High-Performance Content-Based Phishing Attack Detection

Brad Wardman, Tommy Stallings, Gary Warner, Anthony Skjellum
Computer Forensics and Research Laboratory, University of Alabama at Birmingham
{bwardman, tds2, gar, tony}@uab.edu

***Abstract*— Phishers continue to alter the source code of the web pages used in their attacks to mimic changes to legitimate websites of spoofed brands and to avoid detection by phishing countermeasures. Manipulations can be as subtle as source code changes or as apparent as adding or removing significant content. To appropriately respond to these changes to phishing campaigns, a cadre of file matching algorithms is implemented to detect phishing websites based on their content, employing a custom data set consisting of 17,684 phishing attacks targeting 159 different brands. The results of the experiments across different algorithms demonstrate that some of the approaches for phishing detection can achieve a detection rate of greater than 90%.**

***Index Terms*—Phishing; Countermeasures; Content-based approaches; Cybercrime**

I. Introduction

THE social engineering attack known as phishing continues to evolve as the attacks change in response to improved countermeasures such as spam filters and blacklists. Phishing can be summarized as an attack where victims are enticed to submit personal identifying information (PII) online, thinking that they are either verifying their account or going to receive a benefit such as an award for answering a survey. The origins of phishing were observed in the early 1990s on America Online (AOL) when offenders created false AOL accounts using stolen credit cards. However, AOL responded to this hack with new, stronger authentication measures that linked credit card numbers to legitimate users [1]. Subsequently, the identity thieves changed their attack vector by posing as legitimate AOL employees and requesting login credentials from AOL users, typically through email or instant messaging. These early methods have since been used for many years to acquire usernames and passwords for financial accounts and other online authentication systems. More recently, attacks have evolved against newer Internet phenomenon such as social networking sites (*i.e.*, Facebook and MySpace) and gaming sites (*i.e.* World of Warcraft and Steam) [2]. APWG's second half report for 2010 claimed that phishing attacks grew 142% over the first half of 2010. The report also classifies the targets as comprising 37.9% payment services, 33.1% financial institutions, 6.6% classified, 4.6% gaming, 2.8% social networks, and the remainder in other categories [3]. Phishers use a number of techniques to lure their victims, including email messages, instant messages, forum posts, phone calls, and text messages.

The traditional response to phishing by spoofed organizations is to remove the malicious content from the Internet, which is often referred to as "takedown." Takedown has demonstrated the ability to help reduce the number of victims per attack, primarily because the malicious content is removed for future potential victims. However, phishers respond by rapidly creating new phishing websites on other servers, thus creating an unbounded cycle wherein the phished organizations identify new attacks, contact server administrators, and request takedown again. In response, researchers at the University of Alabama at Birmingham (UAB) have developed anti-phishing techniques that aim to automatically identify websites to block Internet users within the browser as well as reduce the time needed to review sets of potential phishing URLs.

Members of the Phishing Operations and Investigations Teams at the UAB Computer Forensics Research Laboratory explore new trends occurring in phishing attacks. The UAB Phishing Data Mine is a system consisting of over 190,000 phishing websites and associated content files that attempts to automatically identify phishing websites based on previously observed patterns. However, if these patterns are not found within a potential website, then a Phishing Operations Team member manually reviews the website. The manual review of new types of phishing attacks helps to improve automated phish detection and the automated process of target brand identification that is also part of the data mining system. This research builds upon existing content-based approaches by analyzing the performance of different content-based techniques and incorporating new techniques to improve performance where known methods were lacking. A further analysis of phishing content not identified by automated solutions in the UAB Phishing Data Mine led to the research presented in this paper. This research aims to increase the

This research was made possible by the support of the UAB Department of Computer & Information Sciences, the UAB Department of Justice Sciences, and funding received from the Bureau of Justice Assistance (#2008-DD-BX-0407) and the Office of Community Oriented Policing Services (#2006-CKWX-0582). Points of view and opinions expressed are those of the authors and are not necessarily those of the United States Department of Justice, the Bureau of Justice Assistance, or the Office of Community Oriented Policing Services. The authors also express appreciation to the staff of the UAB Phishing Operations Team in the UAB Computer Forensics Research Laboratory.

performance (*i.e.* detection and false positive rates) of the automated approaches. Specifically, this paper details the application of popular file matching and string alignment utilities to website content in order to identify phishing attacks. While *diff* and *ssdeep* are typically used to identify file changes, the techniques have not been used as phishing countermeasures. Some of the techniques presented in this research help to overcome phishers' obfuscation methodologies and the dynamic content present in phishing websites that can elude simplistic hash matching algorithms. The results show that phishing websites can be accurately identified in near real time, allowing for prompt updates to blacklists and anti-phishing toolbars and for immediate notification to spoofed organizations.

