Detecting Phishing Websites using Automation of Human Behavior

Routhu Srinivasa Rao, Alwyn R Pais
Information Security Research Lab
National Institute of Technology
Karnataka, India 575025
{routh.srinivas,alwyn.pais}@gmail.com

ABSTRACT

In this paper, we propose a technique to detect phishing attacks based on behavior of human when exposed to fake website. Some online users submit fake credentials to the login page before submitting their actual credentials. He/She observes the login status of the resulting page to check whether the website is fake or legitimate. We automate the same behavior with our application (FeedPhish) which feeds fake values into login page. If the web page logs in successfully, it is classified as phishing otherwise it undergoes further heuristic filtering. If the suspicious site passes through all heuristic filters then the website is classified as a legitimate site. As per the experimentation results, our application has achieved a true positive rate of 97.61%, true negative rate of 94.37% and overall accuracy of 96.38%. Our application neither demands third party services nor prior knowledge like web history, whitelist or blacklist of URLS. It is able to detect not only zero-day phishing attacks but also detects phishing sites which are hosted on compromised domains.

Keywords

phishing; anti-phishing; automation; selenium; heuristics

1. INTRODUCTION

Phishing is an attack which targets online users for extraction of their sensitive information such as username, password and credit card information etc. According to APWG (2016) report [6],there has been a significant increase of phishing attacks in the first quarter of 2016. It has a growth of 250% compared to last quarter of APWG (2015) [5] comprising of 289,371 attacks. There are many tools [1, 4] which create a replica of entire website and are used for hosting phishing attacks. This makes even a fresh phisher to design phishing sites easily. But, when the phisher uses an entire replica of website, there are more chances to get caught by the anti-phishing tools. It is due to the presence of website identity elements of legitimate site. Some of the website identity elements used are title, keywords, copyright, anchor links, DOM etc.

Therefore, to bypass the anti-phishing techniques, attackers try to copy only login pages followed by a login successor page of his own or might redirect to target website. The attackers also design phishing websites with website identifiers removed or misspelled. The techniques which depend on website identity elements may fail to detect above kind of phishing websites [24, 13, 25]. The works which are dependent on third-party services may fail to detect phishing sites which are hosted on compromised legitimate domain [8, 29, 27, 20, 11]. Sometimes attackers may embed legitimate content in an Iframe of phishing page such that anti-phishing techniques [27, 8, 25, 31, 29, 11, 17, 18] which depend on source code of website could be bypassed. In this paper, we address the detection of above mentioned phishing websites with our FeedPhish application.

Traditional approaches like Blacklist [30, 19] and Whitelist [7] techniques works efficiently until the given URL is on the list. If the URL or website is not in the list, the techniques fail to detect them. Those URLs which are out of the list are called as zero-day phishing URLs. These kinds of attacks are either countered by heuristic-based techniques [14, 9, 23, 22] or visual based techniques [10, 28, 12, 21].

Visual based techniques need a large database of images for comparison of suspected website's image with trusted image database. As this comparison is very costly with respect to time and space, we choose heuristic based detection to detect the phishing websites.

Heuristic-based techniques are developed based on the features involved in designing the phishing websites. The features which are mostly common and unique in many of the existing and old phishing websites are taken as parameters to detect new phishing websites.

The Main contributions of our paper are as follows:

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.

To the best of our knowledge, our work is the first to use Selenium WebDriver [3] to automate the login process in each of the websites visited. Earlier works [26, 24, 16, 15] used Selenium or PhantomJS to capture screenshots of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CPSS'17, April 02 2017, Abu Dhabi, United Arab Emirates

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-4956-7/17/04...$15.00

DOI: http://dx.doi.org/10.1145/3055186.3055188

the visited websites or to crawl the website for anchor links, plain text of the web page.

The paper is organized as follows. We discussed some of the existing anti-phishing techniques and compared the same with our work in Section 2. The Proposed work is explained in Section 3. Experimentation and results are given in Section 4. Limitations are discussed in Section 5. We conclude the paper in Section 6.

2. RELATED WORK

In this section, we describe some of the existing heuristic-based techniques used to detect phishing websites.

Varshney et al. [27] proposed a Lightweight Phish Detector (LPD) which detects phishing sites based on the results of Google search engine. The domain is extracted from URL of suspicious website and appended with title of the website to form a search string. This search string is fed to search engine using Google search API. The domain of suspicious website is compared with top N results of search engine to identify the legitimacy of a suspicious website.

