

An Intelligent System for Phish Detection, using Dynamic Analysis and Template Matching

Chinmay Soman, Hrishikesh Pathak, Vishal Shah, Aniket Padhye, and Amey Inamdar

Abstract—Phishing, or stealing of sensitive information on the web, has dealt a major blow to Internet Security in recent times. Most of the existing anti-phishing solutions fail to handle the fuzziness involved in phish detection, thus leading to a large number of false positives. This fuzziness is attributed to the use of highly flexible and at the same time, highly ambiguous HTML language. We introduce a new perspective against phishing, that tries to systematically prove, whether a given page is phished or not, using the corresponding original page as the basis of the comparison. It analyzes the layout of the pages under consideration to determine the percentage distortion between them, indicative of any form of malicious alteration. The system design represents an intelligent system, employing dynamic assessment which accurately identifies brand new phishing attacks and will prove effective in reducing the number of false positives. This framework could potentially be used as a knowledge base, in educating the internet users against phishing.

Keywords—World Wide Web, Phishing, Internet security, data mining.

I. INTRODUCTION

PHISHING is a criminal activity using social engineering techniques [1], [2]. Phishers try to fraudulently acquire sensitive information (e-banking passwords, social security numbers, credit card numbers and so on) by constructing counterfeit websites resembling original ones and deceiving the users into believing that they are legitimate.

Cases of phishing attacks have been rising rapidly as reported by the Anti-Phishing Working Group (APWG). It is estimated that the losses due to phishing in the year 2007 were \$3.2 billion [16]. In addition, the number of unique phishing sites in September, 2007 was 28015 while the number of unique phishing reports was 38514 [4]. This is the prime indicator that static solutions like use of blacklists are ineffective against such a grave problem.

Current Solutions

Although a wide range of anti-phishing products are available, most of them are not able to make a decision dynamically, whether the site is in fact, phished, giving rise to a large number of false positives [18]. According to a study by CyLab, Carnegie Mellon University, heuristics and blacklist are the most popularly used methods against phishing. The different methodologies along with their drawbacks as observed in this study [13], [14] are summarized in Table I.

TABLE I
COMPARISON OF TECHNIQUES

Technique	Drawbacks			
	False Positives	Zero day attack	Fake Interface attack	Slow response time
Blacklist	No	Yes	No	No
Heuristics	Yes	Maybe	No	Maybe
User Polling	Yes	Yes	Yes	Maybe
Third party certification authorities	No	No	Yes	Maybe
Our solution	No	No	No	Maybe

The techniques are described in detail below:

- Blacklist check:

The suspicious URL is matched against a list of known Phishing sites. This method is susceptible to “zero day attacks” [17]. Also, techniques like URL obfuscation and routing through alternate domain name can hinder this method ineffective. As our solution does not employ a blacklist, these problems are not observed.

- Heuristics:

Uses heuristics like domain registration information (owner, age, and country), the number of links to other known-good sites, image hashing, third-party cookies and user reviews. Most of the heuristics used are subjective and produce a large number of false positives. Although some heuristics are used in our solution, they are used only in the pre-processing stages, and the actual phish detection is completely independent of them.

- User rating/polling:

Deem the URL as phished, based on user votes. However, it is ineffective against new phishing attacks and is very subjective. Our solution does not incorporate any kind of polling, thus reducing uncertainty.

- Working with third party certification authorities and reputation services:

Requires an additional interface, which itself is susceptible to phishing. Phish detection in our solution is handled completely on the server side, without involving any third party service.

Another technique is to use page rank methodology, domain analysis, URL type analysis, and word analysis, in order to detect a phishing URL [15]. However, false positives

have been observed in these methods. Also, a web site routed through content distribution network (CDN) [19] would create problems for domain based checks. Our solution is not limited to URL processing, but also analyzes the page layout.

Yet another technique includes phish detection using digital fingerprinting techniques [10]. It is somewhat similar to our technique from the point of view of identifying the corresponding legitimate web pages, but depends on assigning probabilities to the source hashes, which may not be foolproof. Our solution is designed using a comprehensive categorization scheme for source detection, giving better accuracy.

