustering when compared to the traditional clustering algorithm, but the Simple Set Comparison Tool ran more than 37 times faster.

The Simple Set Comparison Tool makes the following contributions:

1. Aggregates phishing attacks against brands.
2. Parallelizes most clustering tasks resulting in a dramatic runtime improvement over traditional clustering algorithms.
3. Has the adaptability to use a wide variety of phish similarity distance metrics and a wide variety of clustering algorithms.

The rest of the paper is laid out as follows, section 2 discusses related work, section 3 describes the data set used for evaluation, section 4 covers the algorithms used in the Simple Set Comparison Tool, section 5 presents and discusses the comparative results between the Simple Set Comparison Tool and a traditional clustering algorithm, section 6 presents the conclusions drawn, and section 7 covers future work.

2. RELATED WORK

Phishing researchers have been mainly focused on identifying phish versus non-phish websites, also known as binary classification. There is some research attempting to classify phish into more than two categories such as by brand or phish author. However, to the authors' knowledge there is no research attempting to classify phish into brand categories using a parallelizable approach with the adaptability to use most distance metrics and clustering algorithms.

Phishing researchers have presented a number of classification methods for binary classification. These methods can be categorized into three general groups: email advertising phish, URL, and content-based approaches. Some email based approaches classify the words in a spam email's body to determine the legitimacy of the email [6]. Other email-based approaches use features derived from the email message

such as the sender email, sender IP address, and non-matching URLs between the hyperlink and anchor tag [7]. These features are used to classify the email through machine learning algorithms [7, 8]. URL-based approaches have been explored. *Gyawali et al.*. [9]and *Ma et al.* [10] proposed solutions to phishing identification by using features that can be derived from a URL. These researchers demonstrated that URL-based methodologies can identify phishing URLs with high accuracy; however, such techniques can be avoided causing lower detection rates by shortening the phishing URLs or other methods to randomize the URL. Content-based approaches use the content of the phishing website for detection. *Dunlop et al.* [11] presents a method for determining the visual similarity between screenshots of phishing websites. Other researchers have used components within the source code [12] [13]. There have also been a number of researchers that use combinations of all three categories for detection [14] [15] [16].

All of the binary classification techniques lack the ability to identify the brand targeted by the phish and they are not scalable as the techniques and algorithms used are not parallelizable. Also, most of the techniques lack adaptability and can be avoided by the attacker adapting the phish as seen with the URL based approaches.

There are several different techniques presented in research to classify phish into more than binary categories or cluster phish. Phish clustering has been an area of interest for researchers that are proactively trying to determine the criminals behind the phishing attacks [12, 17, 18]. Criminals use to create domains on the same IP blocks, which *Weaver and Collins* leverage in a clustering algorithm using the IP address or hosting network to cluster phish [13]. Similar attack patterns against domains have been used to attribute phishing attacks to particular criminals [19]. IP and hosting network based solutions have been avoided by criminals adapting to use botnets or compromised webservers spread over different IPs and hosting networks.

The Deep MD5 technique has been used to cluster phish into groups of related phish using local domain files [20, 21]. The Deep MD5 technique can be avoided by slightly changing phish files from phish to phish, although this behavior has not yet been seen in the wild [20, 21, 22]. The most recent method called Syntactical fingerprinting uses structural components to cluster phishing websites by brand and by criminal [22]. Email addresses receiving phished credentials found in the kits used to create the phish have also been used to cluster phishing websites [23]. The non-binary classification techniques take unique approaches to classifying phish, but none can be performed in parallel. Also, all of the classification techniques lack adaptability as they rely on a particular phish comparison metric or clustering algorithm. All of the non-binary classification techniques presented lack scalability and adaptability.

Existing research is mainly focused on binary classification. However, there is some research focused on non-binary phish classification such as classifying phish by brand or criminal. There is a lack of techniques that can perform phish classification using scalable methods such as parallelization. Also, there is no adaptable and scalable phish classification technique that can liberally use different phish comparison methods and different classification algorithms should criminals adapt their phishing attacks.

3. DATA SET

The phishing URLs are gathered from a large spam-based URL provider, a large anti-phishing company, and a number of other feeds including private companies, security companies, and financial institutions. The source of the URLs is either URLs contained in spam advertising phish or URLs reported by the public to fraud alert email addresses. The data set favors financial institutions and under represents gaming and social media phish when compared to other phishing collections.

A number of methods are used in the industry to count phish. Some methods count distinct URLs. If there is any randomization in the host name, directory path, or arguments it leads to 'over-counting'. Cases where this occurs include wild-card DNS entries, per user customized URLs, or virtual hosts allowing the same directory path for multiple domains to resolve to a single IP address. A conservative counting approach that attempts to de-duplicate URLs leading to the same phishing content is used.

