

# A Method For The Automated Detection Of Phishing Websites Through Both Site Characteristics And Image Analysis

Joshua S. White, Jeanna N. Matthews, John L. Stacy  
Clarkson University, Potsdam, New York 13699  
{whitejs, jnm, stacyjl} @clarkson.edu

## Abstract:

Phishing website analysis is largely still a time-consuming manual process of discovering potential phishing sites, verifying if suspicious sites truly are malicious spoofs and if so, distributing their URLs to the appropriate blacklisting services. Attackers increasingly use sophisticated systems for bringing phishing sites up and down rapidly at new locations, making automated response essential. In this paper, we present a method for rapid, automated detection and analysis of phishing websites. Our method relies on near real-time gathering and analysis of URLs posted on social media sites. We fetch the pages pointed to by each URL and characterize each page with a set of easily computed values such as number of images and links. We also capture a screen-shot of the rendered page image, compute a hash of the image and use the Hamming distance between these image hashes as a form of visual comparison. We provide initial results demonstrate the feasibility of our techniques by comparing legitimate sites to known fraudulent versions from Phishtank.com, by actively introducing a series of minor changes to a phishing toolkit captured in a local honeypot and by performing some initial analysis on a set of over 2.8 million URLs posted to Twitter over a 4 days in August 2011. We discuss the issues encountered during our testing such as resolvability and legitimacy of URL's posted on Twitter, the data sets used, the characteristics of the phishing sites we discovered, and our plans for future work.

**Keywords:** Phishing, Image Analysis, Characteristic Analysis, Social Media

## 1. INTRODUCTION

Phishing [1] is the act of convincing users to give up critical personal information, either through conversation or some form of content manipulation. Most modern day phishing attacks occur by luring users into visiting a malicious web page that looks and behaves like the original. Once there, the user, if convinced that the page is authentic, may give up private information including authentication credentials or banking information. This information is typically used to commit some form of identity theft or fraud.

The most common methods used today for the detection and analysis of phishing web sites are:

- **Manual view and report services such as Phishtank.com [4].** This is by far the most common method which first, requires actual users to find, identify, and report suspicious web-pages and second requires, additional people to verify the status of reported pages.
- **Correlating links in known SPAM email to phishing sites [2].** Industry standard SPAM email detection techniques are used to identify SPAM email and then links from those emails are examined. Typical examination includes looking for URL redirection, or a variety of DNS tricks that might signify a phishing site. The emphasis is typically on examining characteristics of the URL itself. Unless a URL is malformed or some other SPAM-like characteristic is identified, phishing sites are often not identified in this way.
- **Crawler classification of websites.** Pages themselves are examined for suspicious characteristics like misspelled words, link obfuscation, right-click menu disabling, DNS redirection, etc. [3]. This method is similar to our own in that it places a strong emphasis on fetching and analyzing the actual page rather than just the URL. However, it looks for suspicious content in individual pages where we focus on identifying groups of pages that would “look” the same to an unsuspecting user.

## 2. OUR METHOD

Our work takes inspiration from this manual process of finding pages that “look the same”. Specifically, we automate some of what the human does to recognize duplication of an original page. To do this we analyze a web page based on some of its structural characteristics and based on the way it looks visually. Specifically, we record a number of page markup characteristics including the page title text and number of links, images, forms, iframes and metatags. Throughout the rest of this paper we will refer to this 5 tuple as a pages structural fingerprint. The image analysis portion of this work is at it's simplest is done by setting a fixed dimension and quality setting for rendering a page within a headless browser and then taking a page screen-shot. We compute a hash of the resulting image and compare the hash values using the Hamming Distance equation. This process takes on average 4 seconds a page including software build-up/tear-down time. To speed up the process further, we could prioritize image analysis for pages that with matches in the most easily parsed/computed items from the pages markup.

Figure 1 shows a high-level overview of the full process. The first step is the real-time gathering of URL's from social media sites like Twitter. We used a modification of the `get_tweets.php` script provided by the 140Dev Twitter tool [5] to fetching the raw JSON version of each tweet and store it in a MySQL table. Next, we parse the tweets looking for URL's. Since Twitter requires all URL's to begin with the standard “HTTP://” to be hot-linked it was as simple as parsing the JSON data for that expression. We fetch the page content for each URL using PHP's `LoadHTML` and `DOM Object` walk through functions to visit the site, load the HTML into memory and finally count up each of the specific DOM objects. Each time a page has been parsed it is then logged in the master `url_stats.csv` file for later analysis. For each new URL, we also use `XVFB` and `CutyCapt` to render the resulting page and capture a screenshot. Finally, we compute a variety of hash values on the resulting image.