The remainder of the paper is organized as follows: First, the methodology of the research and algorithms are described. Next, the data set is detailed, and then the results of each experiment are presented along with the pros and cons of each file matching or string alignment technique. Finally, conclusions are drawn from the results and discussed.

II. Related Work

Email filters are a commonly applied defense mechanism impeding malicious links from reaching the potential victims' inboxes. Email filters typically use statistical techniques (DSPAM, SpamAssassin, *etc.*), URL blacklists,[1] and sender email information to identify spammed emails. Furthermore, researchers have proposed a number of email-based solutions to classify the phishing emails based on the words that compose the email's body or through features extracted in the email content. Saberi et al. used Poisson filters, K Nearest Neighbor, and Naïve Bayes probabilistic theory to classify the words in the email body as ham, spam, and scam (*i.e.* phishing) [4]. However, text-based approaches against spam were believed to be not as effective against phishing as against spam in general [5] since phishing emails are designed to mimic legitimate email and thereby usually include language and keywords similar to those in legitimate email messages. Researchers have attempted to improve anti-phishing email filters through the use of other key features in phishing email messages. *Chandrasekaran et al.* derived 25 features from emails, used information theory concepts to rank the features, and finally classified the emails using a Support Vector Machine (SVM) on those features [5]. Similarly other researchers have tested the same or similar features using other classifiers such as logistic regression, classification and regression trees, random forests, neural networks, K-means, self organizing maps, and a confident weighted online learning algorithm [6] [7] [8]. While these approaches have demonstrated the ability to detect phishing emails, phishers continue to evolve their attacks to bypass such filters.

Phishers have responded to email filters by hiding misleading, unrelated content within the underlying code of the email message, spoofing the sender's email and IP addresses, and creating numerous URLs that redirect to the malicious content. Redirection is used to create a unique URL for each spam message so that blacklists are unable to keep pace with the phishing attack. But because phishing URLs are not only distributed via email message, other techniques are necessary in order to provide a complete defense against phishing.

URL-based detection techniques use features of the URLs themselves to determine whether the link is malicious [9] [10] [11]. Phishers have used a number of ways to fool victims into believing a link is legitimate. Examples include having multiple components to the hostname such as *www.bankofamerica.com.X.Y.Z.org* or *regions.com.A.B.C.com*. These long hostnames dupe victims because they see the expected brand name within the URL. Additionally, phishers incorporate legitimate paths into their link and often repeat the target brand name throughout the URL in an attempt to make it seem authentic. In response to such cases, researchers have attempted to detect phishing websites using features extracted from the URL directing the potential victims to the malicious content as well as host-based information. Examples of URL-based features include but not limited to the number of dots in the URL, length of the machine name, number of special characters, presence of hexadecimal characters or IP addresses instead of machine name, and URL length of the URL [12] [13] However, many phishing URLs do not have any of these characteristics and cannot be distinguished from any other URL. Furthermore, once a phishing website has been taken down, URL based techniques cannot determine that the content has been changed. Some researchers have investigated the use of content-based approaches to inspect whether the website that a URL refers to is malicious.

Content-based detection techniques generally download the content hosted at the URL and use features extracted from the content to identify phish. These techniques require robust website scraping techniques in order to ensure the content is sufficiently retrieved. Content-based detection can combine techniques that draw features from the text of the main index page, characteristics of sets of component files, and measures of visual similarity among websites to identify phishing attacks [14] [15] [16]. *Pan et al.* used an identity extractor to determine a websites legitimacy using textual clues such as words that appear in the DOM's title, address, and body [17]. *Xiang and Hong* proposed a similar identity extraction algorithm that employs information extraction and retrieval algorithms to differentiate websites that are claiming to be a particular website versus legitimate websites [18]. The information extraction algorithm determines word frequency in fields that commonly contain brand names such as the title and copyright field and searches for the presence of login forms. The information retrieval algorithm uses TF-IDF to rank candidate words in the documents. Finally, the brands from the fields as well as the TF-IDF results are set as inputs to the Google and Yahoo search. *Pan et al.* and *Xiang and Hong* approaches demonstrated good results but are limited to the presence of brand information in the titles and copyright fields [18]. Furthermore, their approaches may not be scalable due to performance with respect to time as they rely on search engine results and issues with consecutively querying search engines are a known problem. Content-based approaches are an effective solution because they analyze the malicious

[1] Blacklists are lists of known websites hosting malicious content. These lists are shared amongst anti-spam vendors and browser developers and used to warn users or to block navigation to malicious links.

content rather than features of an enticing email message or the link to the attack. All of the methods discussed in this research are content-based solutions that rely solely on the content hosted on the website.