The phishing data consists of all files referenced in the potential phish. The website files are fetched using an automated web crawler that makes use of a Firefox mechanization tool [24]. After the files are downloaded, a hash value is generated for each file using the MD5 hashing algorithm. While the MD5 hashing algorithm is not a cryptographically secure algorithm it is not being used for a cryptographic purpose, but rather to identify individual files. MD5 hash values can be changed by slightly altering a file's content each time it is deployed in a phish. However, phish authors would have to create a phish kit to automate the file changes needed to perform this for every file every time a phish is deployed. So far this behavior has not been observed in the wild. For this reason the authors feel the MD5 hashing algorithm is acceptable to use for file identification in this instance. Screenshots and the domain information are manually reviewed to determine whether the potential phish is a phish.

Brand	Count
Tech Company 1	3,815
Telecom Company 1	1,720
Tech Company 2	1,484
Financial Institution 2	1,435
Financial Institution 3	829
Tech Company 3	786
Tech Company 4	709
Financial Institution 4	657
Tech Company 5	589
Financial Institution 5	529

Figure 1 Ten Most Numerous Brands Phished

The data set consists of 19,825 confirmed phishing sites collected between September 1st 2014 and September 30th 2014. There are a total of 245 different brands. Figure 1 shows an anonymized by sector listing of the 10 most phished brands in the data set.

4. ALGORITHMS

The Simple Set Comparison Tool consists of four broadly defined steps.

1. Creating Time Windows
2. Clustering Time Windows
3. Comparing Time Windows
4. Merging Time Windows

In the first step, user specified single time windows are created. A cross time window is created for every combination of user specified single time windows. In the second step, the single and cross time windows are clustered independently of one another. All single and cross time window clustering processes can be run in parallel. In the third step, single time windows are compared to overlapping cross time windows based upon shared cluster members resulting in a cluster similarity graph. The time window comparisons can be run in parallel. In the fourth step, a clustering algorithm is then run over the cluster similarity graph to merge similar clusters. The result is a clustering of the entire data set.

The Simple Set Comparison Tool takes advantage of parallel processing in the second and third steps. To be able to process a large data set in parallel the data set must first be subdivided or partitioned. The phish data is tagged with a received time which allows partitioning on a chronological basis. The parallel processing in steps two and three are the key to reducing the runtime compared to traditional clustering algorithms.

4.1 Creating Time Windows

The data set is subdivided into four different time windows of approximately seven days each consisting of the following date ranges 09/01/2014 to 09/08/2014, 09/09/2014 to 09/15/2014, 09/16/2014 to 09/22/2014, and 09/23/2014 to 09/30/2014. Throughout the rest of the paper these date ranges will be referred to by their date range window number as presented in figure 2.

Begin Date	End Date	Window Number
9/1/2014	9/8/2014	1
9/9/2014	9/15/2014	2
9/16/2014	9/22/2014	3
9/23/2014	9/30/2014	4

Figure 2 Date Ranges and Their Date Range Number

As well as single time date ranges, single time windows, multi-date ranges are created, cross time windows. Cross time windows are created by merging data from two single time windows together. This is performed for every combination of single time windows. In this case, creating combinations of the four single time windows results in the following six cross time window combinations; 1:2, 1:3, 1:4, 2:3, 2:4, and 3:4. Throughout the rest of the paper cross time windows will be referred to by their cross time window number. For example, the cross time window that crosses time windows one and two is 1:2.