Other solutions include – Web Wallet, which is a browser sidebar, used to submit sensitive information online. However, it is susceptible to undetected form attack, and fake interface attack [9]. Our solution does not depend on any such interface, which itself is likely to get spoofed.

Our solution

By analyzing the weak points of the current solutions (Phishing Detection and Prevention [12]), it was inferred, that the best solution:

- Should detect “phishing” dynamically.
- Should be transparent to the user.
- Use foolproof criteria to deem a page as being “phished”.
- Should not be time consuming.

These are the lines on which, our framework was designed. It has a simple client-server architecture. A user agent is installed on the client machine, which acts as an arbiter between the user and the server. The user agent is nothing but a browser plug-in [8], responsible for invoking the necessary server API (Application Programming Interface) for analyzing pages. When a request is received by the server, it performs a complete scan of the suspect page, which includes:

- Finding the original page corresponding to the “suspect page”.
- Performing high level comparison
- Structurally matching the two pages, to uncover any distortions present.

before declaring the page as phished or authentic. A phished page can be declared as “phished” with sufficient accuracy as we have the original page as the basis of the comparison. Thus, a very important part of the entire phish detection process is to identify the original page from the millions of authentic pages. A distinct advantage of this approach is the reduction or a possible annihilation of the “Zero day attacks” (the time between a new phishing attack is launched and before entry for the same gets updated in the blacklist). This was a major problem earlier, as most of the solutions relied heavily on blacklisting for blocking such sites. By performing a complete and thorough scan of the “suspect web page”, it proves to be a dynamic, flexible while at the same time, rigorous solution against phishing.

In order to cut down on complexity, the detection process is completely modularized. Web page preprocessing is done, which enables to not only reduce their disk size, but also assist in the complicated comparison stage. However, the structural matching module is still costly, due to the immense calculations involved.

The rest of the paper is organized as follows. Section II illustrates the nature of the Zone map database. Section III throws light on the very important categorization module. In section IV, we analyze structural matching and the intricacies involved. Section V demonstrates the results of the experiments and we conclude with section VI.

II. ZONE MAPS

The server database consists of an extensive collection of authentic web pages which serve as a point of reference during categorization. These web pages are not stored in their original state, but are processed by the server to form “zone maps”. Motivation for creating these “zone maps” is:

- i) Minimize processing time.
- ii) Facilitate structural matching, which cannot be done, simply by comparing the HTML source codes.

A Zone map is nothing but a partitioning of the HTML page, based on the explicit visual separators that act as page layout demarcations. This partitioning is done based on the X-Y coordinate information for every individual HTML element, which is embedded into the suspect page, by the client side plug-in [6]. This partitioning enables us to concentrate on elements that are visually closer to each other, irrespective of their position in the HTML source document.

A separator is defined as any explicit visual HTML element that serves to demarcate the HTML page into different sections or zones (IMG, HR, whitespace, TD, DIV and so on). The factors used to identify a visual separator are:

- Lies within $1/8^{\text{th}}$ and $7/8^{\text{th}}$ of the width (or height) of the parent container or parent zone.
- Has a width (or height) $\geq 50\%$ of that of the parent container or parent zone.
- Has a visual property, explicitly different with respect to its parent element (for example color).
- White space elements also serve as separators.

The following thresholds have been defined for determining separators:

$\text{Height_threshold} = (\text{zone_y2} - \text{zone_y1}) / \text{WEIGHT_FACTOR.}$ $\text{Width_threshold} = (\text{zone_x2} - \text{zone_x1}) / \text{WEIGHT_FACTOR.}$

where,
 $(\text{zone_x1}, \text{zone_y1})$ $(\text{zone_x2}, \text{zone_y2})$ is the current zone.

Initially, the base zone is the whole page itself. Hence we start with a WEIGHT_FACTOR of 8 which is reduced linearly as the zones keep getting smaller. The HTML element

under consideration should satisfy following thresholds, to qualify as being a separator.

Consider a separator S.

Horizontal separator:

$$S_width / zone_width > S_height / zone_height$$

Vertical separator:

$$S_height / zone_height > S_width / zone_width$$

The idea here is to find the widest separator (horizontal) first, within a zone while vertical separators have a lower priority. In case of a contention regarding which separator to select (horizontal or vertical), the separator having proximity to the center of the parent zone, is finalized.