III. Methodology

This research employs file matching algorithms to determine if one file or set of files can be used to classify a new file or set of files in the same category—in this case, a phishing web page. More importantly, the techniques used in this research need to perform well with respect to runtime. Previous researchers have proposed methodologies that are not practical on live anti-phishing systems. Described below are the details of a research process that results in the identification of high-performance phishing attack detection methods. The file matching and string alignment techniques tested include Main Index File MD5 matching, Deep MD5 Matching, phishDiff, context-triggered piecewise hashing using *ssdeep*, and a novel algorithm named Syntactical Fingerprinting.

The phishDiff and *ssdeep*-based methodologies require additional information to find candidate files to compare with each other, otherwise, these techniques would take too long to run to be useful techniques. Initially, the matching of titles or files of similar size was used to find these candidate files. Matching titles appeared to perform well with respect to time, whereas using the file size performed too slowly. The file size technique was slow because it gathered too many candidate files with no relation to the potential phishing web page. The initial analysis of the title and file size matching found that some files whose titles did not match were still good candidates for matching. Through this process, Syntactical Fingerprinting was developed to determine if web page source code constructs of two files were similar enough to make them good candidates for file matching algorithms. The parsing of syntactical elements and comparison of hash sets is a relatively quick process, making Syntactical Fingerprinting another potential technique used to find candidate files. Additional testing of Syntactical Fingerprinting revealed that it, too, was a potential anti-phishing algorithm; so, experiments were run using Syntactical Fingerprinting as a stand-alone file matching algorithm.

Many phishing main index files include dynamic content and references to absolute file paths that are added to the file when the phishing website is setup on the web server. This content will cause mismatching against both simplistic and fuzzy hash value functions. Additionally, individual phishers add their edits to the files such as changing cases of letters and adding whitespace to further distinguish their files from the root files they are copying. Examples of the dynamic content and edits in files can be observed in the bold black font within the two webpage's source code displayed in Figure 1. Therefore, for the experiments, files and constructs are pre-processed, removing URLs and any whitespace and converting all letters to lowercase. The preprocessing steps were used on every method except for Deep MD5 Matching.

A. Main Index Matching

One technique used by organizations to identify new phishing websites is to compute a hash value for the main index page of a potential phishing website and compare it to previously identified phish. This technique employs a hashing function, usually cryptographic with sparse collisions, to compute a unique value representative of the file and compare that value against previously confirmed phishing main index MD5 hash values. If the hash value of the potential phishing web page matches any hash value of a confirmed phishing web page, then the files are considered to be identical, thus identifying the potential website as a phish.

B. Deep MD5 Matching

Deep MD5 Matching is a technique developed by *Wardman and Warner* [16] to overcome obfuscation and dynamic content that is added to the main index files to render exact matching useless. Many phishing websites consist of file sets that produce the look and feel of the website. Additional files often hosted with the main index page are image files, cascading style sheets, and PHP and JavaScript files. Comparison of these file sets reveal relationships and is implemented in the UAB Phishing Data Mine to automatically confirm phishing websites [19].

Deep MD5 Matching computes a similarity coefficient between a potential phishing website's file set and confirmed phishing website file sets. There are a number of similarity coefficients that could be used to determine the similarity between sets of files, including the Jaccard, Kulczynski 2, and Simpson coefficients. The Jaccard coefficient is defined as the intersection of sets divided by the union of the sets, whereas, the Simpson coefficient is calculated as the intersection divided by the size of the minimal set. The Kulczynski 2 coefficient measures the average of the proportion of matching files in the two sets of files and is preferred because this coefficient gives equal weight to each set. Each of the similarity coefficients is tested in this research.

Selection of the threshold values for the similarity coefficients is an important step when implementing Deep MD5 Matching because changing the threshold values has a significant effect on detection and false positive rates.

C. phishDiff

The ubiquitous Unix command line tool *diff* was used to implement a technique called phishDiff that computes the percentage of different lines between two files. The base algorithm for *diff* is a fast implementation of the longest common subsequence problem that was originally developed by Douglas McIlroy [20]. Software was developed to compute the percentage of different lines of the main index files of phishing websites. The percentage of different lines was varied to determine if better performance could be achieved. The percentages used in phishDiff were 20%, 30%, and 50%.

D. Context-triggered piecewise hashing

Context-triggered piecewise hashing is an algorithm implemented by Kornblum [21] as a forensic tool for determining whether files are similar enough to be considered

the same. It is based on the technique *spamsum* developed by Andrew for identifying spam email [22]. The file comparison tool is named *ssdeep* [23] and employs a rolling hash[2] that computes a fuzzy hash value for each file. Sets of file hashes are compared using a string distance function. Context-triggered piecewise hashing does not perform as fast as creating a cryptographic hash and may decrease performance by 7 to 10 times [24].

