PhishStorm: An Automated Real-time Phishing Detection System for Emails

Gaurav Deshmukh
Department of ComputerEngineering,Savitribai Phule Pune University
Pune, India

Rushikesh Deshmukh
Department of Computer Engineering,Savitribai Phule Pune University
Pune, India

Sanket Fajage
Department of Computer Engineering,Savitribai Phule Pune University
Pune, India

Pratiksha Kate
Department of Computer Engineering,Savitribai Phule Pune University
Pune, India

A.B.Bagwan, PhD
Department of Computer Engineering,Savitribai Phule Pune University
Pune, India

ABSTRACT

In consideration with the security of user on the Internet, Phishing remains an important threat. Currently working systems are based on reactive URL blacklisting technique which is inefficient due to short lifetime of Phishing websites and unavailability of newly launched Phishing sites. In the proposed system we introduce a real-time automated phishing detection system for e-mails which can analyze an URL present in the context of an e-mail to identify potential Phishing sites. Along with the URL, contents of emails are also checked with intra-URL relatedness. Phishing URL's have relationships between their first part (lower domain) and remaining part (upper domain) like path, file folder, etc. URL uniqueness is figured out by evaluating it using identical properties extracted from keywords that compose a URL based on the query data from Google and Yahoo search engines

Keywords

Intra-URL Relatedness, Mails, Phishing detection, Lower Domain, Upper Domain**.**

1. INTRODUCTION

Phishing has its presence in cyber-crime activities. Accurate evaluation of financial loss caused by phishing is unpredictable. In 2013, India witnessed 9 per cent increase in phishing attacks with a total of 308,371 websites were hacked in India from 2011–2013 as per India Risk Survey 2014 by Federation of Indian Chambers of Commerce and Industry (FICCI). Goals behind phishing any website are data, money and credential stealing.

Various techniques are adopted to perform phishing attacks such as DNS cache poisoning, e-mail spoofing Web server takeover, etc. Despite this diversity, one common feature is the use of obfuscated URLs to misdirect users to fake Websites or drive-by downloads. Phishing email contains messages to lure victims into performing certain actions, such as clicking on a URL where a phishing website is hosted, or executing a malware code. Phishing has become the most popular practice among the criminals of the Web. Phishing attacks are becoming more frequent and sophisticated. Techniques like taking down phishing Web sites have been proved difficult and inefficient mainly due to short Web site lifetime (around 12 hours). Hence real-time malicious URL detection is a better technique for defeating phishing.

In this paper, we propose an automated real-time URL phishing detection system to protect users against phishing content embedded in their mail. The proposed system is restricted to accessing the mails from gmail only. Google provides their own gmail's API to retrieve and send mails. Phishers blend many phishing keywords (famous brand, attractive words) into the remaining parts of the URL in order to delude their victims. Seeing words like PayPal, eBay or visa at any level of a URL will make them feel confident that the link actually leads to the official Web site of these brands.

PhishStorm is an automated real-time URL phishing detection system to protect users against phishing content. It is been observed that there are few relationships between the registered domain and the rest of the URL. However, the words that compose the rest of the URL (low level domain, path, query) often have many interrelationships. So identifying whether the URL is legitimate or phishing URL, our approach is to evaluate the relatedness between the registered domain and remaining part of the URL. Here we are using search engine query data from Google and Yahoo Clues to compute this relatedness.

Intra-URL relatedness is defined as an efficient feature computation methods using distributed streaming analytics techniques and space-efficient data structure. These reduce the delay of detecting phishing URLs that has been observed in paper [1] and allows applications such as phishing email or HTTP traffic filtering. We extract 6 features from a single URL which are input to machine learning algorithms to identify phishing URLs. These 6 features are derived from the intra-URL relatedness. In the proposed system, email stream is captured by honey pot subsystem from the Internet and is parsed into a MIME email first; various features are extracted from the email and outputted into feature vectors which are inputted to some classification technique [2].

To summarize the major contributions of this paper:

- To build a phishing detection system that retrieves mails from the users gmail account and analyze it to extract the URL's present in the mails contents.