Map Creation

Initially, we start out with a base zone or the root zone comprising the whole HTML document and go on recursively dividing it into sub-zones, until

- A zone becomes small enough
- Visual separators cannot be found.

An algorithm used to create the different zones is given below:

1. Let base zone = BODY element of the HTML document. Push base zone onto the stack. It contains the (x1,y1) and (x2,y2) coordinate information that defines the boundary of the zone.
2. While stack not empty
 - a. Pop the zone from the top of the stack and extract the (x1,y1) (x2,y2) information. Call it the base zone.
 - b. Check if the base zone is small enough by comparing the zone area against a predetermined threshold (for example 50K square pixels). If yes, finalize the zone i.e. insert it into the zone list and go to step (2.a).
 - c. Extract all the separators falling in the window – ((x1,y1) (x2,y2)).
 - d. If there are no more separators, finalize this zone and insert it into the zone list. Go to step 2.a.
 - e. Based on the factors discussed above, select a primary separator for that zone. Save this separator information in a list (to be used later).
 - f. Based on the primary separator, divide the base zone into two sub-zones, and push them on the stack along with their coordinate information.

At end, we have two lists, a zone list and a separator summary list. After all the zones have been identified and their boundaries been clearly specified, the HTML elements in

the source DOM tree [11] are segregated according to these zones, to form zone sub-trees. A dummy root is created to form a parent node for all such zone sub-trees and the resulting tree structure is written out to an HTML file, with new tags and attributes created, where appropriate. This file is nothing but the “Zone Map”. Zone partitioning is depicted as shown in Fig. 1.

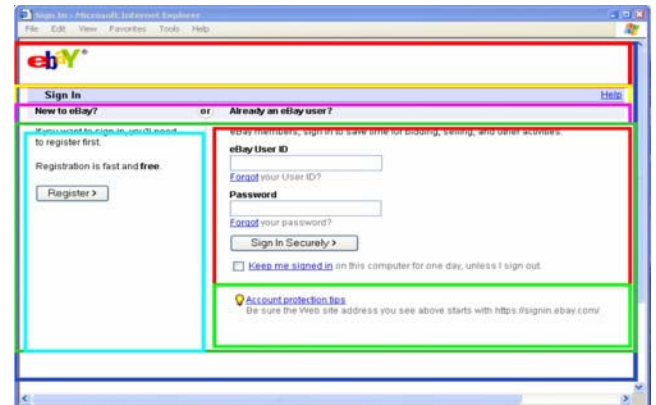


Fig. 1 Zone partitioning

The separators that were identified as a part of building the zone map serve a dual purpose. Their primary objective is to demarcate the zone layout. But they are not discarded, after the boundaries have been identified. Instead, these separators are explicitly written out to another file called “**summary file**” to serve as a final stage categorization mechanism. Intuitively, we can infer that HTML pages, having similar layout are bound to have similar “summary files”.

III. CATEGORIZATION

The categorization stage serves to identify the original web page in correspondence to the suspect page. It is characterized by condensation of the search space that is involved while matching the suspect page with the millions of web pages that exist in our database. It is divided into 2 stages - the first stage involves the determination of the base set of pages. The second stage further narrows down to a single page or a refined set of pages.

Two important data structures are used in the process of categorization – **Map table**, and **lookup table**. The map table comprises of a list of entries, one for each original web page, recording its important attributes such as keywords, URL, domain and so on. The lookup table forms a reverse index on the map table. It essentially maps a keyword, to map table entries containing that keyword. The role played by these data structures is shown in Fig. 2.

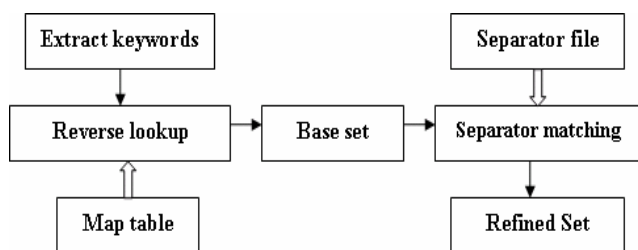


Fig. 2 Categorization process

Stage1

At the inception of stage1, keywords are extracted from the suspect page, which are nothing but the common and proper nouns characterizing that page. The title of the page forms a primary source of these keywords. But in cases, where the title is insignificant, keywords are extracted from other sources, such as the URL of the web page, the meta information, the “alt” attribute of the images occurring in the page and so on. A POS (Parts Of Speech) tagger is used for this extraction [7]. Ultimately, these keywords are matched against the lookup table, to obtain different lists of entries, one list for each keyword, pointing to the map table. These lists entries form a superset of original pages, corresponding to the suspect page. This set is refined, by taking an intersection of these lists, to form the base set. For instance, consider example set given in Table II.I and Table II.II.