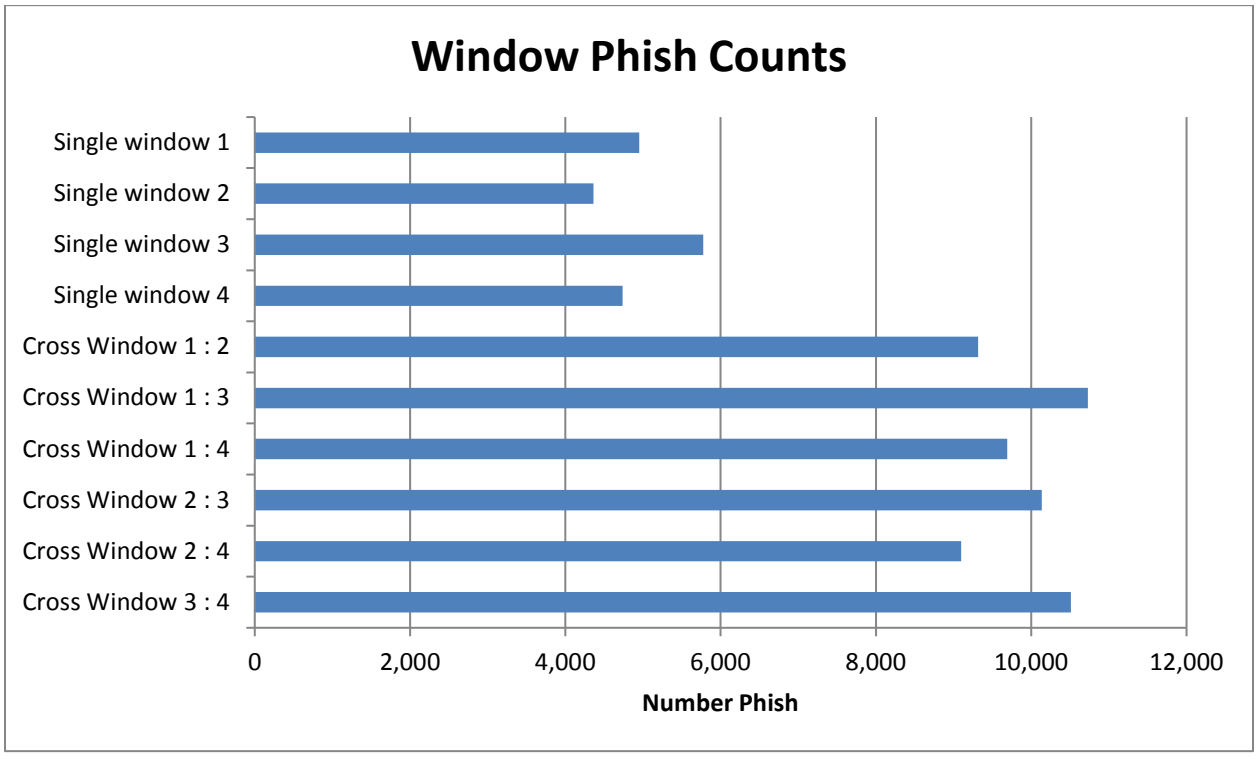

Figure 3 Single and Cross Time Windows and Number of Phish in Each

The number of phish in each window is depicted in figure 3. The four single time windows consist of approximately 4,000 to 5,000 phish each. The six cross time windows contain between 9,000 to almost 11,000 phish. The cross time window data sets are about twice as large as the single time window data sets.

4.2 Clustering Time Windows

The phish time windows and cross time windows are clustered by comparing phishing websites using the Deep MD5 method as a similarity score and a SLINK clustering algorithm to sort the phish into groups based upon their similarity scores [25]. Deep MD5 generates a score based upon file set similarity. Deep MD5 generates a score using the count of candidate one's files (count1), the count of candidate two's files (count2), and the number of matching file MD5 values between candidate one and candidate two (overlap).

$$Kulczynski\ 2\ Coefficient = 0.5\left(\frac{overlap}{count1}\right) + 0.5\left(\frac{overlap}{count2}\right)$$

A Kulczynski 2 coefficient, equation 1, is then applied to count1, count2, and overlap to generate the Deep MD5 score with a value between 0.0 and 1.0. For example two websites, website X and website Y, could be compared using Deep MD5. If website X consists of files {a,b,c,d,e} and website Y consists of files {a,b,f,g} then the overlap count between the two websites' file sets is two (overlap). Website X's file count is five (count1) and website Y's file count is four (count2). Then the Deep MD5 score is 0.5(2/5) + 0.5(2/4) or 0.45.

After the Deep MD5 similarity scores are generated the results are feed to a SLINK clustering algorithm. The SLINK clustering algorithm is a graph theoretic clustering algorithm. The graph has vertices of phishing websites and for each pair of vertices there exists a deep MD5 similarity score. Edges where the similarity score meets or exceeds a threshold are kept and edges not meeting the minimum threshold are discarded. An analysis of Deep MD5 scores between phish showed good matching results between phish with the same brand for threshold values ranging from 0.5 to 0.75 with very little change [26]. A 0.6 value is chosen as a middle ground between the high and low end threshold values. After all edges have been pruned, the SLINK clustering algorithm turns connected components into clusters.

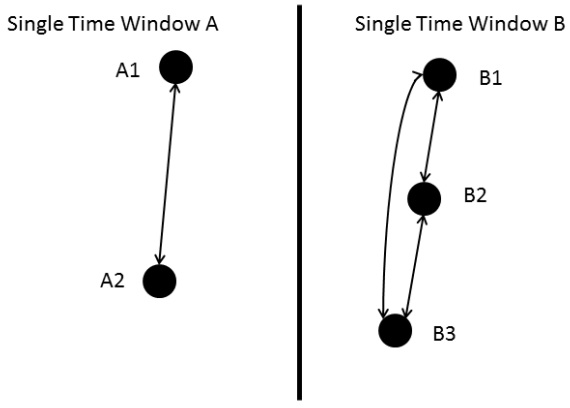

Figure 4 Similarity Score Generation for Single Time Windows A and B

Each single time window is clustered by generating similarity scores for all phish from a single window and then applying a SLINK clustering algorithm. Figure 4 shows similarity scores being generated for two single time windows before the clustering algorithm is applied.