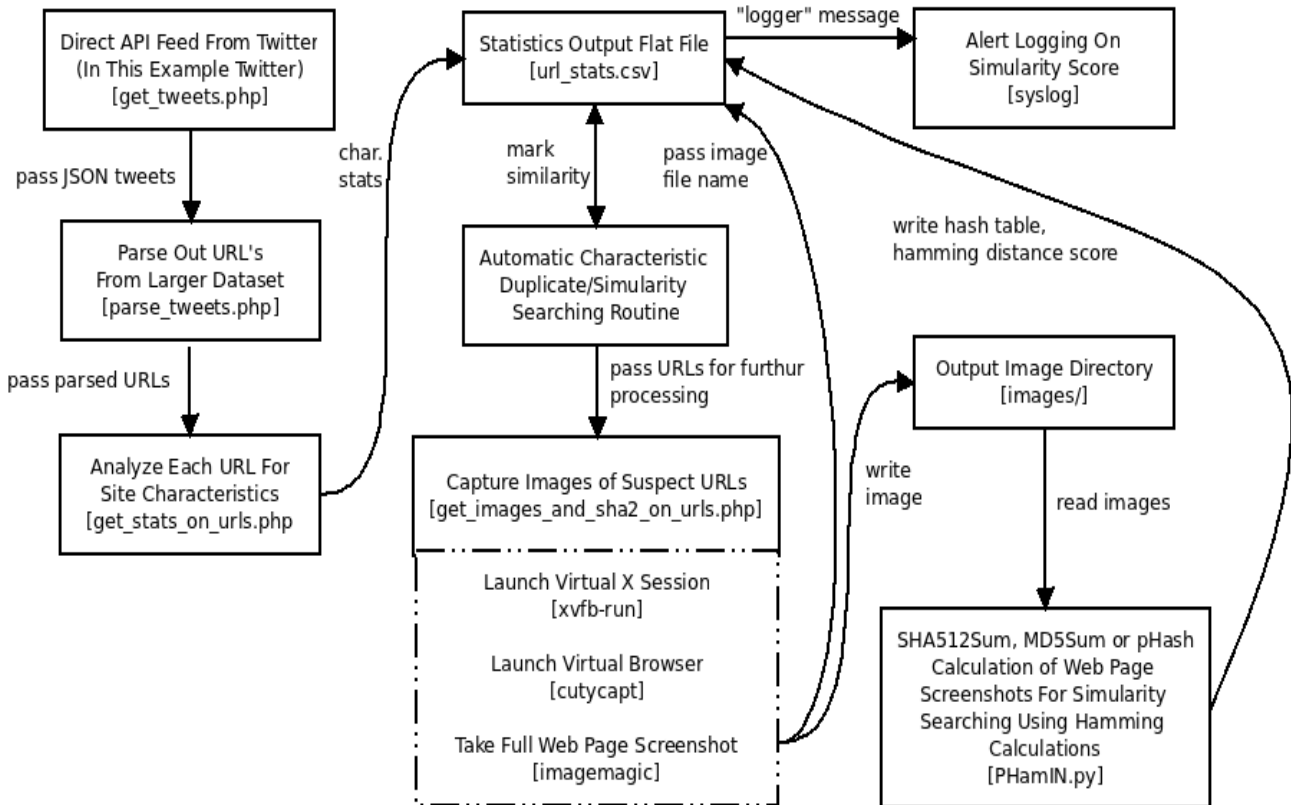


Figure 1: Phishing Detection Process Overview

## 2.1 Hash Values and Hamming Distance Image Analysis

In order to identify when two web sites look the same, we render the site, take a screen shot using CutyCapt and compute a hash of the resulting image. We compare website images by calculating the Hamming distance [14] between their hash values. Hamming distance detects the similarity of two equal strings that the exact same length. The resultant number is the minimum set of substitutions that must occur from one string to the other to make them match. For example, when comparing two 32bit MD5Sum's, the possible Hamming value would be in the range of 0 to 32 where 0 means they are identical and 32 means that every single bit is different. We experimented with a variety of hash calculation methods including cryptographic hashes like MD5, SHA512 and pHash (Perceptual Hash). pHash is a hashing algorithm specifically designed for fingerprinting multimedia files such as images that relies on discrete cosine transformation (DCT) to reduce sampling frequency in the file [15].