TABLE II.I
MAP TABLE

Sr	URL	Keywords	
		Primary	Secondary
1	http://www.bankofamerica.com	Bank, America	e-banking
2	http://www.americastore.com	America, Store	e-money
3	http://www.icicibank.com	Bank, ICICI	Loan, Corporate Banking
4	http://www.ebay.com	Ebay	

TABLE II.II
REVERSE LOOKUP

Reverse index	Keyword
1,3	Bank
1,2	America
3	ICICI
4	Ebay
...	...

Consider a scenario:

Suspect URL:

“http://www.chileaoe.com/db/bankofamerica/cig/update.html”

Keywords extracted:

bank, America, chileaoe

The reverse indices for these keywords and their intersection gives index = 1. However, statistically speaking, primary keywords (proper nouns) carry more weight, and their reverse indices are retained in the base set. Thus the final index list obtained is {1, 2}. It results in the base set given in Table III.

TABLE III
STAGE I RESULTS

URL	Match
http://www.bankofamerica.com	87 %
http://www.americastore.com	63 %

During keyword comparison, the levenstein algorithm [5] for fuzzy string comparison is used, in order to compensate for the textual distortion present. Tunable thresholds have been defined (ranging from 0.5 to 0.9) to deem an abnormal keyword as a match or a mismatch. Table IV lists some of the abnormal keywords detected:

TABLE IV
LEVENSTEIN RESULTS

Original keyword	Suspect keyword	Match level
America	amaeric	0.933
ebay	ebaay	0.889
ebay	ebay-centers	0.5

Stage2

After all the keywords have been exhausted, emphasis is given on the visual layout of the pages, to further refine the base set. This process is called separator matching. The separator files (refer section II.I) corresponding to each of the pages in the base set, are analyzed one at a time. The separators from the suspect page are compared with the separators from this file to obtain a percentage matching. The original pages, whose matching percentage crosses a pre-determined threshold, are retained, while the others are eliminated. The final base set thus consists of those pages, on which structural matching is carried out.

IV. STRUCTURAL MATCHING

Structural matching aims at comparing two pages at a time, in order to determine the percentage matching between the visual layouts of those pages. It helps not only to identify the target original page (having the highest matching percentage and satisfying the specified threshold), but also tracks where exactly, visual distortion has been done. The latter part of the outcome, although seemingly non-productive is in fact the source of user education.

Consider the process of examining objective answer sheets, in which the answers are provided by checking a box or filling a circle for the respective questions. The sheets are examined by using a transparency having all the correct answers, in the form of checkboxes or circles marked appropriately. The transparency is then placed on top of the paper to identify

elements that coincide (the correct answers) and those that differ (the incorrect answers). Similarly, we can think of the original web page derived from the categorization module to be the transparency and the suspect web page to be the paper to be examined. Logically speaking, a zone from the original web page can be placed on top of the suspect page, and the elements falling in this window can be extracted and compared with the corresponding elements in the zone. The algorithm for zone comparison is as shown below –

1. Let cur_zone (x1,y1) (x2,y2) be a zone from the original web page.
2. Extract elements from the suspect web page, that fall within these coordinates:
 - a. Consider an unmarked element (ex1,ey1) and (ex2,ey2). If the element, completely falls inside (x1,y1) and (x2,y2) then add it to the list of selected elements and mark it.
 - b. If more than 50% of the area defined by (ex1,ey1) and (ex2,ey2) falls inside cur_zone, then add it to the list of selected elements and mark it.
3. Perform an element level comparison:
 - a. Segregate elements in the list formed in step 2 into different vectors as – images, links, lines, input elements, text segments and so on. Similar segregation is done for the elements in ‘cur_zone’ as well.
 - b. The vector comparison is done on the basis of individual properties [3]. The criteria used is as described below:
 - Images – dimension, alt, position (x1,y1) and (x2,y2).
 - Links – dimension, position (x1,y1) and (x2,y2), href, value.
 - Input elements - dimension, position (x1,y1) and (x2,y2), type.