- Concept of intra-URL relatedness depicting the relationbetween a registered domain and the words that compose the rest of a URL is evaluated. We then use search engine query data to establish relatedness between words 6 features based on intra-URL relatedness and build a machine learning based approach relying on these for distinguishing between phishing and non-phishing URLs.

2. SYSTEM ARCHITECTURE

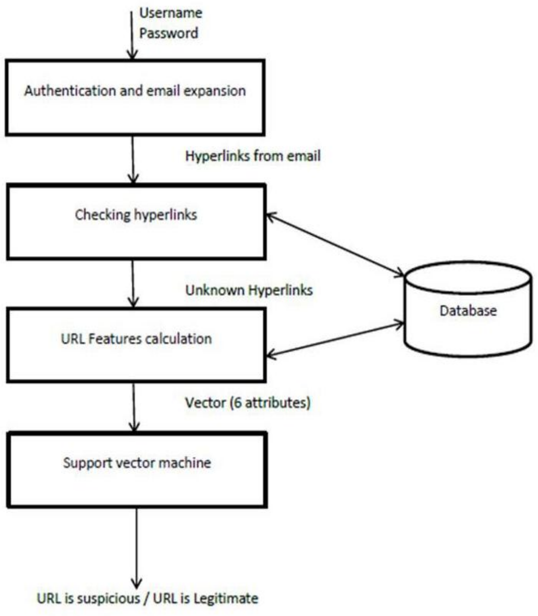


Fig 1: System Architecture

Description
The proposed system assumes that the user has an internet connection also has a gmail account. Firstly user enters the credentials to the system which includes username and password. These credentials would be then authenticated by the Email server after which the mails would be retrieved.The retrieved mails are then analyzed to extract the URL's from it. URL features are then extracted using Google trends and Yahoo Clues, been discussed in Section III. The proposed system aims to extract 6 features depending on the intra-URL relatedness. Since the hypothesis is that the legitimate URL's have high intra-URL relatedness. Values of the features are then given as an input to the classifier mainly SVM (Support Vector Machine) in order to decide whether the URL is legitimate or phishing. The classifier uses PhishTank dataset in order to classify the URL.

If any mail includes a non-legitimate URL then it would notified to the end user (client) about it.

3. PROPOSED SYSTEM

In order to use a domain mydomain.tld and derive several URLs from it: url1.mydomain.tld, url2.mydomain.tld/file, one needs first to register the domain mydomain.tld at a domain registrar, ensures that it cannot be registered by anybody else. Assuming a phisher wants to trap PayPal users, he must use a domain.tld other than paypal.com, as this domain is already registered by PayPal Inc. The phisher must register a domain name mydomain.tld and try to deceive people by blending labels such as PayPal into the rest of the URL: login.mydomain.tld/paypal. A registered domain consists of two parts: a main level domain and a public suffix. A public suffix (or ps) is a domain name suffix under which an Internet user can register a name. It can be just a Top Level Domain like .com, .org or a combination of level domains like .co, .uk or .blogspot.com. A main level domain (or mld) is the level domain preceding a public suffix. A registered domain is then: mld.ps. For instance in www.paypal.com/login, com is the ps and paypal is the mld. The different obfuscation techniques consist of blending either the original domain name or phishing keywords into the remaining part of the URL. These keywords are usually the targeted brand, related services of the brand and other attractive words such as secure, login, protect, etc. The URL's are then split according to non-alpha numeric characters. If extracted parts composed of several words such as paypalitlogin in http://sezopoztos.com/paypalitlogin/us/..We use a dictionary based words splitter [4]. For instance, the three words paypal, it and login are extracted from paypalitlogin through this process.

For example:

http://sezopoztos.com/paypalitlogin/us/webscr.
html?cmd=_login-run

RD_{url} = {sezopoztos, sezopoztos.com}

REM_{url}= {paypal, it, login, us, web, src, html, cmd, login, run}

AS per [3], mld.ps is not split like the other part to keep

themld unmodified, which can be composed of several words.