E. Syntactical Fingerprinting

A novel technique called Syntactical Fingerprinting is used to compare structural components, or constructs, within files to determine whether the files are similar enough to be of the same provenance and thus belong to the same file family. These source code constructs can be standard sections of the file such as the forms, tables, and JavaScript often used in phishing HTML or PHP files. The algorithm for Syntactical Fingerprinting is as follows:

```
Input: potential phishing URL (D),
confirmed phishing construct hash set
(HS), threshold value (tAST)
Output:  Labels for potential phishing
URLs
for each URL U_i in D do
  mainPage_i = get_main_page(U_i)
  segmentSet S =
parse_segments(mainPage_i);

      for each seg in S do
            H >> compute_MD5(seg)

      simCoef = compute_similarity(H, HS)
      if simCoef >= tAST then
            confirmPhish(U_i);
end
```

In the *compute_similarity* method, the set of file component hash values from the potential phishing website is compared to sets of file constructs of previously confirmed phishing websites using the value of their Kulczynski 2 coefficient [25]. The Kulczynski 2 coefficient is expressed in the equation below where *a* is the number of matching file component MD5s between the sets 1 and 2, *b* is the number of elements in set 1 that are unique to set 1, and *c* is the number of elements in set 2 that are unique to set 2.

$$Kulczynski\ 2 = \frac{1}{2}\left[\frac{a}{a+b} + \frac{a}{a+c}\right]$$

The result of evaluating Equation 1 is a similarity value for the two file component sets. The Kulczynski 2 similarity coefficient was selected over other coefficients because each set should have equal weight to the percentage of matching components and not discriminate against either file's component set. In these experiments, the threshold for Syntactical Fingerprinting was set at 10%, 50%, and 85%.

[2] "Rolling hashes" refers to a sliding window approach where a hash is continually computed over byte segments between position P and P + N, where N is the size of the window. The rolling hash function continually updates the hash value based on the previously computed hash value and the hash value of the window.

IV. DATA SET

A critical component of testing all of the methodologies described above is the availability of an appropriate data set. The UAB Phishing Data Mine [19] is used to acquire potential phishing URLs that are submitted via a number of source feeds. These URLs are de-duplicated[3] within the system and inserted into a database. The next step in the process is an attempt to download the content files associated with the potential phishing website using GNU Wget 1.11.4 Red Hat modified. Downloading this content grants researchers the ability to test techniques on the downloaded data, and thus researchers can replay experiments and tune threshold values to achieve better performance.

HSBC	**2,323**	**eBay**	**217**
Chase Bank	**1,053**	**Santander**	**200**
Bank of America	**904**	**FNB (South Africa)**	**154**
PayPal	**766**	**Egg Bank**	**150**
Alliance & Leicester	**473**	**Standard Bank**	**150**
Lloyds TSB	**471**	**HM Revenues & Customs**	**142**
Halifax	**414**	**Craigslist**	**127**
NatWest	**275**	**Bank of Montreal**	**106**
Bradesco	**273**	**ANZ Bank**	**96**
Caixa	**267**	**MasterCard**	**88**

Table 1: The 20 most-phished organizations and their URL count in the data set.

The data set used in this research was collected from August 7 through October 31, 2010 and consists of 88,331 URLs and associated website content files[4] gathered from a number of source feeds. Of these 88,331, there were 49,840 URLs for which website content could be downloaded. Therefore, these 49,840 URLs are used as the data set for the content-based techniques examined in this work. The data set was compared with labels made by the security company, Cyveillance, to ensure the accuracy of the data. If a URL was marked as a phish by both UAB and Cyveillance, then the URL was not manually reviewed for accuracy. However, after removing the overlapping URLs, the rest of the data set was manually reviewed to ensure accuracy. The results showed that 17,992 of the 49,840 URLs were phish targeting 159 different organizations while the other 31,848 websites were non-phish. The size and diversity of this data set is an important part of this research as other researchers have used data sets [6] [7] [26] [27] [28] that consist of a lower number of phish targeting not as many organizations. Table 1 shows the twenty most-phished brands in the resulting set of phishing URLs. The total number of website content files in the data set is 343,688.

The training data set, or the website files used in the comparisons, consisted of all URLs that were either manually or automatically confirmed in the UAB Phishing Data Mine from August 2 through October 31, 2010. The labels and

[3] De-duplication refers to removing duplicate URLs already present in the UAB Data Mine. Duplicate URLs are those that contain the same domain name, path, and filename, ignoring varied parameter values that may follow the filename after a question mark or ampersand.

[4] A recursive download was used to retrieve the website content files. However, files residing on web servers other than the domain hosting the phishing content were not downloaded.

mislabels for these URLs demonstrate how using such a training data set would work on an active system, not a predetermined theoretical one with perfect labels. Additionally, the corrected, labeled URLs from this data set were added after each day to mimic a daily batch system in which confirmed URLs are added on a daily basis.