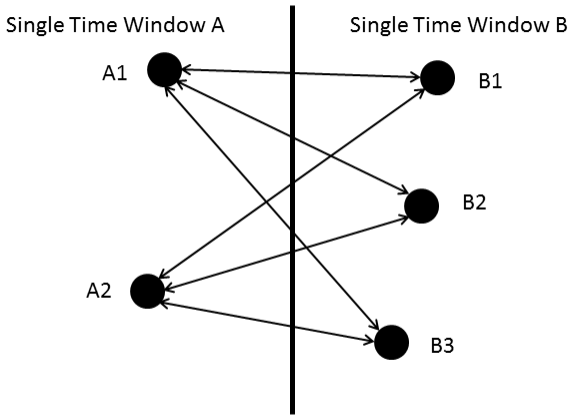

Figure 5 Similarity Score Generation for Cross Time Window A:B

The six cross time windows are clustered by generating a similarity score for all phish from one window compared to another window and applying a SLINK clustering algorithm. Phish from the same time window are not compared, only phish from different windows are compared. Figure 5 shows similarity scores being generated for a single cross time window before the clustering algorithm is applied. Clustering the four single time windows and six cross time windows can be performed independently of one another, allowing all clustering processes to be run in parallel instead of in sequence.

4.3 Comparing Time Windows

The cross time window clusters are used to merge the individual clusters from different time windows. Clusters from single time windows are compared to clusters from overlapping cross time windows. The clusters are compared by counting the number of phish shared between the two clusters (overlap) divided by the total number of phish in the single time window cluster (count1) resulting in a score between 0.0 and 1.0.

$$Cluster\ Membership\ Similarity = \frac{overlap}{count1}$$

The cross time window's size is not included as it will dilute the similarity score. Because the cross time window clusters can incorporate phish from two time windows they are generally much larger. Each single time window to cross time window comparison can be run independently of one another. Comparing time window clusters can be performed in parallel. Comparing time windows based upon shared cluster members results in a cluster similarity graph for all clusters from all time windows.

4.4 Merging Clusters

A clustering algorithm is then run over the cluster similarity graph to merge similar clusters from different time windows. A SLINK clustering algorithm is used to determine the cluster meges. The SLINK clustering algorithm is chosen because of its simplicity as this is an initial investigation into the effectiveness of the Simple Set Comparison tool. The clustering algorithm used in this step of the Simple Set Comparison Tool is interchangeable. The only requirement is the clustering algorithm takes an edge based representation of a graph and produce non-overlapping clusters.

5. RESULTS

The Simple Comparison Set Tool is compared to a traditional clustering algorithm, a SLINK clustering algorithm, run over the same data. The SLINK clustering algorithm uses a Deep MD5 similarity score with a threshold of 0.6. The Simple Set Comparison Tool uses a SLINK clustering algorithm with a DeepMD5 threshold of 0.6 for its first step. A SLINK clustering algorithm with a Deep MD5 threshold of 0.6 is chosen to provide an apple to apples comparison to the Simple Set Comparison tools results.

The Simple Set Comparison Tool and the traditional clustering both ran on the same hardware, a 64 bit Windows 7 Enterprise desktop with an Intel Core2 Quad CPU running at 2.83GHz and 8.00 Gigabytes of RAM. Both methods use the same java implementation of the SLINK clustering algorithm. All results are stored to the same Postgresql data base on the local machine.

The first subsection compares the clustering quality produced by different runs of the Simple Set Comparison Tool with varying cluster merging thresholds, used in step four. A single merging threshold value is chosen after analysis of the clustering quality and is used to compare against the traditional clustering algorithm. The second subsection compares the quality of the clustering produced by the Simple Set Comparison Tool and the traditional SLINK clustering using three different cluster quality measures. The third subsection shows an anecdotal comparison of the largest ten clusters generated by the Simple Set Comparison Tool and the traditional clustering algorithm. The fourth subsection computes the runtime of the Simple Set Comparison Tool and compares it to that of the traditional clustering algorithm. The fifth subsection discusses the algorithms used in the Simple Set Comparison Tool. The sixth subsection discusses issues that may cause runtime performance to decrease.

5.1 Comparing Cluster Merging Thresholds

The clustering quality is measured using three different entropy based metrics; homogeneity, completeness, and V-measure [28]. All three measures are based upon evaluating the clustering results compared to a ground truth label assigned to all data points. The ground truth label assigned to the data points represents a perfect clustering of the dataset. The three different measures evaluate how close to a perfect clustering is created. The ground truth label used is the phish brand.

Homogeneity evaluates how well the clustering is at placing members that should be in the same cluster in the same cluster. A perfect homogeneity score is achieved when all clusters only contain members with the same label. Completeness evaluates how well the clustering came to determining the correct number of clusters. A perfect completeness score is achieved when there is only one cluster for each label. V-measure is the harmonic mean of the homogeneity and completeness scores, a blend of homogeneity and completeness scores. Also included is the number of clusters created.

The Simple Set Comparison Tool is evaluated over eleven different thresholds ranging from 0.001 to 1.0. These ranges are chosen as the smallest and largest thresholds to evaluate because the smallest cluster similarity score found above 0.0 is approximately 0.0094 and the largest cluster similarity score is 1.0. The other thresholds used range from 0.1 to 0.9 with 0.1 increments between to get the best coverage without an exhaustive search of all threshold values.