Chiew et al. [8] proposed logo based technique to detect phishing sites. Authors used machine learning algorithms to extract logo from a suspicious website. The logo is fed to Google image search engine to get the portrayed identity of a suspicious website. Based on the domain comparisons of search engine results and suspicious website domain the legitimacy of a website is calculated.

Ramesh et al. [20] proposed a technique which performs intersection operation on direct links set and indirect links set of suspicious site to detect phishing behavior. The direct links set is extracted from DOM of suspicious site where as indirect links set is extracted from search engine results. Target domain of phishing website is calculated from result set of intersection. Target domain's IP address is compared with suspicious website's IP address to calculate the legitimacy of website.

PhishWho[25] technique classifies websites based on the difference between suspicious website identity and targeted website. N-gram model is used to extract website identity keywords from textual content of the website. These keywords are fed to the search engine for evaluating the target website domain from the search engine results. Comparison of target website domain with suspicious domain is checked to identify the legitimacy of a website.

Cantina [31] technique detects phishing sites based on the textual content of website. It uses a well known data mining algorithm called Term Frequency-Inverse Document frequency for extracting highly important words from the website content. TF-IDF assigns weights to each word in the website and top most weighed words are fed to search engine to identify the legitimacy of website.

Prakash et al. [19] proposed a black list based approach to detect phishing sites. They proposed heuristic features which predicts new phishing URLs based on the combinations of existing URLs. They performed combinations operation on IP address, top level domains (TLD), Directory structure, Query string and Brand names of existing phishing URLS to generate new phish.

Cantina+ [29] technique is an extension to cantina where additional heuristic features are included to improve the accuracy of a model. Unlike Cantina, authors in Cantina+ experimented their framework with a large dataset consisting of 4883 legitimate and 8118 phishing websites. However, as Cantina+ uses same TF-IDF and search engine based features, it also faces the same limitations of Cantina.

Garera et al. [11] proposed an approach which uses structure of a URL as parameter to detect phishing sites. Authors proposed features based on page rank, whitelist, URL obfuscations and words. They studied structure of URLs of various phishing websites and found that detection of phishing sites is possible without the information of source code of a web page.

Moghimi & Varjani [17] proposed a technique which uses SVM_DT algorithm to detect phishing sites. SVM_DT is the combination of support vector machine and decision tree algorithm. SVM is used to train the model with the given dataset and then rules are generated from input and output relationship existing within the model. New rules are generated using decision tree algorithm C4.5. From these rules, they are able to detect phishing sites which target banking applications. The authors concentrated on frequency of secure HTTPS anchor links, secure images, secure scripts and secure styles within a website. Higher the frequency of secure objects, higher the chance to be a legitimate site.

Pan & Ding [18] proposed an approach which detects phishing sites based on anomalies present in the web page. Authors claim that there exists a discrepancy when phishing sites attempts to fake visual similarity of a legitimate site. The discrepancy may include website identity, HTTP transactions and structure of the website. Anchor links, title, DNS record, Form handler and SSL certificate are used as features to detect the phishing sites.

We provide the summary of existing works in comparison to our work with five features in Table 1. The features include detection of phishing sites hosted on compromised domains; detection of phishing sites which use embedded objects (iframes) for displaying login page; detection of phishing sites without the use of third-party services like Search engine features, Page rank and WHOIS database, detection of unidentified phishing sites(zero-day), method to identify target website of a phishing site.

3. PROPOSED WORK

In this section, we present the working of our approach in detecting phishing sites. This is achieved by feeding fake values into login fields, followed by further filtering process to be either classified as a phishing site or legitimate site. The entire work is divided into three modules which perform filtering process at each level. Hence, we also term these modules as filters. The filters of our approach are *LoginCheck*; *FeedFakeCredentials*; and *HeuristicsCheck*. The flowchart of our application is shown in Figure 1. To ease our discussion we define the following terms.

- **login page** The web page which seeks sensitive information such as username and password.
- **login successor page** The resulting web page upon submission of sensitive information.
- **target website** The website which is imitated by the attacker to perform phishing attack.
- **website identity** The real brand name of a website.
- **compromised domain** The domain which is hacked by the attackers.

