Research Report: Phishing Attack Detection Techniques

Analyzing Previous Algorithms

**Using Public Key Certificates**

One method proposed for the detection of phishing websites is the use of Public Key Certificates. In an article by Dong, Kapadia, Blythe and Camp [1], they highlight a method of using the structure of X.509 certificates to confirm phishing websites. Their method uses machine-learning to build a phishing detection system, which includes the following four components [1]:

1. Certificate Downloader: needed to obtain the certificates from a server, either though HTTPS or using a TCP call to the server on port 443.
2. Feature Extractor: needed to parse the certificate and extract useful fields to be used in determining patterns of phishing.
3. Classification Executor: applies six machine-learning models to the output of the extractor module. The six models are Random Forrest, K-Nearest Neighbors, C4.5, Decision Table, Naive Bayes Tree, and Simple Logistic.
4. Decision Maker: Makes the final decision on whether a website is phishing or no-phishing, using two algorithms, Random Forest and Average Probability.

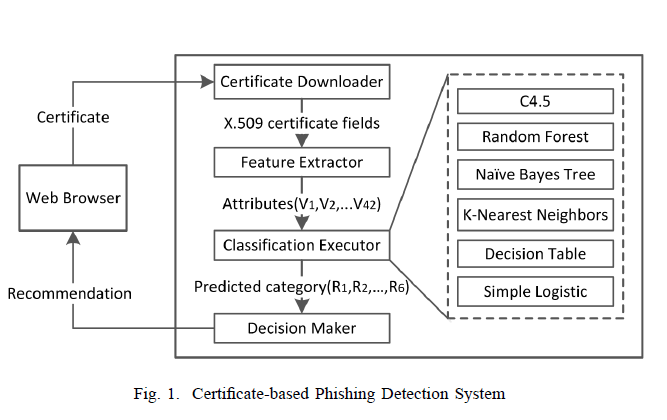


Fig. 1 is a representation of the proposed method described above [1].

One of the most important steps in developing this system, was the selection of appropriate Features that can be used in classifying whether or not the certificate belongs to a phishing URL. A total of 42 features were selected based on the structure and content of the X.509 certificate. These features included:

1. The Expiration Date of the certificate
2. The Issuer, Subject and Domain Name of the certificate
3. Whether or not the Certificate Authority is Trusted
4. The length of some Sub-fields including Email, Organization, City, State, Country of the Issuer and Subject.
5. Cryptographic algorithms and hash functions, as these are important indicators of insecure certificates.
6. Certificate Extensions, including some of the most common extensions such as authorityKeyIdentifier, subjectKeyIdentifier, certificatePolicies, isExtendedValidation, etc.

Pros

1. This model is an improved alternative to traditional detection approaches, like blacklisting, as it can be run by end users in real time.
2. This model also does not require frequent updates from a central server, thereby eliminating any vulnerabilities users may face if they were using blacklists.
3. Experimental results indicate that this method of anti-phishing can identify phishing websites that use the HTTP protocol as well as HTTPS and classify sites as either phishing or non-phishing.

Cons

1. Only one source, PhishTank, was used to illicit the verified phishing sites used in the dataset.
2. This model is susceptible to sophisticated learning attacks such as evasion and poisoning.
3. Need to train data before classification can occur.

**Using Heuristics**

In an article by Rao and Ali [2], they developed a desktop application called PhishShield, which is able to detect phishing URLs based on heuristics and whitelists. The proposed system is divided into 5 levels of detection, as follows [2]:

1. Use of Whitelists: the domain of the URL is compared with trusted website list known as Whitelist. If the URL is not on the whitelist, the webpage is parsed and stored as a Document Object Model (DOM) element. The html page is parsed for the presence of a login page. If the page is not found, the application stops, otherwise, it continues to level 2.
2. Zero Links in Body of HTML: if there are no hyperlinks in the body of the login page, the URL is deemed to be phishing.
3. Footer Links Pointing to NULL: if the anchor tag in the footer section is pointing to NULL, the URL is deemed to be phishing. This is suspicious because legitimate websites were found to Not contain NULL links in their footer.
4. Use of Copyright and Title Content: the <div> tag containing copyright information and the <title> tag content is extracted from the DOM object, then analyzed and compared with the white lists for a match. If a match is found, the URL is deemed to be phishing.
5. Website Identity: if the domain entered in PhishShield does not match the domain having the maximum frequency, then the URL is deemed to be phishing. This is because legitimate websites have a high frequency of hyperlinks pointing to its own domain.