File Name	File pHash	File MD5Sum	File SHA512Sum	Hamming Score (pHash)	Hamming Score (MD5Sum)	Hamming Score (SHA512Sum)
clarkson-paypal-page-phishing-site-original.png	18446710817257652609	8b0914d7d2544f97b6e7a36adad92da1	95552b3e553530f5f00e476d1a64316d0f08fcc6a4d415fa77aafcd9de248afe9613363056acd0aa40db4a3f5b19ac0eb2cce5b3a808d794fb575aad000057c	0	0	0
clarkson-paypal-page-phishing-site-rev2.png	18446710817257652609	d2b931f6b29bf7dcb99601f7a7c7d12b	4d39160800e9250ded72108cdb14a73cd015f7a30d2d9e01de8836ae5073ee179cd2800cd32ea9e190ffebe1dacd11ec364e26dd5147faa77eb4a12191c67b69	0	8	38
clarkson-paypal-page-phishing-site-rev3.png	18446710817257652609	b9858446a3f38dd9e9116f545f31500f	c9d09715010886682aec120a29e2135b0b3ea49474b9ff99504026406ab7ac7ab73c3f9c570ee274d014257e5ad95f5bf5d9cb18585f45c080da400a4daff632	0	8	9
clarkson-paypal-page-phishing-site-rev4.png	18446728408906826113	94b551da15320573cd51b3ee7dcbb7fa	8029e65c7294f6c8c317b8d136ab399f04b318782c885dd16ab74b20843eec0401b99b68154fbaf5798844408414db255f7154e0569590286a711c32137fc094	2	9	8
clarkson-paypal-page-phishing-site-rev5.png	18446710816720798145	df78813010280c2fde8f5e4270b928ff	55e15dae72375e9e537aabccad76d04f85a13465846b750f85a2ae5b9d95b78c255182f57a52a51d2aab2b736b0179e59d4b2a601e291d43ff73e16b29e963b8	3	10	24
ezine-screenshot-same-characteristics-as-paypal.png	15835713822348909969	8e434d107b43941c39ac4bda2c806fbb	f3c52d2d4fe608cc6f500d661260204a9c4fff9a06c24ad39513d67dacff3ed08f5a4bdf93997f00dc7b3f4d3775ee81cbcd8eba961219fc af1c5ec67b338886	19	1	9

Table 1: Comparison of Hashing Methods and Hamming Distance Calculations on Captured Phishing Site

In Table 1, we describe a set of experiments to compare these three hashing algorithms on a sample PayPal phishing site captured in a local honeypot in December 2007. We introduced a series of minor changes to the site and computed the Hamming difference relative to the original site for each hash algorithm. Our first change was minor modifications in the page text. Specifically, we removed all cases of the word “the” from the page. Relative to the original image, the Hamming Distance for the MD5 hashes was 3, while the SHA512Sum resulted in a distance of 0 and the pHash distance was 0. After completely replacing a few sentences, we began to see larger distances with the SHA512Sums as well. However, even with a significant portion of the text changed and the color of the PayPal logo altered slightly we still received a Hamming Distance of 0 using pHash. It was not until we changed the color of multiple items on the page and rearranged things that we started to see higher distance values. This means that a phishing site would need to be significantly different from the original page that it was trying to spoof for it to go undetected with the pHash technique.

## 2.2 Method Verification Using Phishtank

We performed additional validation of our method using known phishing sites. Specifically, we identified 5 fraudulent sites found on Phishtank.com [4] and matched them with their legitimate counterparts. We deliberately chose sites that had not yet been taken down so that we would be able to run our analysis tools on them.

Table 2 shows the results of our page component parsing script on these 5 fraudulent sites and their legitimate counterparts. We matched 4 of 5 phishing pages to their legitimate counterparts using only the structural statistics we recorded. Adding the comparison of the screen-shot captures allows us to identify all 5. Other tools for used for phishing website identification report success rates in the range on average between 20% and 60% effective with only 1 tool in our literature review reaching nearly 90%. In addition the average false-positive rate for these tools is 42% [16, 17, 18].