V. Results

The tables and graphs below present the performance for identifying phish from the suspicious URLs in the data set. Note that DR stands for the detection rate, while FP is the false positive rate. Each algorithm performs better with respect to detection rate when the files are pre-processed by removing whitespace and anchored URLs and by changing all of the letters to lowercase.

Technique	Total Data Set	
	DR	FP
Main Index	2.4%	0.4%
Main Index (Pre-processed)	15.2%	0.1%

Table 3: The results of Main Index Matching on the data set.

Table 3 contains the results of Main Index Matching on the data set. Pre-processing of the files for Main Index Matching demonstrated over a 7 times increase in the detection rate as well as a decrease in the false positive rate. The benefit of using this methodology is the ease of implementation and the speed of the algorithm.

Technique	Total Data Set	
	DR	FP
Jaccard	27.4%	0.8%
Kulczynski 2	29.7%	0.9%
Simpson	33.0%	0.9%

Table 4: Results of Deep MD5 Matching with a 75% threshold.

Table 4 contains the results of using Deep MD5 Matching using a 75% threshold for the similarity coefficient. The best detection rate occurred when using Simpson coefficient. Although the detection rates seem low, Deep MD5 Matching correctly identified 82.7% of phish in cases where the file set contained more than one file. There was no significant difference (less than 0.02% for DR and FP) between the threshold values of 50% and 75%; however, when raising the threshold from 75% to 85%, the detection rate decreased by nearly 2.5%.

The phishDiff technique takes advantage of the common Unix utility by using some of *diff*'s command line parameters to pre-process the files (*i.e.* whitespace removal and converting the file to case-insensitive) and greatly outperformed the previously discussed techniques. Figure 2 (on the last page of document) displays the results of the phishDiff experiment. The experiment showed that by increasing the threshold value, or number of different lines, from 20% to 50%, the detection rate increased by nearly 7% yet increased the false positive rate by only 0.4%. Additionally, the pre-processing file step increased the total detection rate at a 50% threshold by 4.5% while only increasing the false positive rate by 0.1%. The phishDiff technique also performed better with respect to detection rate as the training data set or candidate file pool grew in size (*i.e.* as the data set progressed from month to month).

Technique	Total Data Set	
	DR	FP
ssdeep (Title)	83.1%	1.9%
ssdeep (AST)	93.3%	2.9%

Table 5: Results of *ssdeep* matching using only main index pages.

The *ssdeep* implementation using matching titles as a means for finding candidate files achieved an 83.1% detection rate while having a relatively low false positive rate. However, when using the Syntactical Fingerprinting methodology to find the candidate files, *ssdeep* achieved a 93.3% detection rate with a 2.9% false positive rate.

As observed in Figure 3 (on the last page of document), varying the threshold values for Syntactical Fingerprinting had an impact on both the detection and false positive rates. There was a 6%-7% increase in the detection rate when lowering the threshold from 85% to 10% in both experimental runs. The false positive rates for the 85% threshold were 1.9% and 2.0%, respectively, whereas the false positive rates for an 85% threshold increased 12.5% and 13.5% in the two experiments. Such high false positive rates are considered unacceptable for enterprise anti-phishing solutions. There were 1,981 websites (11%) in this data set whose main index page did not contain any AST constructs. This may indicate that these website files need to be examined to find additional constructs to add to the algorithm.

VI. Discussion

The main index page matching techniques had the lowest detection and false positive rates among all experiments. It was observed that pre-processing the main index page as described above in Methodology made a significant improvement in the detection rates for each technique. The main reason for false positives, which applies to other tested techniques, was due to the website fetching method used to download the phishing content. The mis-fetched content occurred when humans confirmed a website as a phish, but, when content was later requested, it had already been removed. In such instances, the automated fetch returned common web server response pages instead. Examples of these default web pages are the Apache web server file displaying the message "No Cookie For You" or a web page displaying information about the web hosting company of the domain.

The website content file downloader only retrieved files that made up the website when the files were hosted on the web server that is hosting the phishing website. Deep MD5 Matching showed a good ability to detect phishing websites that had more than one file downloaded. Analysis showed that Deep MD5 Matching was able to detect 82.7% of websites in which more than one file was downloaded. One downside of this technique is its poor performance in case where only one file was downloaded, as the algorithm essentially reduces to Main Index Matching when only one file is downloaded. The speed in which this algorithm runs, its good detection rate when more than one file is downloaded, and its less-than-1%

false positive rate makes it a good candidate for combination with other anti-phishing techniques. Even though Deep MD5 Matching using the Simpson coefficient outperformed the Kulczynski 2 coefficient, it is believed that brand identification results are more accurate using the Kulczynski 2 coefficient.