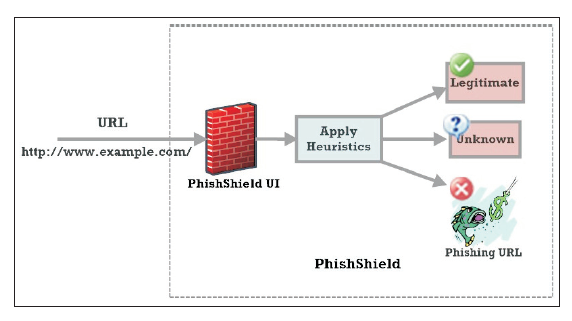


Fig. 2 Architecture of PhishShield [2].

Pros

1. PhishShield is able to detect phishing sites that employ Image Based Phishing Attacks, and it is able to do this with acceptable execution time.
2. Utilizes a heuristic approach which results in fewer false positives or false negatives.
3. Simple application design which is easy for users to make use of, as they simply have to copy the suspected URL into PhishShield and click “Check the URL”.

Cons

1. Focus was only put on examining the logon page of webpages to make a decision on whether or not the URL is legitimate or in fact, phishing.
2. PhishShield is unable to detect phishing when all the filters are bypassed. This occurs when some phishing sites fail to use targeted legitimate content to display webpages.
3. PhishShield fails when its parser component, JSoup, fails.

**Using Human Behavior**

Automation of Human behavior is a phishing website detection method that proposed by Rao and Pais [3]. This method will detect phishing site by submitting fake credential, as author mentioned in this paper: “Attacker’s main target is to collect the sensitive information rather than validation of sensitive information [3].” Thus, they proposed a set of checking mechanism to find the phishing website.

**Features of this algorithm:**

1. Identifying the phishing sites based on the login status of a website when fed with fake credentials such as username and password.
2. Detecting phishing pages which are hosted on compromised domains.
3. Detecting phishing sites which use Iframe for embedding imitated content.

**Pro:**

1. Can be automated;
2. Do not depends on search engines, WHOIS or page rank;
3. No need for training data, black list or history;
4. Able to detect the phishing websites based on compromised domains;
5. Able to detect the phishing websites which use captcha verification in login page;
6. Able to detect the phishing websites contains embedding HTML text;
7. Domain conflict checking.

**Con:**

1. Only useable on the phishing website needs login credential; In
2. Failed to handle the anchor text replaced with icon or images;
3. Not useable on Single Sign-On (SSO) websites (Failed to handle the login procedure via JavaScript);
4. Assume that attacker does not verify the input login credential;

**Algorithm:**

Architecture of the application (FeedPhish)

|  |
| --- |
| Input: URL   1. **Login Check – module 1** 2. IF (Login page Found == NO) then   **Legitimate site;**  **End If**   1. Feed Fake Credentials – module 2 2. **IF** (! Login Successful) then   **Heuristic Check – module 2**  **IF** (Features Cost > Threshold == NO) then  **Legitimate site;**  **End If**  **End If**   1. Target Domain Check 2. **IF** (Target Domain Found == NO)   **Phishing Site Found;**  **End IF**   1. Target Domain Message 2. Phishing site Found |

Login Check (module 1)

|  |
| --- |
| Input: URL   1. Navigate URL with Selenium 2. **IF** (Password Field Found) **then**   **Feed Fake Credentials**  **EndIF**   1. IF (Login Links/Buttons Found) **then**   Click Link/Button  **Feed Fake Credentials**   1. **IF** (iframes Found total iframes # = n)   **while** (i < n)  Navigate iframe i  **IF (**Password Field Found**) then**  **Feed Fake Credentials**  **End if;**  i++;  **End while;**  **End IF**   1. Login page Absent 2. **Legitimate** |

Heuristics Check (module 2)

|  |
| --- |
| 1. Feed Fake Credentials 2. **IF** (Login Successful == NO) **then**   Heuristics Check  Extract Features using JSoup  **IF** (Zero Links Found) **then**  **Phishing Site Found;**  **End IF;**  **If** (Common Page Ratio > 0.75 == NO) **then**  **Legitimate;**  **End IF;**   1. Target Domain Check 2. **Phishing Site Found;** |

**Using Data Mining**

In this paper, the authors proposed a datamining approach to extracting URL links from emails. The proposed algorithm has two phases to detect the phishing website: phase one will extract all URLs from an email, the application (DC scanner) will verify the domain authority and other information that might be hidden under the HTML code; second phase will check that the web page can send the contents of a form to a web server, the application will verify that the URL links in the web page is the same as in the domain. If the URLs pass all the checks in phases 1 and 2, the email/link will be deemed as non-phishing website.

**Features of this algorithm:**

1. Domain Verification: this step will check website’s full address and Domain Associated Authority name in the webpage in the head section.
2. Webpage Scanning: Checking for input control and webpage tag has any suspicious behavior.
3. Script Scanning: Checking associated script codes such as Javascript could redirect user’s input to other domain or any links that contain malicious websites.

**Pro:**

1. Use datamining strategy to identify the malicious URLs in email and html code.
2. Reuse the same strategy for checking webpage and script.
3. Expandability for future refinement.