(F)raud / (L)egit	URL	Structural Fingerprint	Page Title	pHash Value	Hamming Score
<b>EBAY Fraudulent</b>	http://fgn218.internetdsl.tpnet.pl/ws/eBayISAPI/SignIn.php?&errmsg=8&pUserId=&co_partnerId=2&siteid=0&pageType=-1&pa1=&i1=1&UsingSSL=1&k=1&bshowgif=0&favoritenav=&ru=http%3A%2F%2Fwww.ebay.com%2F&pp=	22,5,3,0,1	Welcome to eBay&nbsp;-&nbsp;&nbsp;SignIn	18415095248092893183	0
<b>EBAY Fraudulent</b>	http://fgn218.internetdsl.tpnet.pl/track.php	22,5,3,0,1	Welcome to eBay&nbsp;-&nbsp;&nbsp;SignIn	18415095248092893183	0
<b>EBAY Legitimate</b>	https://signin.ebay.com/ws/eBayISAPI.dll?SignIn&UsingSSL=1&pUserId=&co_partnerId=2&siteid=0&ru=http%3A%2F%2Fmy.ebay.com%2Fws%2FEBayISAPI.dll%3FMyEbayBeta%26MyEbay%3D%26gbh%3D1%26guest%3D1&pageType=3984	22,5,3,0,1	Welcome to eBay - Sign in	18415095248092893183	0
<b>CIBC Fraudulent</b>	http://www.shop-cafe.ru/	24,10,2,0,1	RETURNED NOTHING	13771761369894338304	0
<b>CIBC Legitimate</b>	https://www.cibconline.cibc.com/olbtxn/authentication/SignOn.cibc	24,10, 2,0,1	RETURNED NOTHING	13771761369894338304	0
<b>HALIFAX Fraudulent</b>	http://www.njcabinet.com/old_stuff/data/tmp/Halifax/index.html	21,2,2,0,7	Welcome to online banking	16716169687489823731	0
<b>HALIFAX Legitimate</b>	https://www.halifax-online.co.uk/_mem_bin/formslogin.asp?source=halifaxcouk&simigvis=Mi43Ni4yNTMwMTAyMzgZOTQ3MS4xMzEyMDQ2OTewMjAz*	24,2,2,0,7	Welcome to online banking	16716169687489823731	0
<b>Paypal Fraudulent</b>	http://si4r.com/_paypal.co.uk/webscr.html?cmd=SignIn&co_partnerId=2&pUserId=&siteid=0&pageType=&pa1=&i1=&bshowgif=&UsingSSL=&ru=&pp=&pa2=&errmsg=&runame=	0,7,1,0,2	RETURNED NOTHING	16716169687489831923	1
<b>Paypal Legitimate</b>	https://www.paypal.com/cgi-bin/webscr?cmd=_login-submit&dispatch=5885d80a13c0db1f8e263663d3faee8d1e83f46a36995b3856cef1e18897ad75	27,3,0,0,2	Redirecting - Paypal	18439707190431836096	0

Table 2: Known Phishing/Non-Phishing Site Characteristics

Our experience with these five sites illustrates the importance of using both the easy to compute structural characteristics and the image comparison via hash value. Four of the five fraudulent sites can be detected based on structural characteristics alone. Due to the nature of how some sites are copied and used for phishing, we won't always have exact matches based purely on page markup characteristics. To illustrate this we look at the analysis of Paypal and its cloned site in Table 2. In this case the fraudulent site was of very poor quality as set against phishing standards. The site did not replicate any of the functionality of the page it was spoofing other than the login form. As such, none of the other page characteristics matched up. Yet, when we look at the output of our image capture as shown in Figure 2, we can see that the page still pulls off the “authentic” look part of the site<sup>1</sup>.

<sup>1</sup>The Page Title field of the fraudulent Ebay page is formatted differently than the legitimate Ebay Sign-in page. This is easily overcome by removing the special HTML encoding characters.

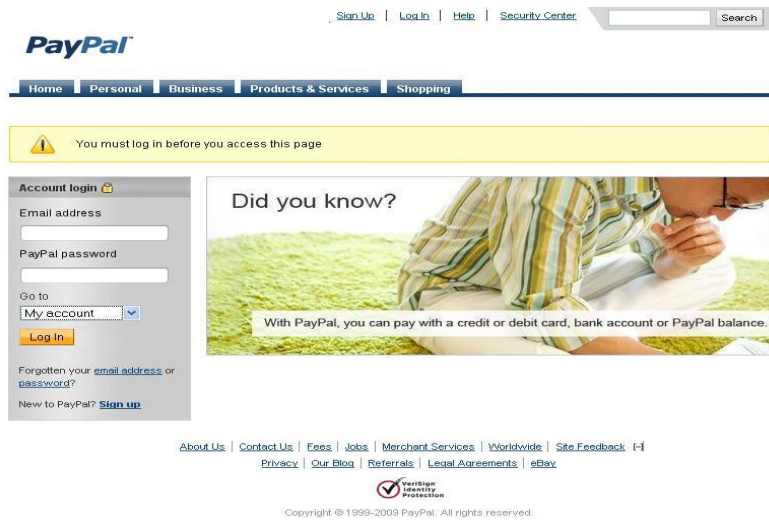


Figure 2: Example Image Capture of EBay.Com Phishing Page Using ImageMap

### 3. ANALYSIS OF LINKS FOUND IN TWITTER DATA

After verification of techniques we moved on to the larger “wild” data set of unknown URLs. Specifically, we look for URLs in Twitter posts. Previous studies have shown that 12% of all URL's posted on Twitter may be malicious. [6,7]

A real-time source of candidate phishing sites is increasingly important as attackers shorten the time a given phishing site is active before the content is moved onto a new compromised host. We also chose Twitter for the collection of URLs because it allowed for real-time interaction. Our work aims to shorten that life-cycle by providing near real-time feedback and thus needs a way to bring URL's in that are relevant and live at the time of processing. Traditional crawlers are not up to the task and we do not have the resources of companies like Google at hand. With their real-time API, Twitter offers one of the best means of potentially malicious near-real-time URLs.