Table 1: Comparison of existing works

Existing work	Third-party independence	Compromised domains	Zero-day attacks	Target website detection	Embedded objects
Varshney et al. [27]	no	no	yes	no	no
Chiew et al. [8]	no	no	yes	no	no
Ramesh et al. [20]	no	no	yes	yes	yes
PhishWho[25]	no	no	yes	yes	no
Cantina [31]	no	no	yes	no	no
Prakash et al. [19]	no	no	no	no	no
Cantina+ [29]	no	no	yes	no	no
Garera et al. [11]	no	no	yes	no	no
Moghimi & Varjani [17]	yes	yes	yes	no	no
Pan & Ding [18]	no	no	yes	yes	no
our work	yes	yes	yes	yes	yes

3.1 LoginCheck

This module checks for the presence of login page in a given URL. We observed that login page can be found at three locations. 1) Home page of URL. 2) Modal window. 3) Iframes. Attackers use Iframes to load the imitated content such that techniques which are dependent on source code of the website will fail to detect. It is because the content in Iframe is hidden from the source code of given URL. Hence, to counter these kind of phishing attacks, we connect to each Iframe of the given URL for the presence of login page unless it is not found in previous locations. The flow chart of the module is given in Figure 2.

Our application FeedPhish takes URL as input and invokes Selenium WebDriver, which open Firefox browser externally to perform browser like action as an automated process. Selenium is a web browser automation tool which helps in constructing browser-based regression automation suites and tests.

Once the browser is opened, it navigates to the given URL and checks for password field in the web page. If found, FeedFakeCredentials module is initiated else it checks for the modal window login pages. The anchor link texts are compared with keywords set (signin or sign in or log in or login) to identify login modal window. Once the password field is found in modal window login page, FeedFakeCredentials module takes the control otherwise the entire website is crawled for Iframes n. Each Iframe i is navigated for the presence of password field. If Password field is found FeedFakeCredentials module is initiated or else the website skips detection process and is classified as a legitimate website. We believe that attackers target online users for sensitive information which is mostly acquired from login pages. Hence the web pages with no login forms are classified as legitimate sites.

3.2 FeedFakeCredentials

This module is initiated, once the login page is detected. The WebDriver identifies the input fields of username and password with input selectors as identifiers. We automate the process of feeding fake credentials into the input fields with the help of Selenium WebDriver and Java programming language. We submit fake values to the respective login page with submit() method. We used submit() method instead of click() because submit() method can be used on any input field where as click() method has to be applied only on button. This submit() method avoids client side validation or JavaScript action attached to submit button leading to submit invalid fake values to the websites.

Once the fake values are submitted, the resulting page i.e. login successor page is set as a parameter to detect legitimacy of the website. The absence of password field in login successor page is classified as a phishing website. It should be noted that client side validation attached to the form is not addressed in this work. Due to this, some phishing sites which have validation check on the form are able to bypass this filter.

The rationale behind this idea is that attackers never use any database authentication while extracting sensitive information. Attacker's main target is to collect the sensitive information rather than validation of sensitive information. We found 87.84 % of phishing sites identified under this category. Some of the phishing websites, upon receiving the sensitive information, redirects the login successor page to legitimate web page. These kind of phishing sites are identified through the domain conflict checking i.e. primary domain of login page and its successor page is compared. If comparison is unsuccessful then the website is classified as phishing site. There are some other phishing sites which always respond with failure login page on submission of fake credentials. The remaining 12.16 % of phishing sites comes under above failure login page category. This kind of behavior is as similar to legitimate sites. Because, legitimate sites also respond with failure login page when wrong credentials are submitted. The above category of websites undergoes Heuristic filter process for identifying phishing behavior.

3.3 HeuristicsCheck

This module attempts to distinguish between phishing and legitimate websites when they respond with failure login page on submission of fake credentials. This module addresses the problem with two features. It checks for the presence of defined phishing features in the suspicious website. If found it is classified as a phishing site else legitimate site. The flow chart of the module is given in Figure 3.