**Con:**

1. Require web pages must base on the certain design rules in order to efficient and accurately identify the malicious webpages;
2. Unable to detect malware-based phishing attack; (“For bringing correct result we depend on some essentials of web page design. We define these essentials and suggest the whole cyber society to carve these instructions as standard to design web pages.” [4])

**Algorithm:**

Phase I:

|  |
| --- |
| **Input**: Link Addresses in an Email.  **Output**: Verified Links  I. email\_Link [] = Anchor tags in Email  II. Found\_ URL [] = value (href attribute of email\_link [])  Repeat below steps for every Found\_ URL []:  1. domain \_ name = domain (Found\_ URL)  2. page \_ content = html content (Found\_ URL)  3. domain\_tag [] = A body tag with (body.id/name = domain\_name) //As per definition 3.  4. **If** domain\_tag[] = ""  **then** // The web page doesn't follow the design instruction  -Alert "This link may have phishing page"; **-Stop**  **Endif**  5. **If** Array\_ Length (domain\_tag []) > 1  **then**  -Alert "This link may have phishing page"; **-Stop**  **Endif**  6. **If** domain\_name != domain(domain\_tag[0])  **then** // The link URL is different than what is claimed in the webpage  -Alert "This link may have phishing page"; **-Stop**  **Endif**  7. author\_tag = A head (hl, h2, h3) tag with (id/name = domain\_name)  8. domain\_info \_ url = innerHTML of author\_tag  9. domain\_info\_web = Filtered whois(found\_URL) // Use who is command on the web //URL and find a unique information about domain authority.  10. **If** domain\_info\_web !=domain\_info\_url then // what  Organization is being shown here is not the exactly  -Alert ("This link may have phishing page")  **Endif**  11. **web\_forms** = HTML form in page\_content;  12. **If** web\_form [0] ="" **then**  -Stop // No forms then no chances of attacks  **Endif**  13. **while** (web\_forms [])  **directed\_url** = value of action attribute in web \_ forms []  **If** url\_format (directed\_url)  **then** // url\_format is a function to check if the action attribute contains a web URL.  **If** domain(directed\_url) != domain\_name  **then** // The action attribute directs to another website  --Alert ("This link may have phishing page")  **End if**  **End if**  **End while** |

Phase II:

|  |
| --- |
| **Inputs**: Source codes of open web page  **Output**: Verified web page  1. page = HTML contents of open web page  2. scripLsections [] = Parse page for <script> tags  3. page \_ domain = domain(page) // Find the domain of open web page  4. **while** (script\_sections [])  -script\_url [] = scan script\_sections[] for urIs  -**while** (script\_ url [])  -url\_domain[] = domain(script\_url[]) // Get domains of urIs found in script sections  -**If** url\_domain[] != page\_domain **then** //The script\_uri redirects to malicious web servers  -Alert ("This may be a Phishing page")  -Stop  -**Endif**  **-Endwhile**  **-Endwhile**  5. domain\_tag = A body tag with (id/name=page\_domain)  // As per the definition 3  6. author\_tag = A head(hl,h2,h3) tag with  (id/name = page \_domain)  7. **If** (domain \_ tag == hidden) **then**  //website address is available in page but it is hidden  -Alert ("This may be a Phishing page")  **Endif**  8. **If** (author\_tag == hidden) **then**  // website authority info is not shown as it is to forge the DC Scanner  -Alert ("This may be a Phishing page")  **Endif** |

**References**

[1] Z. Dong, A. Kapadia, J. Blythe, and L. J. Camp, “Beyond The Lock Icon: Real-Time Detection Of Phishing Websites Using Public Key Certificates,” in Electronic Crime Research (eCrime), 2015 APWG Symposium on. IEEE, 2015.

[2] R. Rao and S. Ali, "PhishShield: A Desktop Application to Detect Phishing Webpages through Heuristic Approach", Procedia Computer Science, vol. 54, pp. 147-156, 2015.

[3] Rao, R. S., & Pais, A. R. (2017). Detecting Phishing Websites using Automation of Human Behavior. *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security - CPSS 17*. doi:10.1145/3055186.3055188

[4] Kumar, B., Kumar, P., Mundra, A., & Kabra, S. (2015). DC scanner: Detecting phishing attack. *2015 Third International Conference on Image Information Processing (ICIIP)*. doi:10.1109/iciip.2015.7414779

**Group Member Contributions**

Shinelle Hutchinson

1. Read and Analyzed two articles, [1] and [2] above, writing on the algorithms proposed in both.
2. Helped in creating the PowerPoint Presentation.

Zhaohe Zhang

1. Read and Analyzed two articles, [3] and [4] above, writing on the algorithms proposed in both.
2. Helped in creating the PowerPoint Presentation.