We captured 81.9 GB of raw JSON data representing roughly 4 days of Twitter traffic including a total of 19,624,335 tweets. Table 3 shows the details of the dataset. In our initial experiments, we were able to capture 2,813,476 unique URL's. However, in practice we were only able to process 1,829,531 via our page characteristic technique. This was due to a large number of URL's being un-resolvable by the time we processed them. It is difficult to know in all cases why a URL was unresolvable, but we can identify a number of factors besides phishing sites that have been taken down by attackers. In Section 3.2, we elaborate on these other reasons URLs were unresolvable including typos and unavailable URL shortening services.

<b>Duration</b>	7-25-2011(13:15:45 EST) --> 7-29-2011(11:01:32 EST)
<b>Number of Tweets Captured</b>	19,624,335
<b>Number of URLs Within Tweets Captured</b>	3,902,699
<b>Number of Unique URLs From Total Tweet Dataset</b>	2,813,476
<b>Number of URLs Processed For Page Characteristics</b>	1,829,531

Table 3: URL Data-set Collected From Twitter

At this time we have not had the opportunity to sufficiently analyze the captured URL dataset from Twitter. In the future we are interested in measuring the frequency at which legitimate URL's versus Phishing URL's are posted, and how many URLs come from individual users and measuring the number of intentional versus accidental posting of Phishing URLs. In the following sections, we elaborate on some of the challenges we faced in processing and characterizing links found in Twitter data.

### 3.1 Data-Collection Issues

The total size of captured and stored data from the Twitter API is even larger than the raw feed after the JSON stream is decoded. We tried a number of popular database systems like MySQL, Hadoop and Cassandra. With each system tried, we ran into substantial problems in achieving fast access to million+ entry tables. In this prototype, we used MySQL and divided logical database tables into roughly 9 hour increments to allow for smaller table sizes with a limited number of entries in each. After decoding and importing the 81.9 GB of JSON data into MySQL, this data was slimmed down to 27.3 GB of pure content, which shows how much overhead is included in each tweet. After parsing the MySQL database and creating a new table of just URL's, we ended up with 193.5 MB of usable links.

### 3.2 URL's Processed for Page Characteristics Issues

From Table 3, we note that the number of URLs that were processed for page characteristics was inconsistent with the number of unique URLs gathered. In this section, we discuss some of the reasons for this difference:

- After manual investigation, it appears that many of the URLs are bad/incomplete from the moment they are posted. For example, a URL might end with .ocm instead of .com. We suspect simple typos. Without analyzing all of the URLs, we can't put a percentage on the number of typographical error related failures across the entire data set, but an evaluation of 1000 non processable URL's revealed that roughly 20% were due to typographical error.
- The next largest contributor was expired or non-working shortening services. A great example of this was goo.gl which is fairly new at the time of this research. The goo.gl shortening service is a google labs program that has been plagued by outages. At the time of the 1000 URL manual analysis, failed expand URL shortening caused roughly 20% of all the non-processed pages.
- The remaining 60% of non-processed manually examined URLs were simply unreachable in one way or another, some had unresolvable DNS entries and some returned "page not found" errors. As we discuss later, we used a page load time of 3000 ms and suspect that some tuning of that value may be necessary. In future work, we plan to explore a wider range of time out values and investigate the cause of truly unresolvable URLs in more detail.

### 3.3 URL's as Images & Image Collection Issues

Image collection was done using a combination of two tools, CutyCapt [8] and XVFB [9]. The first of which, CutyCapt, takes screen-shots of a fully loaded web-pages by launching a Webkit based GUI-less web-browser. XVFB was used to provide CutyCapt with a virtual frame buffer, since the server this entire processes resided on did not have an X server installed. The problems involving these two pieces of software are as follows:

- CutyCapt can only process one URL at a time and launches a new instance of its self every time it does so. On a headless system, this requires XVFB to launch a virtual X session every time CutyCapt processes a URL. In conjunction with the additional time required to launch, there is an associated tear down time for that X session. If a delay is not included at the end of the CutyCapt URL parsing loop, then an X session collision occurs and will keep occurring for every link read into CutyCapt until the X session fully closes. To deal with the previous issue of X session build up and tear down time, we attempted to run multiple instances of XVFB and CutyCapt simultaneously on different groups of the URLs. The basic concept of this was to split the URL input file into equal pieces and when requesting the Image capturing script we would then increment the virtual display number in XVFB. Each instance could then append their results to the output CSV file. However this proved unreliable since XVFB uses such a high percentage of system resources when starting up. More than two instances of XVFB running at one time slowed the Quad-Core test system down to a crawl.
- Website load time was the last major contributing factor to the limited number of URLs processed for image comparison. We put in a hard limit of 3000ms for a page to load before CutyCapt would simply stop and move onto the next, this proved insufficient for many pages. There is of course a trade-off between the timeout value and our ability to process all URLs in real-time.