3.3.1 Zero links in the body of source code:

This feature checks for the presence of anchor links in body of source code. We observed web pages that seek sensitive information have at least a single anchor link such as "forgot password" or "help" or "contact us" or "about us" etc. This feature counters the phishing sites which mimic the legitimate site by replacing the entire text of legitimate

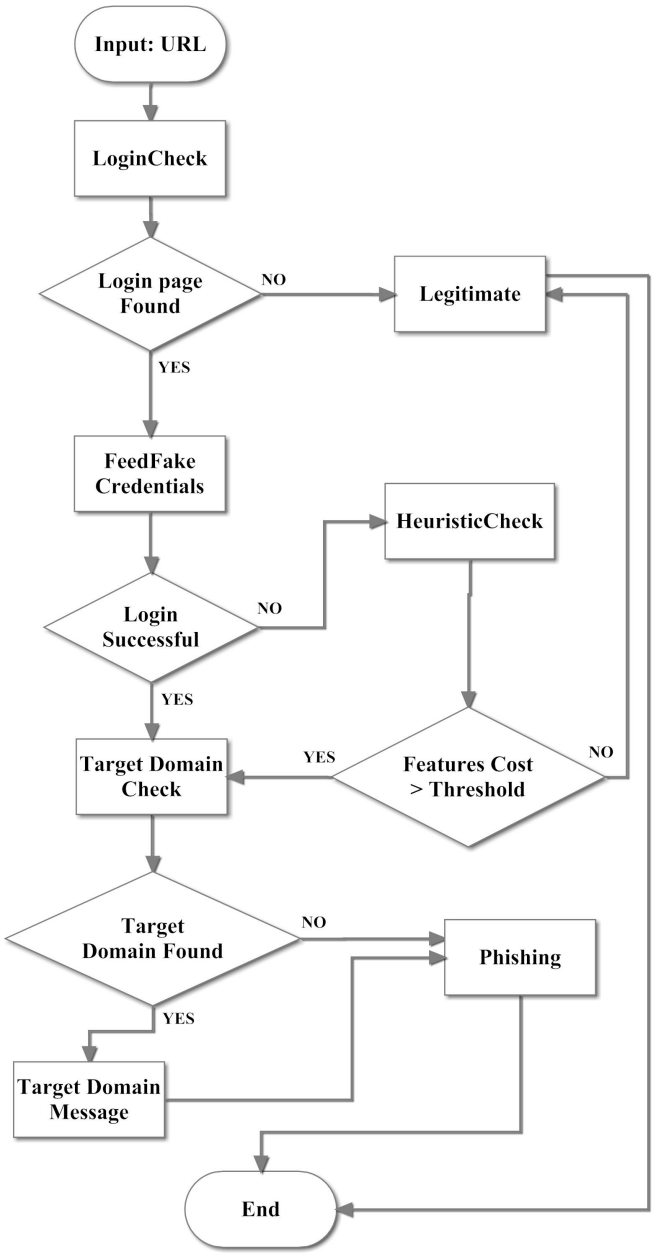

Figure 1: Architecture of our application

content with a single image. So, the websites which have no anchor links in the body of HTML is classified as phishing otherwise it undergoes next filtering process.

3.3.2 Common Page Redirection ratio (CPRR):

This feature calculates the ratio of anchor links pointing to common page out of total number of links. Attackers may redirect a few or all of the links in login page to a common page. The common page can be of his own successful login page or current login page (null link) or may redirect to target legitimate website. We believe that attackers take less effort in designing phishing site by mimicking the login page and redirects all/some of the links