Threshold	Number Clusters	Homogeneity	Completeness	V-Measure
0.001	1,248	0.9871	0.6778	0.8037
0.1	1,281	0.9872	0.6761	0.8025
0.2	1,321	0.9872	0.6739	0.8010
0.3	1,324	0.9872	0.6729	0.8003
0.4	1,332	0.9872	0.6716	0.7994
0.5	1,332	0.9872	0.6716	0.7994
0.6	1,335	0.9873	0.6702	0.7984
0.7	1,368	0.9866	0.6683	0.7969
0.8	1,377	0.9851	0.6641	0.7934
0.9	1,386	0.9845	0.6606	0.7907
1	1,416	0.9850	0.6352	0.7723

Figure 6 Simple Set Comparison Tool Clustering Quality Measures

The cluster quality measures stay very consistent across the thresholds. As the threshold increases the homogeneity scores only changes in the third and fourth decimal places. Oddly though the homogeneity scores rise slightly until the 0.6 threshold and then fall slightly until the 1.0 threshold. This may be due to an unusual breakdown of good similarity clusters at very high thresholds. The completeness score decreases slightly but consistently from the 0.001 threshold to the 1.0 threshold. The v-measure score that measures the tradeoff between homogeneity and completeness slightly decreases from the 0.001 to the 1.0 threshold. The degradation of completeness without a corresponding improvement in homogeneity begins at 0.3. The relative degradation of the clustering is also reflected in the declining v-measure score from the 0.3 to 0.4 thresholds. The clustering result produced by the 0.3 merging threshold is chosen as a best representative to compare to the traditional clustering algorithm on clustering quality.

5.2 Comparing Clustering Quality Results

The 0.6 threshold value used for traditional clustering is the same value used in the Simple Set Comparison Tool during the initial clustering of the time windows. The 0.6 threshold should serve as a good benchmark. As it is noted in a Deep MD5 evaluation paper [26] there is little difference between the 0.5 and 0.75 DeepMD5 threshold values.

Figure 7 Traditional Clustering and Simple Set Comparison Quality Measures

Comparing the clustering quality measures shows almost no differences as the homogeneity, completeness, and v-measure scores are all relatively similar. Traditional clustering produces a slightly better completeness score. There are negligible differences between the quality of clusterings produced.

5.3 Anecdotal Comparison

An anecdotal comparison between the ten largest clusters generated by traditional clustering and the Simple Set Comparison Tool is presented below.

Cluster Brand	Traditional Clustering Cluster Size	Simple Set Comparison Tool Cluster Size
Telecom Company 1	1,291	1,291
Tech Company 1	915	915
Financial Institution 3	794	794
Financial Institution 2	770	770
Tech Company 3	637	637
Tech Company 2	567	567

Tech Company 1	365	365
Financial Institution 6	303	303
Telecom Company 1	303	303
Financial Institution 4	302	302

Figure 8 Ten Largest Clusters Produced By Traditional and Simple Set Comparison Tool Clustering

Traditional clustering and the Simple Set Comparison Tool both produced the same ten largest clusters. Meaning, both sets of ten are perfectly homogeneous, have the exact same cluster sizes, and have the same brand label. The individual phish that make up both of the sets of ten were not compared to determine if they have exactly the same phish contained in each corresponding cluster.

5.4 Runtime Comparison

The total runtime for the Simple Set Comparison Tool is computed by adding the runtime for each step together. There is no runtime spent for the first step as the chronological dividing points for each time window are chosen before the tool is run. The second step is run in parallel, ideally each clustering process is run on a separate machine; the runtime for step two will be the longest runtime out of the group. The third step's runtime is the longest comparison runtime out of all single to cross time window comparisons. Since the third step is run in parallel, ideally each comparison process is run on a separate machine; the runtime for step three will be the longest runtime out of the group. The fourth step is not parallelized. The fourth step's runtime will be the runtime it takes to assemble a global clustering out of the cluster similarity graph generated in step three. The parallel clustering processes run in step two have the largest runtimes out of all of the steps and their runtimes are presented in Figure 9.