The phishDiff experiments demonstrated the ability to robustly detect phishing content. When pre-processing the files, the detection rate increased an average of nearly 4.0%, while the false positive rate only increased 0.1%. Additionally, there was a significant increase in detection rate, while not causing a significant increase in the false positive rate when adjusting the overlapping line threshold from 80% to 20%. This indicates that the phishDiff should use file pre-processing and can employ lower threshold values for better performance in detection while not greatly increasing the false positive rate.

The phishDiff and *ssdeep* implementations demonstrated the importance of finding good candidate files to compare the potential phishing web page against. In these experiments, Syntactical Fingerprinting outperformed title matching in detection rate and made no significant increase in the false positive rate. In fact, the *ssdeep* implementation using Syntactical Fingerprinting for candidate file selection had the best overall error rate. The reason for the better detection rate is the candidate file selection pool size that the algorithms use. It has been noted that phishers often make small changes to the source code in order to avoid detection. The title of the webpage is one commonly edited feature of the source code. Therefore, title matching may not find all candidate files to compare against. This indication is what led to the development of Syntactical Fingerprinting for file candidacy. An analysis of the results showed that Syntactical Fingerprinting found more candidate files to compare against while not hindering the techniques with time performance issues.

There is a 1% difference between the false positive rates of *ssdeep* using titles versus using Syntactical Fingerprinting for finding candidate files. Analysis of the 1% difference showed that using common constructs often finds more candidate files to compare against thus presenting more opportunities for mislabels. These experiments also demonstrate how advertising websites for target organizations often reuse code from the legitimate organizations website. A large percentage of the false positives that occurred within the 1% difference were advertising for two organizations, Sainsbury's and ato.gov.au websites.

The results of the Syntactical Fingerprinting experiments had the best overall detection rates and had manageable false positive rates. The only false positive that is too high for most anti-phishing situations is when the threshold is set to 10%. In the analysis of false positives and negatives, the limitations of the current implementation of Syntactical Fingerprinting become apparent.

The false negatives for this approach were primarily due to two factors. The first is that new source code or spoofed organizations were introduced in the test set that were not present within the training data. These files were not modifications to previously seen main phishing web pages. The detection rate is expected to improve as more confirmed phishing web pages are used for comparisons. Another contribution to false negative labels has to do with a problem with syntactical elements used in the approach. The current syntactical elements were not considering elements that were capitalized, meaning that the algorithm searched for a *<table>* tag but not a *<TABLE>* tag, for example. Minor modifications can be made to catch such instances. The addition of other syntactical elements not used in these experiments may also improve AST detection.

The false positive rate for Syntactical Fingerprinting was mainly affected by phishers reusing components from legitimate websites. Phishers reuse generic JavaScript functions and tables and forms from the spoofed organization to make the website look and feel as close as possible to the spoofed organization's authentication website. Future work may show that certain common constructs could be given less weight in the matching algorithm to help reduce false positive labeling. Furthermore, the false positive rate of Syntactical Fingerprinting can be reduced in the future by employing whitelists and Google's PageRank. Previous researchers [14] [29] [30] have shown the ability of these methodologies to reduce the false positive rate while not affecting the detection rate.

VII. Future Work

Future work on this topic will test an ensemble approach using weights contributed by each algorithm or additional techniques could be implemented for better results. It is anticipated that the combination of the content-based approaches present in the experiments within this research could demonstrate even better detection rates. Preliminary results show that using the OR operator in an ensemble approach using Deep MD5 Matching, *phishDiff*, and Syntactical Fingerprinting outperformed each single technique. There are a number of other ensemble approaches that could be tested such as AND, majority vote, and weighted voting.

Moreover, this ensemble approach could use other techniques, such as URL-based described by previous researchers, to improve the ensemble. Researchers mention that the use of whitelists and Google's PageRank can reduce false positive rates. Such techniques and others could be utilized within a weighted ensemble approach to ensure robustness.

Syntactical Fingerprinting, which is similar to Deep MD5 Matching, has demonstrated the ability to identify websites based on similar file constructs. This algorithm can be applied as a distance metric, similar to *Wardman et al.* [19], to cluster websites. These clusters can help investigators to better understand trends in phishing activity.

In addition, a suggestion is to determine the relationship between adding and removing the pre-processing steps used in Syntactical Fingerprinting. For example, it is hypothesized that some individual phishers make specific edits to the files that are characteristics of that phisher such as adding a space at the end of lines, capitalizing specific alphabetic characters throughout a document, or putting an extra return carriage after every N lines of code. Future research should investigate how these changes interact and how they can be used to define

file provenance. The process would consist of creating clustering views using Syntactical Fingerprinting at each pre-processing step as well as views of permutations of the pre-processing steps. Better understanding of these relationships may present investigators and researchers with a greater knowledge of phishing activity.