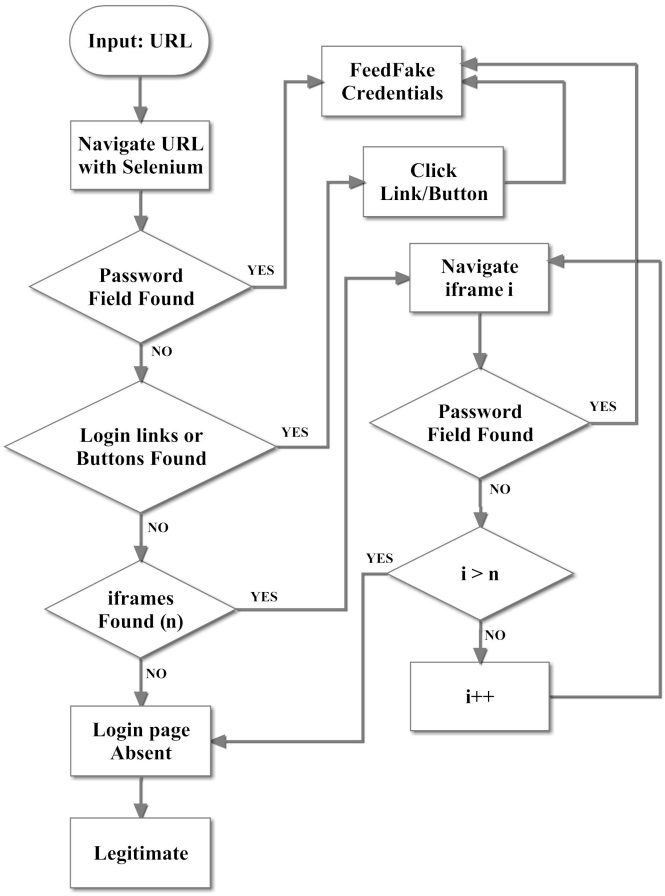

Figure 2: flow chart of LoginCheck module

to respective target website links or his own login successor page. Usually, null links are used to redirect a link to its current page. Hence attackers use this technique to make users stay on the same login page by assigning all or some of the anchor links to the null value (#). For example, $< a\ href = \text{“}\#\text{”} > forgotpassword < /a >$ or $< a\ href = \text{“}\#someid\text{”} > contactus < /a >$.

Also, some phishers try to bypass null links based anti-phishing tools by replacing null links with its own page. For example, a login phishing website with URL http://www.signin.ebaye.com/secure/login/index.php has five anchor links in which all or most of its links may point to its own successor page like http://www.signin.ebaye.com/secure/loginsuccess/success.php. Usually, the anchor links in legitimate sites point to different web pages of same local domain. This fact of observation made us propose this feature in detecting phishing behavior. We set a threshold of 0.75 based on our experimental results to achieve better results.

$$\begin{cases} \frac{al_{f_w}}{al_{n_w}} & \text{if } al_{n_w} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where al_{f_w} is frequency of most frequent anchor link and al_{n_w} is total number of anchor links in a website.

3.4 TargetDomainCheck

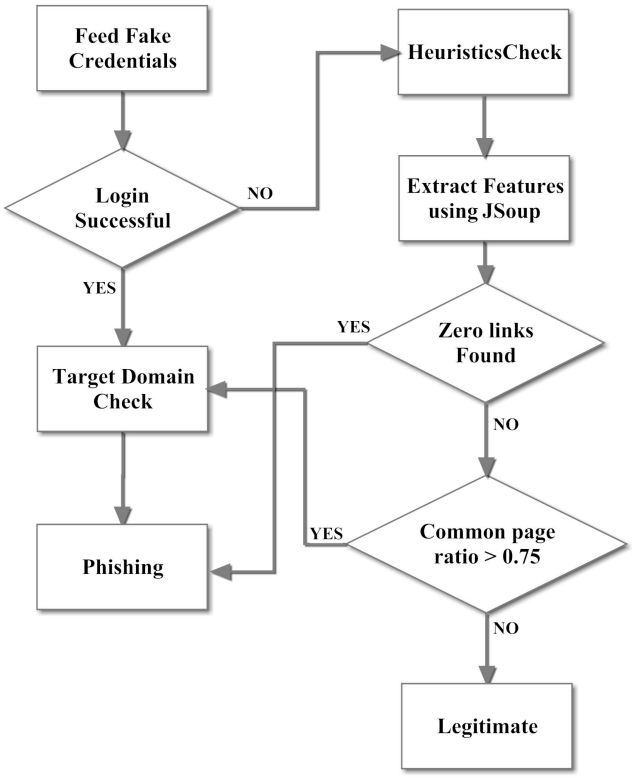

Figure 3: flow chart of HeuristicsCheck module

We compute identity of the website based on the frequency of the domains in anchor links. The domain with maximum frequency is declared as an identity of that website. This module gives the target of a phishing site only when the phishing site mimics anchor links of legitimate site. Hence, the websites which gets identified by the Zero Links filter will not undergo Target Domain Check process as shown in Figure 3. For most of the legitimate sites, Target domain and local domain will be same where as for the phishing sites they are different. It should be noted that phishing sites which contain more number of null (#) links or common page redirection also have same local domain and target domain.