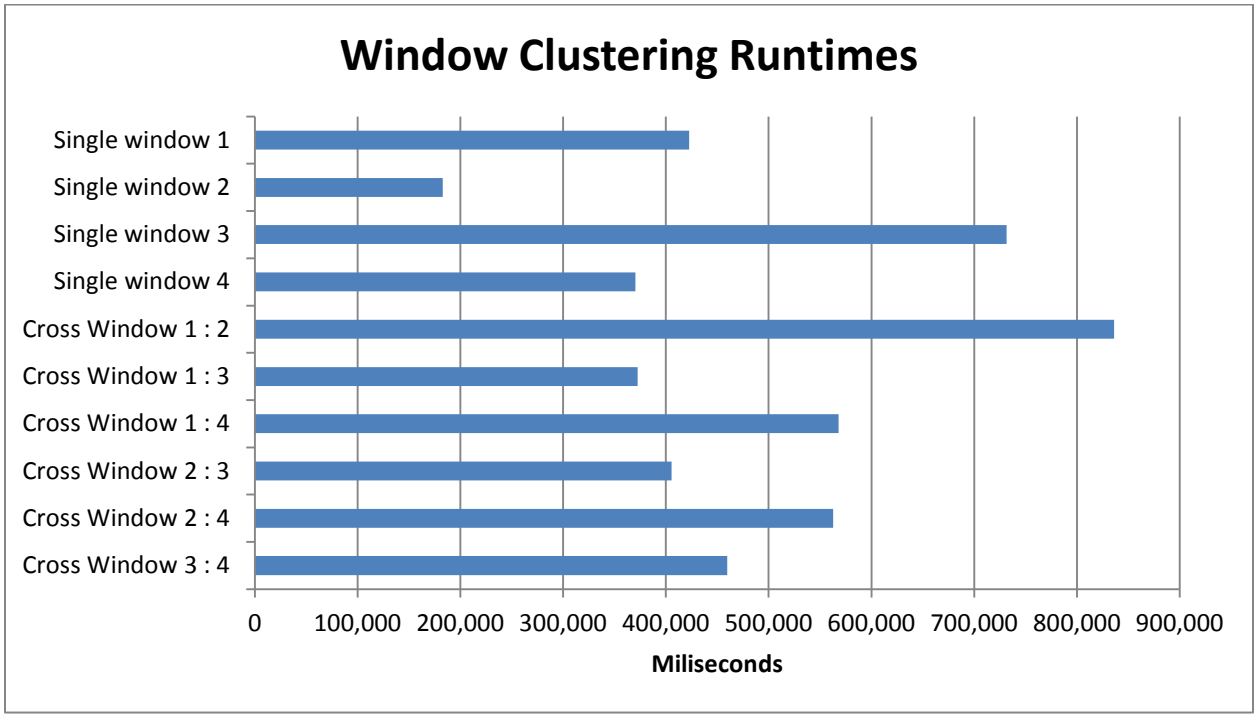

Figure 9 Clustering Runtimes for Single and Cross Time Windows.

Clustering the single time windows takes between three minutes for the fastest and twelve minutes for the slowest. Clustering the cross time windows takes between almost seven minutes and almost fourteen

minutes. The longest runtime out of the group is cross window 1:2 at almost 14 minutes, 836,108 milliseconds. Step two, comparing single to cross time window clusters, took very little time. All twelve of the comparisons took only 640 milliseconds combined. The longest comparison took 93 milliseconds and the shortest took 31 milliseconds. The longest runtime out of the twelve comparisons is 93 milliseconds. Step four, merging time windows, is not parallelized and has a single runtime of 1,324 milliseconds. Adding the longest runtime for step two (836,108 milliseconds), the longest runtime for step three (93 milliseconds), and step four's runtime (1,324 milliseconds) results in a total runtime of almost fourteen minutes (837,525 milliseconds). The biggest contributor of total runtime comes from step two, clustering single and cross time windows. In particular it comes from clustering the cross time window 1:2. The traditional clustering algorithm took over eight and a half hours, 31,044,322 milliseconds, to complete. The Simple Set Comparison Tool's runtime is more than 32 times faster than the traditional clustering algorithm's runtime.

There is almost no difference between the quality of clusterings produced by the Simple Set Comparison Tool and traditional clustering. The runtime is the biggest difference between the two. The Simple Set Comparison Tool is more than 37 times faster at producing results for the monthly dataset.

5.5 Interchangeable Algorithms

The Simple Set Comparison Tool has three interchangeable pieces. The distance metric used for comparing phish in the first step, the clustering algorithm used to cluster phish in the first step, and the clustering algorithm used when merging similar clusters in the third step.

The Simple Set Comparison Tool can make use of a variety of phish similarity metrics. The tool only requires the similarity measure be numeric and have an upper and lower bound. The Simple Set Comparison Tool can make use of a variety of clustering algorithms. The Simple Set Comparison Tool requires a clustering algorithm to take an edge representation of a graph as input and produce non-overlapping clusters. These requirements allow the Simple Set Comparison Tool to use a variety of existing clustering algorithms.

The Deep MD5 metric is being used as a similarity measure in the first step as it has been shown to be useful for clustering phishing websites [26] [21] [20]. The Deep MD5 metric is not fool proof as it relies on file reuse by phish. Small changes to a file will change the MD5 value for that file. If a phishing author was so inclined all content files referenced by a phish could be slightly changed each time a particular phish was created. The result would be a Deep MD5 score of 0.0 between two phish created by the same author that targeted the same brand with the same functionality and appearance. However, this has not been noticed to be prevalent in the wild at this time. If this does occur at some future date, the similarity metric used by the Simple Set Comparison Tool is interchangeable and another more sufficient phish similarity metric can be used in place of Deep MD5. The only requirement the Simple Set Comparison Tool has for a comparison metric is that the metric produces a single numerical value within a defined upper and lower bound. The Deep MD5 similarity metric is being used as an example similarity metric that is currently effective in this particular use case.