VIII. Conclusions

The experiments conducted in this research demonstrate a variety of string matching algorithms to quickly and accurately identify phishing websites using content-based approaches. Some techniques displayed minor impact on detection, while others provided a robust framework for identifying phishing content. These techniques were tested on a large, diverse set of phishing websites targeting over 150 different organizations, and the experiments showed that some of the techniques could achieve better than 90% detection rates while maintaining reasonable, manageable false positive rates.

References

[1] Jakobsson, M., & Myers, S. (2006). *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft.* Hoboken, NJ: Wiley-Interscience. Hoboken, NJ : Wiley-Interscience, 2006.

[2] Cluley, G. (2011, February 16). *Steam phishing targets video game players | Naked Security*. Retrieved March 28, 2011, from Naked Security: http://nakedsecurity.sophos.com/2011/02/16/steam-phishing-targets-video-game-players/.

[3] APWG. "Phishing Activity Trends Report 2010." (July 30, 2011).

[4] Saberi, A., Vahidi, M., & Bidgoli, B. M. (2007). Learn to Detect Phishing Scams Using Learning and Ensemble Methods. *Web Intelligence and Intelligent Agent Technology Workshops* (pp. 311-314). Silicon Valley, CA: IEEE.

[5] Chandrasekaran, M., Narayanan, K., & Upadhyaya, S. (2006). Phishing E-mail Detection Based on Structural Properties. *New York State Cybersecurity Conference Symposium on Information Assurance: Intrusion Detection and Prevention*, (pp. 2-8). Albany, NY.

[6] Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007). A Comparison of Machin Learning Techniques for Phishing Detection. *eCrime Researchers Summit* (pp. 60-69). Pittsburgh, PA: APWG.

[7] Basnet, R., Mukkamala, S., & Sung, A. H. (2008). Detection of Phishing Attacks: A Machine Learning Approach. *Studies in Fuzziness and Soft Computing 226* (pp. 373-383). Springer-Verlag.

[8] Basnet, R., & Sung, A. (2010). Classifying Phishing Emails Using Confidence-Weighted Linear Classifiers. *International Conference on Information Security and Artificial Intelligence (ISAI)* (pp. 108-112). IEEE.

[9] Blum, A., Wardman, B., Solorio, T., & Warner, G. (2010). Lexical feature based phishing URL detection using online learning. *3rd Workshop on Artificial Intelligence and Security.* Chicago, IL.

[10] Ma, J., Saul, L., Savage, S., & Voelker, G. (2009). Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. *KDD '09.* Paris, France: ACM.

[11] Ma, J., Saul, L., Savage, S., & Voelker, G. (2009). Identifying Suspicious URLs: An Application of Large-Scale Online Learning. *Proceedings of the 26th International Conference on Machine Learning.* Montreal, Canada.

[12] Chen, J., & Guo, C. (2007). Online Detection and Prevention of Phishing Attacks. *Communications and Networking in China* (pp. 1-7). Beijing: IEEE.

[13] Garera, S., Provos, N., Chew, M., & Rubin, A. (2007). A Framework for Detection and Measurement of Phishing Attacks. *WORM '07*, (pp. 1-8). Alexandria, VA.

[14] Dunlop, M., Groat, S., & Shelly, D. (2010). GoldPhish: Using Images for Content-Based Phishing Analysis. *The Fifth International Conference on Internet Monitoring and Protection* (pp. 123-128). IEEE.

[15] Suriya, R., Saravanan, K., & Thangavelu, A. (2009). An Integrated Approach to Detect Phishing Mail Attacks A Case Study. *SIN '09* (pp. 193-199). North Cyprus, Turkey: ACM.

[16] Wardman, B., & Warner, G. (2008). Automating Phishing Website Identification through Deep MD5 Matching. *eCrimes Researcher Summit.* Atlanta, GA: IEEE.

[17] Pan, Y., & Ding, X. (2006). Anomaly Based Phishing Page Detection. *ACSAC 06'* (pp. 381-392). Washington D.C., USA: IEEE.

[18] Xiang, G., & Hong, J. (2009). A Hybrid Phish Detection Approach by Identity Discovery and Keywords Retrieval. *WWW 09'* (pp. 571-580). Madrid, Spain: ACM.

[19] Wardman, B., Warner, G., McCalley, H., Turner, S., & Skjellum, A. (2010). Reeling in Big Phish with a Deep MD5 Net. *Journal of Digital Forensics, Security and Law. 5(3).*

[20] Hunt, J., & McIlroy, M. D. (1976). *An Algorithm for Differential File Comparison.* Computing Science Technical Report, Bell Laboratories.

[21] Kornblum, J. (2006). Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation 3* , 91-97.

[22] Andrew, T. (2002). *Spamsum README.* Retrieved from http://samba.org/ftp/unpacked/junkcode/spamsum/README.