4. EXPERIMENTATION AND RESULTS

Our application FeedPhish is implemented in Java platform. It takes URL as input and outputs the status of website as legitimate or phishing site along with target website. We have used Selenium WebDriver for automation of feeding fake credentials into login fields such as username and password. Use of Selenium invokes a browser externally, with input URL given by the user in the application. The browser can be either a chrome or Firefox or opera or explorer. We have used Firefox as browser for opening the URLs. Prior to this, validation of login page is performed to apply the phishing detection process.

We used Jsoup library [2] for extracting hyperlinks of suspicious websites. Jsoup is a HTML parser which provides convenient API for extracting, finding, manipulating data

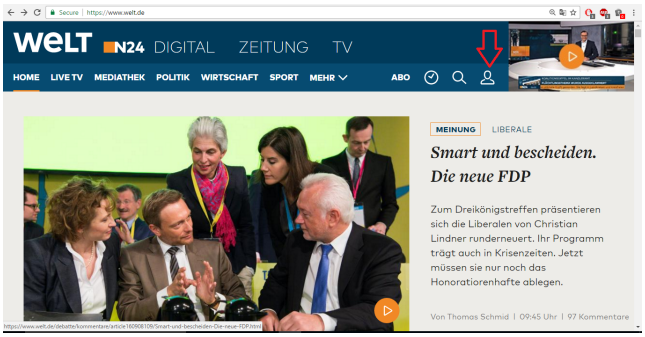

Figure 4: Modal window LoginCheck failure

from a URL. Heuristic features such as zero links in the body of HTML, Common Page Redirection ratio (CPRR), and identity of the website are calculated using Jsoup API.

We have collected 2342 websites for checking the status of legitimacy, out of which 1459 are phishing sites and 883 are legitimate sites. We randomly picked legitimate sites from 50000 websites of Alexa database such that less traffic websites are also been considered. The Attacker attempts to hack the less traffic domains as it is less secured website and easy to compromise the domain. Hence, to prevent biasing on popular websites, we considered random websites from a large database unlike considering from top 1000 or 3000 websites. All the URLS are collected from two well-known streams. Phishing sites are collected from PhishTank repository and legitimate sites are collected from Alexa database as shown in Table 2.

The performance of LoginCheck module is assessed in detecting login pages of a given URL. The results are shown in Table 3. 98.64% of legitimate login pages and 97.53% of phishing login pages are correctly detected by this module. The failure detection of login pages in both legitimate and phishing sites is due to the different designs followed by the websites. The websites which design their password field with *type=text* input type instead of *type=password* are considered as non-login pages. This behavior is mostly found in phishing sites and less reputed websites. These failure cases can be reduced by searching for keywords of password input fields in the websites. Websites with Modal window seeking login information may not have any sign in anchor link or button instead it may have an image or icon to access the login modal window as shown in Figure 4. The attempt to detect the login page for the above cases through LoginCheck module are unsuccessful.

We considered only successful login page detected sites as dataset for the experimentation in our application. We have considered five metrics to calculate the performance of our application in detecting phishing websites. *True positive* (phishing sites classified as phishing), *True Negative* (legitimate sites classified as legitimate), *False positive* (legitimate classified as phishing site), *False negative* (phishing site is classified as legitimate), *Accuracy* (rate of correctly classified instances. The performance results of our work is given in Figure 5.

We achieved False positive rate (FPR) of 5.63 % due to the following scenarios.

1. On use of submit() method on the form, we are able

Table 2: Source of websites

Type	Source	Total instances	Link
legitimate	Alexa Global Sites	883	http://www.alexa.com/topsites
phishing	PhishTank	1459	https://www.phishtank.com/developer_info.php

Table 3: Result of LoginCheck module

Type	Total instances	Successful detection	Accuracy(%)
legitimate	883	871	98.64
phishing	1459	1423	97.53