The SLINK clustering algorithm is used in step one for clustering time windows and in step three when merging clusters. The same clustering algorithm does not have to be used in both step one and step three. Indeed there may be circumstances where using a different clustering algorithm in step one and step three may produce better results. However, in this particular case using the SLINK clustering algorithm to cluster the time windows in step one and merge clusters in step three is effective as it produces a clustering of similar high quality to traditional clustering. The SLINK clustering algorithm is not the best or newest clustering algorithm. It is a simple clustering algorithm and has been shown to produce good results when applied to clustering phish [26] [21] [23] [27]. Like the similarity metric, the clustering algorithm used by the Simple Set Comparison Tool is interchangeable. The Simple Set Comparison Tool

only requires a clustering algorithm take an edge based representation of a graph as input and produce non-overlapping clusters within a single data set.

5.6 Performance Discussion

The Simple Set Comparison Tool results show a drastic runtime gain when compared to the traditional clustering algorithm's runtime. However, issues can arise that would increase runtime. Steps two and three are performed in parallel and are scalable. Step four is not run in parallel and the runtime could become a problem under certain circumstances. The key driving factor for step four's runtime is the size of the similarity graph, or number of clusters, produced by step three.

There are two ways the number of clusters created would increase. The first is by increasing the number of time windows. For example if an existing time window is subdivided into two time windows the number of clusters produced would be approximately double assuming each of the clusters generated from the original time window would effectively be split in half. The second way the number of clusters created would increase is if the clustering algorithm used in step two has a low completeness score, thus producing many more clusters than needed. The Simple Set Comparison tool's runtime improvement versus the traditional clustering algorithm is achieved over the month long data set by using appropriately sized time windows containing many phish and using a clustering algorithm in step two that has a sufficient completeness score. While the window sizes used in this evaluation have not been optimized through an exhaustive search the selected window sizes achieve a large performance gain and generate a clustering of equivalent quality to the traditional clustering algorithm.

6. CONCLUSIONS

The clustering quality metrics show the Simple Set Comparison tool's results are essentially equivalent to the traditional clustering output, which are good at a cluster homogeneity score above 0.98. The Simple Set Comparison tool's runtime is drastically better than the traditional clustering runtime. The runtime improvement is due to the Simple Set Comparison Tool partitioning the dataset and performing a majority of its clustering in parallel.

The Simple Set Comparison Tool works well with the Deep MD5 comparison metric and SLINK clustering algorithm when clustering phish data. However, the Simple Set Comparison Tool is adaptable enough to make use of many different comparison metrics and clustering algorithms. It can quickly create quality phish clusters that can be used for phish identification or aggregation by phishing investigators.

7. FUTURE WORK

Further evaluations need to be performed on different data sets to determine the general applicability of the Simple Set Comparison tool. A future goal is to evaluate the Simple Set Comparison tool's ability to deal with heterogeneous data. One example would be creating clusters consisting of phish, spam advertising phish, and kits used to create phish. Incorporating two more sources of data, especially spam, would significantly increase the amount of data to cluster. It will also require the use of multiple similarity metrics. Deep MD5 can only be used to compare two phish and cannot be used to relate URLs found in spam to phishing websites. Other similarity metrics will have to be developed.

The inclusion of multiple similarity metrics used over heterogeneous data may necessitate the use of more sophisticated clustering algorithms. Each similarity metric would represent a different type of relationship between data points such as phish and spam email versus phish and phish kits. The different types of relationships may have different value ranges and distributions over their respective value ranges. A more locally adaptable clustering algorithm may be required to generate adequate clusters over such heterogeneous data.

REFERENCES

[1] APWG, "About the APWG," APWG, 2014. [Online]. Available: https://apwg.org/about-APWG/ . [Accessed 11 December 2014].

[2] PhishTank, "PhishTank FAQ," PhishTank, [Online]. Available: http://www.phishtank.com/faq.php#whatisphishtank. [Accessed 11 December 2014].

[3] Kaspersky Lab, "Kaspersky Lab," Kaspersky Lab, 8 November 2013. [Online]. Available: http://www.kaspersky.com/about/news/virus/2013/Malware_spam_and_phishing_the_threats_most_commonly_encountered_by_companies. [Accessed 11 December 2014].

[4] G. Aaron and R. Manning, "Phishing Activity Trends Report 2nd quarter 2014," 29 August 2014. [Online]. Available: http://docs.apwg.org/reports/apwg_trends_report_q1_2014.pdf. [Accessed 11 December 2014].

[5] J. Han and M. Kamber, Data Mining: Concepts and Techniques, San Diego, CA: Academic Press, 2001, p. 337.