In the future, we would like to explore modifying CutyCapt to add multi-threaded support and X session reuse to the code base. By reusing a single X session we would reduce the multiple seconds of latency that is incurred each time that build up and tear down occurs. Adding multi-threaded support that incorporates virtual X session reuse would be even more beneficial. We would be able to increase load time wait in CutyCapt to support slow sites, without waiting for one site to complete, before processing another.

### 3.4 Twitter Link Analysis Results

Now we can look at the analysis of our known fake/non-fake Ebay pages as outlined in Table 4. Ebay is a great example for this work since it at the moment it is one of the top spoofed sites. [10,11] Table 3 shows significant correlation of both the structural site characteristics and image hashes between the site characteristics between the fake pages pointed to in our Twitter data set and the legitimate Ebay site. Table 4 shows the page screen-shots as well as pHash values of the real and fake Ebay login Pages. Notice that the hashes are identical for the legitimate and fraudulent sites.

The system that we have build currently allows for exact match searching, that is, results that have an exact characteristic match and/or exact distance score. In addition we have the ability to search for other combinations of characteristic measurements and specific distance scores. This allows us to search on less precise matches. We used this to search for exact matches of the top 5 most phished sites and Table 5 lists the number of exact matches for each. This could also be used by the administrator of any web site to search for matches to their site.

Table 6 also lists the characteristics of what has been reported as the top most copied main site pages in the last year. These characteristics were used to complete a basic search of the data set.




Status	URL	Screen-shot	Structural Fingerprint	pHash
Fake	http://fgn218.internetdsl.tpnet.pl/ws/eBayISAPI/SignIn.php?&ermsg=8&pUserId=&co_partnerId=2&siteid=0&pageType=-1&pa1=&i1=1&UsingSSL=1&k=1&bshowgif=0&favorite nav=&ru=http%3A%2F%2Fwww.ebay.com%2F&pp=		22,5,3,0,1	18415095248092893183
Fake	http://fgn218.internetdsl.tpnet.pl/track.php		22,5,3,0,1	18415095248092893183
Legitimate	https://signin.ebay.com/ws/eBayISAPI.dll?SignIn&UsingSSL=1&pUserId=&co_partnerId=2&siteid=0&ru=http%3A%2F%2Fmy.ebay.com%2Fws%2F%2F%2FmyEbayBeta%26MyEbay%3D%26gbh%3D1%26guest%3D1&pageType=3984		22,5,3,0,1	18415095248092893183

Table 4: Phishing and Non-Phishing Site Image Hashing Comparison

Using just the structural characteristics, we found a number of false positives. For example, we found other links that contained the same pattern of tag information, such as with Paypal's "71,6,3,0,3." Interestingly, EzineArticles.com sub-pages like: <http://EzineArticles.com/6429341> use a strict format template that happens to have characteristic matches to Paypal.com's landing page at the time of these experiments. However the addition of the screen-shot image information correctly identifies this false positive.

Site	Characteristic Map	pHash	Distance Score
<a href="http://EzineArticles.com/6429341">http://EzineArticles.com/6429341</a>	71,6,3,0,3	15835713822348909969	24
<a href="http://www.paypal.com">http://www.paypal.com</a>	71,6,3,0,3	18446677844303266811	0

Table 5: False Positive Elimination Using Hash Distance Scoring

As Table 6 shows, when comparing the structural characteristics of two distinctly different pages we can not distinguish any difference. However when adding in the pHash value of the pages and using them to calculate Hamming distance we can see that indeed the pages do not match.

URL	Structural Fingerprint	Phash Value	Sites With Matching Structural Characteristics	Sites With Exact Match pHash Values	Sites With Hamming Distance Scores < 10 (Highly Related)
<a href="http://www.facebook.com">http://www.facebook.com</a>	20,1,1,0,3	35046933135104	16	0	0
<a href="http://www.us.hsbc.com/1/2/3/personal?home=personal">http://www.us.hsbc.com/1/2/3/personal?home=personal</a>	72,16,2,1,6	18446744039097171959	10	0	0
<a href="http://www.paypal.com">http://www.paypal.com</a>	71,6,3,0,3	18446677844303266811	172	0	1
<a href="http://www.irs.gov">http://www.irs.gov</a>	85,27,2,0,2	3607948993784217088	0	0	0
<a href="http://us.battle.net/wow/en">http://us.battle.net/wow/en</a>	96,6,1,0,2	9259558749612483190	61	0	0

Table 6: Markup Characteristics of “Most Spoofed Sites” Landing Pages

#### 4. Related Work

Anti-Phishing research has been done for a number of years and falls into a number of categories. A.P.E Rosiello, et. al. define these categories as Email-Based, Blacklist-Based, Visual-Clue-Based, Website-Feature-Based, and Information-Flow-Based. The method employed in their work could be best categorized as Website Feature Based and consists of a string search technique which identifies the DOM objects within a page and builds a tree of those objects. This tree is directly compared to those of known legitimate pages. When a page claims to be a specific site, all DOM trees are compared with a known good sample to look for dissimilarities [19]. This approach works as a client side browser plugin and thus is useful on an individual user basis but is not a conducive method for broader analysis.