[23] *Fuzzy Hashing and ssdeep*. (n.d.). Retrieved July 8, 2011, from SourceForge: http://ssdeep.sourceforge.net/.

[24] Hurlbut, D. (2009). *Fuzzy Hashing for Digital Forensic Investigators.* AccessData.

[25] Kulczynski, S. (1927), "Die Pflanzenassoziationen der Pienienen," Intern. Acad. Pol. Sci. Lett. Cl. Sci. Math. Nat, Ser. B, 1927(2): 180.

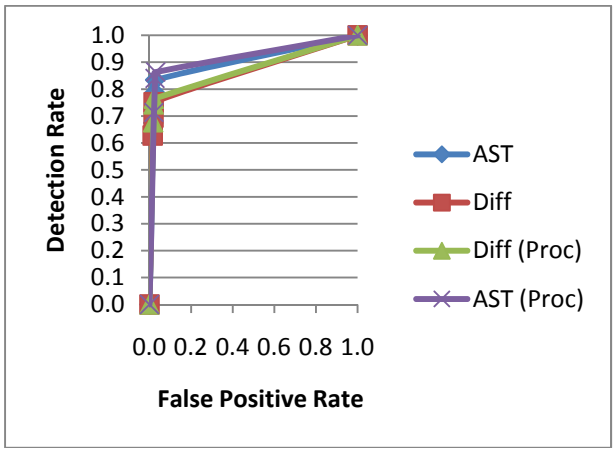
[26] Bergholz, A., De Beer, J., and Glahn, S. (2010). New Filtering Approaches for Phishing Emails. *Journal of Computer Security. 18(1)*, 7-35.

[27] Fette, I., Sadeh, N., and Tomasic, A. (2007). 'Learning to detect phishing emails'. International Conference on World Wide Web. May 8-12, 2007. Banff, Alberta, Canada.

[28] L'Huillier, G., Weber, R., and Figueroa, N. (2009). 'Online Phishing Classification Using Adversarial Data mining and Signaling Games'. ACM SIGKDD Workshop on Cybersecurity and Intelligence Informatics. June 28, 2009. Paris, France.

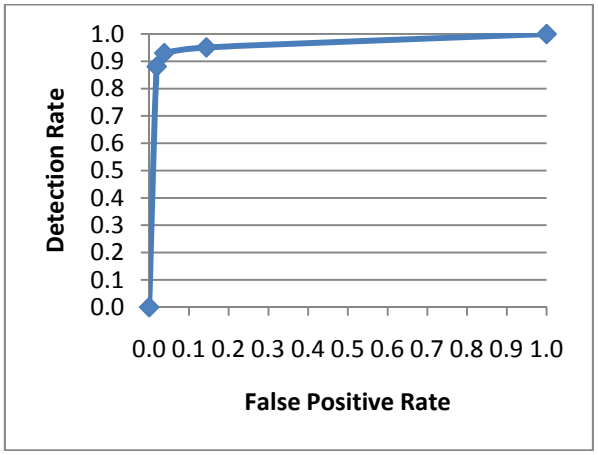
[29] Whittaker, C. Ryner, B., and Nazif, M. (2010). 'Large-Scale Automatic Classification of Phishing Pages'. Network and Distributed Systems Security Symposium. February 28-March 3, 2010. San Diego, CA.

[30] Zhang, Y., Hong, J.I., and Cranor, L.F. (2007). 'Cantina: A Content-based Approach to Detecting Phishing Web Sites'. International Conference on World Wide Web. May 8-12, 2007. Banff, Alberta, Canada.



Technique	Total Data Set	
	DR	FP
phishDiff (Title 20%)	75.1%	2.0%
phishDiff (Title 50%)	69.4%	2.0%
phishDiff (Title 80%)	62.8%	1.5%
phishDiff (Title 20%, Proc)	83.4%	2.7%
phishDiff (Title 50%, Proc)	78.8%	2.5%
phishDiff (Title 80%, Proc)	67.5%	1.9%
phishDiff (AST 20%)	86.4%	3.0%
phishDiff (AST 50%)	84.0%	2.7%
phishDiff (AST 80%)	71.4%	2.2%
phishDiff (AST 20%, Proc)	76.4%	2.1%
phishDiff (AST 50%, Proc)	74.3%	2.0%
phishDiff (AST 80%, Proc)	67.7%	1.8%

Figure 2: The ROC curve and the detection and false positive rates of each phishDiff experiment at select thresholds



Technique	Total Data Set	
	DR	FP
AST (85%)	88.1%	1.9%
AST (50%)	93.0%	3.8%
AST (10%)	95.1%	14.4%

Figure 3: The ROC curve and the detection and false positive rates of the AST experiments at select thresholds.