Once two sets are obtained that is Registered andRemaining set then we define four sets of words built from a URL url:

REL_{rd} (url), REL_{rem}(url), AS_{rd} (url) and AS_{rem} (url)

Where REL is relatedness and AS defines Associated Set

Mining search engine query data for word relatedness measurement is relevant in a phishing context. To achieve this goal, we use search engine query data from two top ranked search engines: Google and Yahoo. Both offer services that given a term provide some insights on requesting trends concerning it. These services are respectively Google Trends and Yahoo Clues. In the context of this paper we define a term t as a set of words w. {paypal} and {paypal, login, secure} are two examples of terms.

REL_{rd} (url) = $\{w \in t \mid t \in Term_{w'}, w' \in RD_{url}\}$

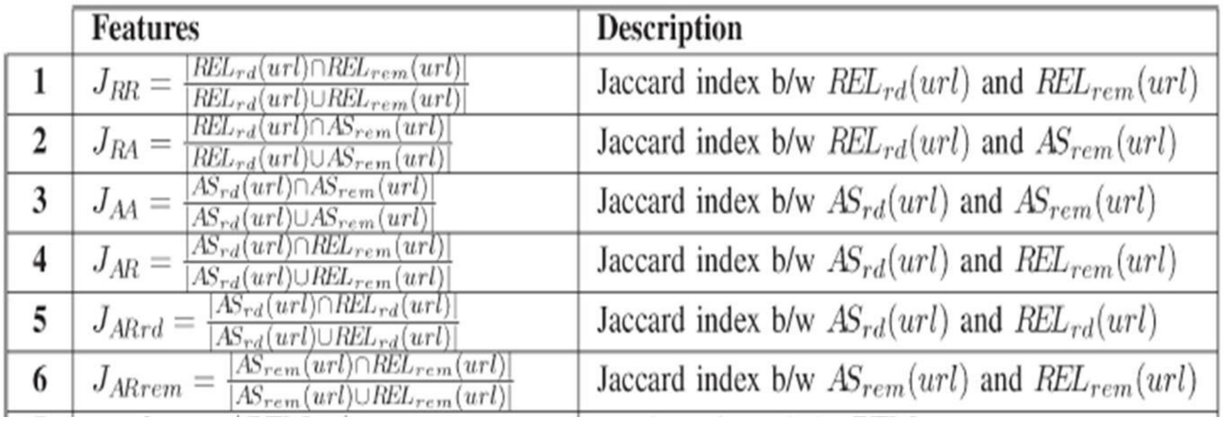
REL_{rem}(url) = $\{w \in t \mid t \in Term_{w'}, w' \in REM_{url}\}$

AS_{rd} (url) = $\{w \in t \mid \exists w' \in RD_{url}, w' \in t, w' \neq w\}$

AS_{rem}(url) = $\{w \in t \mid \exists w' \in REM_{url}, w' \in t, w' \neq w\}$

Based on the sets, 6 features are introduced and these features are described in **Table I**.

Table 1: Features



	Features	Description
1	$J_{RR} = \frac{\lvert REL_{rd}(url) \cap REL_{rem}(url)\rvert}{\lvert REL_{rd}(url) \cup REL_{rem}(url)\rvert}$	Jaccard index b/w $REL_{rd}(url)$ and $REL_{rem}(url)$
2	$J_{RA} = \frac{\lvert REL_{rd}(url) \cap AS_{rem}(url)\rvert}{\lvert REL_{rd}(url) \cup AS_{rem}(url)\rvert}$	Jaccard index b/w $REL_{rd}(url)$ and $AS_{rem}(url)$
3	$J_{AA} = \frac{\lvert AS_{rd}(url) \cap AS_{rem}(url)\rvert}{\lvert AS_{rd}(url) \cup AS_{rem}(url)\rvert}$	Jaccard index b/w $AS_{rd}(url)$ and $AS_{rem}(url)$
4	$J_{AR} = \frac{\lvert AS_{rd}(url) \cap REL_{rem}(url)\rvert}{\lvert AS_{rd}(url) \cup REL_{rem}(url)\rvert}$	Jaccard index b/w $AS_{rd}(url)$ and $REL_{rem}(url)$
5	$J_{ARrd} = \frac{\lvert AS_{rd}(url) \cap REL_{rd}(url)\rvert}{\lvert AS_{rd}(url) \cup REL_{rd}(url)\rvert}$	Jaccard index b/w $AS_{rd}(url)$ and $REL_{rd}(url)$
6	$J_{ARrem} = \frac{\lvert AS_{rem}(url) \cap REL_{rem}(url)\rvert}{\lvert AS_{rem}(url) \cup REL_{rem}(url)\rvert}$	Jaccard index b/w $AS_{rem}(url)$ and $REL_{rem}(url)$