For example, consider two images: 1 original (A) and 1 suspect (B), obtained from two corresponding vectors.

difference_area =
area defined by the coordinates:
 $(\max(Ax1, Bx1), \max(Ay1, By1)), (\min(Ax2, Bx2), \min(Ay2, By2))$
total_area = combined area of two images
(compensating for overlap portion)

In case of non-overlapping images, difference_area is set to 0. Two main structural distortion factors:

i) Dimensional distortion (dim) =
difference_area / total_area

ii) “alt” attribute distortion (alt) =
percentage difference between the two “alt” strings, based on Levenstein algorithm [5].

Consider two tunable factors: F1 and F2, having initial values 5 and 2 respectively. F1 gives more preference to dimensional distortion and F2, to alt attribute distortion.

Distortion for an image pair with respect to zone: Zone_i –

$$(dis_i) = \frac{(F1 \times dim) + (F2 \times alt)}{F1 + F2} \tag{1}$$

Distortion percentage for all images in the zone:

- i) Let W1, W2, W3, ... Wn, be the weights or the image area of the individual images in the zones under consideration.
- ii) Averaging factor for each image:

$$Fi = \frac{Wi}{\text{Total image area}} \tag{2}$$

Total distortion percentage for images:

$$Zp = \frac{Fi \times dis_i}{\sum (Fi)} \tag{3}$$

Similar distortion percentages would be calculated for links, input elements etc.

- c. Also consider visual properties of the container element. The comparison essentially follows a bottom up approach, starting from the elements towards their nearest common parent.
4. After all the zones have been processed, there exists a list of percentages one for each of the zones.
5. Weights are assigned to the zones in a decreasing order of priority of logos, images, input elements, links and finally other HTML elements.
6. Using these individual weights and distortion percentages, a weighted average is calculated and returned according to the following formula –

$$\frac{(Z1p \times Z1w) + (Z2p \times Z2w) + \dots + (ZNp \times ZNw)}{Z1w + Z2w + Z3w + \dots + ZNw} \tag{4}$$

Where,
ZNp : Zone percentage(mismatch%),
Z1w : Zone weight.

From the list of percentages derived for each of the pages in the base set, we select the page satisfying a pre-defined threshold as the corresponding original page. Thus at the end of this module, we can not only identify whether the suspect page is phished or authentic but also quantify it, based on the distortion percentage obtained. In other words, given a distortion percentage – ‘dp’, we can state that:

- The suspect page is phished, if $dp > 0$.
- The suspect page differs from the original page by a measure of 'dp %'.

The original page detected in this process may be displayed to the user, as a confirmation mechanism. This can provide a foundation for a more enhanced form of user education, against phishing.

The case where none of the pages in the base set satisfy the criteria (threshold) has two possibilities:

- Either the suspect page is highly distorted
- Or, the corresponding original page is not present in the zone map database.

In such a case, the server sends an output back to the user agent labeled "non-deterministic".

V. EXPERIMENTS

We conducted the tests with approximately 70-90 URL's, the outcome of which was optimistic. Included in this test set were "phished" pages, then available on the internet, spoofed web sites developed by our team, and some authentic pages as well. Table V lists some of the prominent spoofed web sites, which the server was able to detect, among many others.

The success rate was calculated as 81 %. The authentic pages were used to test if the system reported any false positives. In 95 % of these tests, the pages were correctly identified as being authentic. The remaining 5% pages were deemed non-deterministic, due to the fact that those original pages were not incorporated in the database, at that point of time.