To date the most widely deployed approaches for protecting users from Phishing attacks are Blacklist-Based. Blacklist-Based tools that enable them to be warned away from sites that are known to be malicious but have yet to be taken down. These tools rely on known blacklisting services such as Phishtank.com that rely on the submission of suspicious URLs for analysis. After a suspicious URL is submitted, a variety of techniques are employed to triage the link including suspicious DNS reputation, suspicious URL format, URL containment of other domain names in a directory field, and actual matching of known URL terms using Lexical analysis [13, 20, 24, 25, 28]. These methods reduce but do not eliminate the manual work required. Most have high false positive detection rates and all methods get the URL's that they are analyzing from other services which feed them primarily malicious or at least suspicious URL's to analyze. They have not been shown to be effective at finding malicious URLs in a large live data set of predominately legitimate URLs.



Google has developed and deployed a hybrid approach. Their system relies on both Google's own page ranking system and an email SPAM filtering system to pre-identify potentially malicious pages before analysis [26]. They also apply a machine learning classifier which considers the characteristics of the URL and the website or message content. While this method works well in Google's own environment, it is unreasonable to assume that an individual institution, or even government would be able to deploy their own version of it given Google's unique place in the Internet infrastructure.

Some other methods for detecting phishing websites that are related to our own Website-Feature-Based and Visual-Clue-Based. C. Ludl, et. al. presented evaluation results of a number of anti-phishing tools and determined that the best relied on both blacklist sites that used manual reviewers and those that did some sort of analysis using specific page characteristics. We validate our choices of page characteristics in our own method against those stated in their work [22]. We further validate our method by considering the effectiveness of methods such as the use of Page Color Histogram comparison matching and Color Vector Distance calculations which have been proven to be very effective at detecting matches between known phishing and legitimate sites [23, 29]. The main issue with both of these methods being the load that they put on a system which restricts them from scaling. Comparisons done using these systems range from .02 to 11.2 seconds per page while using up to 512 MB of RAM. Our method is able to process each page on average within 4 seconds using less than 32MB of RAM. Additionally our design easily facilitates simultaneous processing of pages.

## 5. CONCLUSIONS AND FUTURE WORK

This paper has discussed the groundwork for a method and tool that can detect phishing sites. We have promising initial results comparing known phishing sites and their legitimate counterparts. We were able to find a number of phishing sites from a live dataset and we also gained critical insight into how to effectively and efficiently gather, format, and analyze, this social data.

As this work has matured, we have continued building on our general social media analytic collection and analysis process. To date, we have improved our collection process and storage method to the point where we are consuming and storing upwards of 70 Million messages a day. In our current work includes the creation of routines that automatically find correlations between potential phishing pages and known trusted sites. Specifically, we use cluster analysis to identify clusters of pages with similar characteristics.

We are in the process of completing the missing pieces of overall analysis architecture to enable full scale, real-time analysis of the Twitter data feed. Specifically, we are completing the automated characteristic comparison routines as well as the alerting functionality as illustrated in Figure 1. In addition we plan to implement a true multi-threaded design for simultaneous processing of pages which will replace our current parallel page processing instance launching code.

We are also working on cluster analysis software that will identify the closest matches in the whole data set without requiring a legitimate site to match against.

As we continue this work we intend on implementing functions for dealing with inconsistent page load times. Currently we stop trying to process a page after 3000ms however in some select cases pages may need more time to load. We intend to implement a system whereby pages that have not loaded after 3000ms will be handed off to a child process that will continue to wait an additional period of time. This will stop a single long loading page from slowing down the rest of our analytic process.

## 6. REFERENCES

- [1] The Anti-phishing Working Group, <http://www.Antiphishing.org>
- [2] Tyler Moore, Richard Clayton, and Henry Stern. 2009. Temporal correlations between spam and phishing websites. In *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more* (LEET'09). USENIX Association, Berkeley, CA, USA, 5-5.
- [3] Maher Aburrous, M. A. Hossain, Keshav Dahal, Fadi Thabtah, "Predicting Phishing Websites Using Classification Mining Techniques with Experimental Case Studies," *Information Technology: New Generations*, Third International Conference on, pp. 176-181, 2010 Seventh International Conference on Information Technology, 2010