[6] A. Saberri, M. Vahidi and B. M. Bidgoli, "Learn to Detect Phishing Scams Using Learning and Ensemble Methods," in *Web Intelligence and Intelligent Agent Technology Workshops*, Silicon Valley, CA, 2007.

[7] S. Abu-Nimeh, D. Nappa, X. Wang and S. Nair, "A Comparison of Machine Learning Techniques for Phishing Detection," in *eCrime Researchers Summit*, Pittsburgh, PA, 2007.

[8] I. Fette, N. Sadeh and A. Tomasic, "Learning to Detect Phishing emails," in *Proceedings of the 16th International Conference on World Wide Web*, Banff, Alberta Canada, 2007.

[9] B. Gyawali, T. Solorio, M. Montes-y-Gomez, B. Wardman and G. Warner, "Evaluating a Semisupervised Approach to Phishing URL Identification in a Realistic Scenario," in *Conference on Email and Anti-Spam*, Perth, Australia, 2011.

[10] J. Ma, L. Saul, S. Savage and G. Voelker, "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, 2009.

[11] M. Dunlop, S. Groat and D. Shelly, "GoldPhish: Using Images for Content-Based Phishing Analysis," in *The Fifth International Conference on Internet Monitoring and Protection*, 2010.

[12] R. Basnet, S. Mukkamala and A. H. Sung, "Detection of Phishing Attacks: A Machine Learning Approach," in *Studies in Fuzziness and Soft Computing*, 2008, pp. 373-383.

[13] R. Suriya, K. Saravanan and A. Thangavelu, "An Integrated Approach to Detect Phishing Mail Attacks A Case Study," in *Proceedings of the 2nd International Conference on Security of Information and Networks*, North Cyprus, Turkey, 2009.

[14] C. Whittaker, B. Ryner and M. Nazif, "Large-Scale Automatic Classification of Phishing Pages," in *Network and Distributed Systems Security Symposium*, San Diego, CA, 2010.

[15] G. Xiang and J. Hong, "A Hybrid Phish Detection Approach by Identity Discovery and Keywords Retrieval," in *Proceedings of the 18th International Conference on World Wide Web*, Madrid, Spain, 2009.

[16] Y. Zhang, J. Hong and L. Cranor, "CANTINA: A Content-Based Approach to Detecting Phishing Web Sites," in *International Conference on World Wide Web*, Banff, Alberta, Canada, 2007.

[17] D. Irani, S. Webb, J. Griffon and C. Pu, "Evolutionary Study of Phishing in eCrime Researchers Summit," in *eCrime Researchers Summit*, Atlanta, GA, 2008.

[18] R. Weaver and M. Collins, "Fishing for Phishes: Applying Capture-Recapture Methods to Estimate Phishing Populations," in *Proceedings of the Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit*, Pittsburgh, PA, 2007.

[19] B. Wardman, G. Shukla and G. Warner, "Identifying Vulnerable Websites by Analysis of Common String in Phishing URLs," in *eCrime Researchers Summit*, Tacoma, 2009.

[20] J. Britt, B. Wardman, A. Sprague and G. Warner, "Clustering Potential Phishing Websites Using DeepMD5," in *Proceedings of the 5th USENIX Conference on Large-Scale Exploits and Emergent Threats*, 2012.

[21] B. Wardman, G. Warner, H. McCalley, S. Turner and A. Skjellum, "Reeling in Big Phish with a Deep MD5 Net," *Journal of Digital Forensics, Security, & Law,* vol. 5, no. 3, pp. 33-55, 2010.

[22] B. Wardman, J. Britt and G. Warner, "New Tackle to Catch A Phisher," *International Journal of Electronic Security and Digital Forensics,* vol. 6, no. 1, pp. 62-80, 2014.

[23] S. Zawoad, A. Dutta, A. Sprague, R. Hasan, J. Britt and G. Warner, "Phish-Net: Investigating Phish Clusters Using Drop Email Addresses," in *2014 APWG eCrime Researchers Summit*, San Francisco, 2013.

[24] M. Maischein, "WWW::Mechanize::Firefox," [Online]. Available: http://search.cpan.org/dist/WWW-Mechanize-FireFox/.. [Accessed 2013 01 01].

[25] R. Sibson, "SLINK: An optimally efficient algorithm for the single-link cluster method," *The Computer Journal,* vol. 16, no. 1, pp. 30-34, 1973.

[26] B. Wardman, T. Stallings, G. Warner and A. Skjellum, "High-Performance Content-Based Phishing Attack Detection," in *eCrime Researchers Summit*, San Diego, CA, 2011.

[27] A. Rosenberg and J. Hirschberg, "V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure," *EMNLP-CoNLL,* vol. 7, pp. 410-420, 2007.

[28] B. Wardman, J. Britt and G. Warner, "New Tackle to Catch a Phisher," *International Journal of Electronic Security and Digital Forensics,* vol. 6, no. 1, pp. 62-80, 2014.