TABLE V
EXPERIMENT RESULTS

Suspect URL	Distortion
http://218.108.235.43/cgi-bin/.webscr/	12.65 %
http://62.60.137.21/articles/.paypal.com/secured/login/cmd=_login	32.92 %
http://62.60.137.21/articles/.paypal.com/secured/login/cmd=_login	11 %
http://www.chileaoe.com/db/cig/update.html	21.5 %
http://195.137.222.129/Amazon/www.amazon.com/	20 %

But, none of the authentic pages were labeled as being "phished", which is a significant achievement. However, one of the tests discovered a weakness in the current design of structural matching. In case of pages, having lesser number of keywords, and having a visual layout matching the original page, but constructed with images, it gives a wrong indication of the matching percentage. Such pages could escape the detection process. Although this is a rare scenario, measures like image correlation and structural matching at a deeper granularity could minimize such risks.

On an average, 50KB of storage is required for a zone map. If we consider 10 pages per web site, for a collection of 1 million sites, the total size of the zone map database comes to 500 gigabytes of storage, which is quite feasible. However, the turnaround time for the server response ranges from 15 seconds to over 2 minutes, depending upon the size of the base set, which is considerable and needs to be improved.

VI. CONCLUSION AND FUTURE WORK

This paper highlighted a novel, completely automated approach against phishing. Undoubtedly, having the original page to prove whether the suspect page is phished or authentic, is the ideal mechanism for providing security. Its role in reducing zero day attacks is remarkable. Moreover, having a foolproof solution against phishing is more important than an ineffectual, faster solution. A success rate of 81% brings this design closer to the ideal solution than others, existing in the market. At the same time, there are some proposed improvements:

- User education – Giving details of visual distortion, using color codes.
- User polling – In case of nondeterministic output, enabling user to vote, whether a page is phished or not.
- Improving structural matching – also incorporating an image processing mechanism.
- Improving the response time of the system.

ACKNOWLEDGMENT

Chinmay Soman thanks Mr. Abhay Shete for his precious guidance in formulating the categorization scheme.

REFERENCES

- David Watson, Thorsten Holz and Sven Mueller, -"Know your enemy: Phishing, behind the scenes of Phishing attacks", The HoneyNet Project & Research Alliance.
- Rachna Dhamija, J. D. Tygar, Marti Hearst - "Why Phishing works"
- HTML element - wikipedia http://en.wikipedia.org/wiki/HTML_element
- Anti Phishing Working Group – Phishing Activity Trends Report – September, October 2006, and September 2007.
- Levenstein, A., Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (1966):707–710
- Jie Zou, Daniel Le and George R. Thoma "Combining DOM tree and geometric layout analysis for online medical journal article segmentation", National Library of Medicine.
- Steven Abney – "Parts Of Speech Tagging (POS) and Partial Parsing", 1996
- Dino Esposito, "Browser helper Objects: The Browser the Way You Want It", Microsoft-Corporation <http://msdn2.microsoft.com/en-us/library/bb250436.aspx>
- Min Wu, Robert C. Miller and Greg Little – "Web Wallet: Preventing Phishing attacks by revealing user intentions", MIT Computer Science and Artificial Intelligence Lab.
- Jonathan Zdziarski, Weilai Yang and Paul Judge – "Approaches to Phishing identification using match and probabilistic digital fingerprinting techniques.", CipherTrust, Inc.
- Suhit Gupta, Gail Kaiser, David Neistadt and Peter Grimm – "DOM-based Content Extraction of HTML Documents".
- Tod Beardsley – "Phishing detection and prevention: practical counter-fraud solutions".

- [13] Min Wu, Robert C. Miller, Simson L. Garfinkel – “Do security toolbars actually prevent Phishing attacks?”, MIT Computer Science and Artificial Intelligence Lab.
- [14] Lorrie Cranor, Serge Egelman, Jason Hong, and Yue Zhang – “Phinding Phish: An evaluation of anti-Phishing toolbars”, CyLab, Carnegie Mellon University.
- [15] Sujata Garera, Niels Provos, Monica Chew and Aviel D. Rubin – “A framework for detection and measurement of Phishing attacks”.
- [16] <http://www.gartner.com/it/page.jsp?id=565125>
- [17] The Zero-Day Attack, PC magazine,
- [18] <http://www.pcmag.com/article2/0,1759,1880013,00.asp>
- [19] False positives : Type I and type II errors, wikipedia - http://en.wikipedia.org/wiki/Type_I_and_type_II_errors
- [20] Content Distribution Network, Wikipedia - http://en.wikipedia.org/wiki/Coral_Content_Distribution_Network