- [4] Report a Phishing website, <http://www.phishtank.com>
- [5] Twitter to MySQL Tool, <http://140dev.com/free-twitter-api-source-code-library/twitter-database-server/>
- [6] Bruce Prince, "Twitter Fights Phishing / Malware with Link Scanning Service", Eweek.com, <http://www.eweek.com/c/a/Security/Twitter-Fights-Phishing-Malware-With-Link-Scanning-Service-556715/3/10/2010>
- [7] Elinor Mills, "Kaspersky Tool Detects Malware in Twitter Links", ZDNet.com, <http://www.zdnet.com/news/kaspersky-tool-detects-malware-in-twitter-links/358596>, October 29, 2009
- [8] CutyCapt, <http://cutycapt.sourceforge.net/>
- [9] XVFB, <http://www.x.org/releases/X11R7.6/doc/man/man1/Xvfb.1.xhtml>
- [10] <http://www.esecurityplanet.com/views/article.php/3875866/Top-Ten-Phishing-Facts.htm>
- [11] <http://liambean.hubpages.com/hub/DNS-Report-The-Most-Banned-Spoofed-and-Sought-After-Sites>
- [12] J. Hellerstein, "The Commoditization of Massive Data Analysis," O'Reilly Radar, Nov. 2008. <http://radar.oreilly.com/2008/11/the-commoditization-of-massive.html>
- [14] C. Aggarwal, J. Wolf, P. Yu, "A New Method for Similarity Indexing of Market Basket Data," IBM T. J. Watson Research Center, SIGMOD 99, Philadelphia PA, 1999, ACM 1-58113-084-8/99/05
- [13] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. 2010. Building a dynamic reputation system for DNS. In *Proceedings of the 19th USENIX conference on Security (USENIX Security'10)*. USENIX Association, Berkeley, CA, USA, 18-18.
- [15] Zauner, Christoph "Implementation and Benchmarking of Perceptual Image Hash Functions," Master's thesis, Upper Austria University of Applied Sciences, Hagenberg Campus, 2010.
- [16] Yue Zhang, Serge Egelman, Lorrie Cranor, and Jason Hong. Phishing Phish: Evaluating Anti-Phishing tools. In NDSS '07: Proceedings of the 14th Annual Network and Distributed System Security Symposium, February 2007.
- [17] Min Wu, Robert C. Miller, and Simson L. Garfinkel. Do security toolbars actually prevent phishing attacks? In CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, pages 601–610, New York, NY, USA, 2006. ACM.
- [18] Liu Wenyin, Ning Fang, Xiaojun Quan, Bite Qiu, and Gang Liu. 2010. Discovering phishing target based on semantic link network. *Future Gener. Comput. Syst.* 26, 3 (March 2010), 381-388. DOI=10.1016/j.future.2009.07.012 <http://dx.doi.org/10.1016/j.future.2009.07.012>
- [19] A. P. E. Rosiello, E. Kirida, C. Kruegel, F. Ferr, P. D. Milano, and P. D. Milano, "A layout-similarity-based approach for detecting phishing pages," in *SecureComm '07: Proceedings of the 3rd IEEE International Conference on Security and Privacy in Communication Networks*, 2007.
- [20] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *WORM '07: Proceedings of the 5th ACM Workshop on Recurring Malcode*, 2007.
- [21] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content based approach to detecting phishing web sites," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*. New York, NY, USA: ACM, 2007, pp. 639–648.
- [22] C. Ludl, S. McAllister, E. Kirida, C. Kruegel, and C. Kruegel, "On the effectiveness of techniques to detect phishing sites," in *DIMVA '07: Proceedings of the 4th International Conference on Detection of Intrusions, Malware, and Vulnerability Assessment*, 2007, pp. 20–39.

- [23] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in SecureComm '08: Proceedings of the 4th IEEE International Conference on Security and Privacy in Communication Networks, 2008.
- [24] J. Ma, L. K. Saul, S. Savage, G. M. Voelker, and G. M. Voelker, "Identifying suspicious urls: an application of largescale online learning," in ICML '09: Proceedings of the 26th International Conference on Machine Learning, 2009, p. 86.
- [25] A. Blum, B. Wardman, T. Solorio, G. Warner, and G. Warner, "Lexical feature based phishing url detection using online learning," in AISEC '11: Proceedings of the 3rd ACM workshop on Artificial intelligence and security, 2010, pp. 54–60.
- [26] C. Whittaker, B. Ryner, M. Nazif, and M. Nazif, "Largescale automatic classification of phishing pages," in NDSS '10: Proceedings of the 17th Annual Network and Distributed System Security Symposium, 2010.
- [27] G. Xiang, J. I. Hong, C. P. Ros, L. F. Cranor, and L. F. Cranor, "Cantina+: A feature-rich machine learning framework for detecting phishing web sites." in ACM Trans. Inf. Syst. Secur., 2011, p. 21.
- [28] M. Khonji, Y. Iraqi, A. Jones, and A. Jones, "Lexical url analysis for discriminating phishing and legitimate websites," in CEAS '11: Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, 2011, pp. 109–115.
- [29] A. Y. Fu, W. Liu, and X. Deng, "Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd)," 2006, pp. 301–311.