The features define intra-URL relatedness by calculating the Jaccard index pairwise between the four sets (REL_{rd} (url), REL_{rem}(url), AS_{rd}(url) and AS_{rem}(url)). (REL $_{rd}$ (url), REL_{rem}(url), AS_{rd} (url) and AS_{rem} (url)). The Jaccard index is a metric used to calculate similarity and diversity between two sets A and B. The closer J (A, B) is to 1 the more similar are A and B. These six features quantify the relatedness between the two parts of the URL (mld.ps and the rest) through J_{RR}, J_{RA}, J_{AA} and J_{AR}, as these compute Jaccard indexes between sets extracted from different parts (RD $_{url}$ and REM $_{url}$). These also measure the relatedness inside each part through $J_{AR\ rd}$ and J_{ARrem},as these features are calculated from sets extracted from the same part of a URL.

In order to classify the url as either illegitimate or legitimate, the system is first trained with tentatively 50 URL's each (legitimate and illegitimate). Once the system is trained by calculating the features of all the URL's, next Url scanned in the mails are evaluated for 6 features and based on the features the Url is classified as either legitimate or illegitimate.

SVM (Support Vector Machine) is a binary classifier. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

However, in this context, SVM is been implemented using Wekatool[7].

4. MATHEMATICAL MODEL

Let S be the system.

S={Input, Output, Constraints, Functions, Success, Failure}

Where

Input={Input 1,Input 2,Input 3}

Input 1: User credentials for G-mail

Input 2: Mails

Input 3: Dataset

Output={Result 1, Result 2}

Result 1: E-mails with legitimate URL's are shown in general color Result 2: E-mails with Phishing URL's are shown with Red mark

Data structures: Bloom Filter

Divide and conquer strategies: Dividing URL and extracting features from the URL

Parallel processing: Searching the URL in Phishing and Non-Phishing URL database Concurrent: Features extraction

Constraints={Constraint 1, Constraint 2} Constraint 1: User should have an email account Constraint 2: System should have a dataset

Functions={Function 1, Function 2, Function 3, Function 4} Function 1 = Credential validation

Function 2 = E-mails extraction

Function 3 = Features extraction

Function 4 = SVM classifier to identify E-mail is Phishing or not

Functional relations: E-mails will get extracted only if user's credentials are valid

Success Conditions: Detection of phishing URL's

Failure Conditions: A phishing website is detected as a legitimate one.

5. TECHNICAL DETAILS

URL is to be classified with SVM (Support Vector Machine) classifier using WEKA [7]. Accuracy of each classifier for URLs:

- Phishing classified as phishing: true positives (T P) and TP $_{rate}$ = $\frac{TP}{TP + FN}$

- Legitimate classified as phishing: false positives (FP) and

  FP $_{rate}$ = $\frac{FP}{TN + FP}$

- Legitimate classified as legitimate: true negatives (TN) and TN $_{rate}$ = $\frac{TN}{TN + FP}$

- Phishing classified as legitimate: false negatives (F N) and $FN_{rate} = \frac{TP}{TP + FN}$

and the accuracy: $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$

The bottleneck for feature computation is the sequential network communication overhead with the Google and Yahoo servers. However, this bottleneck can be easily removed by leveraging distribute parallel computations over STORM topologies. Because of the transactional support, we use the Trident topologies; as proposed in paper [3]. Within such a topology, nodes perform a processing logic. Nodes are connected with links that indicate how the data is processed. Data is sent within a "Stream"and STORM can distribute the computation along a sequence of nodes. There are two types of nodes: *Spouts*and *Bolts*. *Spouts*represent the source of data. For our architecture, the *spout* (URL-Spout) is a URL extraction component that extracts individual URLs. Each individual URL is sent to four different *bolts*. This is done using one of the streams grouping method. Several such methods exist and their working depends on how a *spout* decides to split the output to the connected *bolts*. The *All-grouping* method is the replication to all attached *bolts*. Sem-Bolts in our architecture have a simple function. They connect to the Google and Yahoo servers and retrieve the list of semantically equivalent words. The intersection among this needs to be performed by the Intersection-Bolt.

Using such a simple STORM topology, we can reduce the communication time by a factor of four. Further improvements can be obtained by increasing the degree of parallelism through an increased number of *bolts.* In this case, the *spout* tokenizes the URLs into different words. Each individual word is sent to a URL-Bolt. In this case, the *Field-grouping* method is used. This ensures that a word occurring in several URLs will be always sent to the same URL-Bolt and therefore a caching strategy can avoid repeating the same requests. Each URL-Bolt will furthermore replicate the input to four Sem-Bolts, where each Sem-Bolt is responsible to communicate with a Google, respectively Yahoo server. In this case, we can reduce the communication overhead to the single round trip time between our platform and the Google/Yahoo servers.

6. ACKNOWLEDGMENTS

It gives us immense pleasure in presenting the project 'PHISHSTORM: An Automated Real-Time Phishing Detection System for emails'. I would like to take this opportunity to thank my internal guide Dr. A. B. Bagwanfor giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful. I am also grateful toDr. A. B. Bagwan, Head of Computer Engineering Department,JSPM's Rajarshi Shahu College of Engineering for his indispensable support, suggestions. In the end our special thanks to Ms. V. M. Barkade for providing all needed resources and valuable suggestionsfor our Project.

7. CONCLUSION

This paper introduces PhishStorm, an efficient phishing URL detection system relying on URL lexical analysis. The approach is based on the intra-URL relatedness. This relatedness reflects the relationship among the words blended into a URL and particularly into the part of the URL that can be freely defined and the registered domain. We make use of search engine query data in order to extract 6 features from aURL.Future work will consist in releasing components of the tools as an add-on for a Web browser such as Mozilla Firefox. In addition, the technique proposed in [6], that is complementary to one introduced in this paper, can be merged to create a phishing detection system with a larger scope of action. The proposed system can be implemented for yahoo, twitter provided the API's for it.

8. REFERENCES

[1] S.Marchal, J. François, R. State, and T. Engel, "PhishScore: Hacking phishers minds," in Proc. 10th Int. CNSM, 2014, pp. 46–54.

[2] Yanhui Du, Fu Xue "Research of the Anti-Phishing Technology based on E-mail extraction and analysis."

[3] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel "PhishStorm: Detecting With Streaming Analytics"

[4] T. Segaran and J. Hammerbacher, "Beautiful Data: The Stories Behind Elegant Data Solutions."Sebastopol, CA, USA: O'Reilly Media, 2009.

[5] M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In CRYPTO, pages 162–177, 2002.

[6] S. Marchal, J. François, R. State, and T. Engel, "Proactive discovery of phishing related domain names,"in Research in Attacks, Intrusions, and Defenses. New York, NY, USA: Springer-Verlag, 2012, pp. 190–209

[7] M. Hall *et al.*, "The WEKA data mining software: An update,"*ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, Jun. 2009.

[8] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL names say it all," in *Proc. IEEE INFOCOM*, 2011, pp. 191–195.

[9] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishingdetection," in *Proc. 4th Int. Conf. Security Privacy Commun. Netw.*, 2008,pp. 1–6, ACM.

[10] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists:Learning to detect malicious web sites from suspicious URLs," in *Proc.15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 1245–1254, ACM.

[11] J. Zhang, P. Porras, and J. Ullrich, "Highly predictive blacklisting," in*Proc. 17thConf. Security Symp.*, 2008, pp. 107–122,USENIXAssociation.