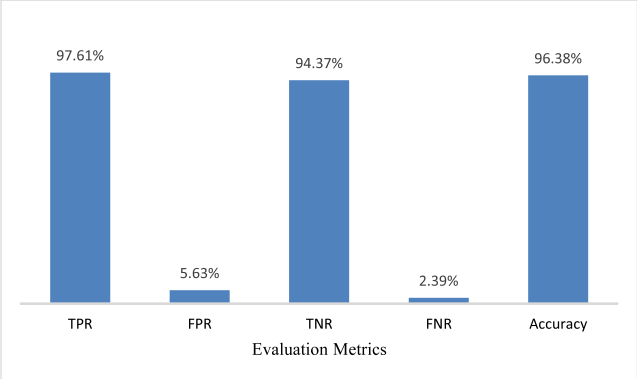

Figure 5: Performance results of our FeedPhish application

to bypass client side validation involved on submit button but due to which we also arrive with not found page or undesired login successor page as shown in Figure 6. For example, consider a login page of website with URL https://bforexlogin.tradenetworks.com which seeks login information as shown in Figure 6. When we submit the sensitive information with click() method we get the output as shown in Figure 6 (b) whereas if we submit the login information with submit() method it results the undesirable output as shown in Figure 6 (d). We can find login page before submit operation in (a) and URL of website in (c).

2. On submitting fake credentials, some legitimate sites resulted "incorrect username or password" or reset anchor link page as a login successor page in place of displaying repeatable login page.

3. On performing submit() method on login modal window of some legitimate sites, we reach home page as login successor page in place of arriving at modal window login interface.

In the above cases, FeedPhish is classifying the website as phishing resulting in false positive rate of 5.63 %. It should be noted that, most of the legitimate websites which are classified as phishing are either less reputed or poorly designed websites. We achieved 2.39% false negative rate (FNR) as phishing sites bypassed all filters (FeedFakeCredentials, zero links, CPRR) in the application.

Before discussing contribution of each filter, we explain the selection of threshold (t) for CPRR. Out of total 2294 websites, 1276 websites are classified by feedFakeCredential filter i.e. websites which have successfully logged in. 108 out of remaining 1018 are filtered by zero links filter i.e. websites which have no links. For the remaining 910 websites, common page redirection ratio filter (CPRR) is applied with different thresholds(t) ranging from 0 to 1 and corresponding false positive rate (FPR) and false negative rate (FNR) are calculated as shown in Table 4. It should be noted that, for a better anti-phishing technique, FPR and FNR should be as minimum as possible.

From Table 4, it is observed that, at threshold t=0.75, all the remaining 910 websites are correctly classified with an acceptable percentage of FPR and FNR of 1.56 % and 45.33% respectively. It is also observed that least misclassification rate of 5.16 % is found at $t = 0.75$. We also found that false positive rate (FPR) increasing on increase of threshold and false negative rate (FNR) decreasing on decrease of threshold. Misclassification rate is the number of misclassified websites out of total number of websites. Overall Accuracy measures total accuracy calculated with combination of all three filters. Predicted Phish represents the number of websites which are classified as phishing sites based on the given threshold t. The websites which are above and equal to t are classified as phishing otherwise legitimate. L and P represents actual status of website as legitimate and phishing respectively.

For instance, consider a threshold $t = 0.75$, 54 websites are above or equal to t which are classified as predicted phish out of 910 total websites. Remaining 856 (910-54) are classified as legitimate sites. Out of 54 predicted phish, 13 legitimate sites are misclassified as phishing and out of remaining 856 (910-54) predicted legitimate websites, 34 (75-41) phishing sites are misclassified as legitimate site. Hence, total Misclassification rate with respect to CPRR filter leads to 5.16 % ((34+13)/910). The Misclassification rate for each filter is calculated and added to get the total Misclassification rate (TMCR) for FeedPhish application. This parameter (TMCR) is used to calculate the overall Accuracy of the application. We observed that, FeedPhish achieved maximum accuracy of 96.382 % at threshold 0.75.

Note that the threshold $t = 0.75$ is set just based on the empirical experience. It is not guaranteed to be optimal because attackers may employ different strategies while designing the new phishing websites. Hence, the experimental evaluation results distinct thresholds for different datasets.

The contribution of each filter in detecting phishing sites is given in Figure 7. FeedFakeCredentials filter shared major contribution with 87.84% in detecting phishing websites followed by zero links filter with 6.89% detection ratio. We also calculated the performance of each individual filter in detecting phishing websites.

The Performance of individual filter in detecting phishing sites is given in Table 5. FeedFakeCredentials filter has given significant results than the other filters. The above filter is able to detect 87.84% of total phishing sites followed by Zero Links filter with a detection ratio of 52.35%.

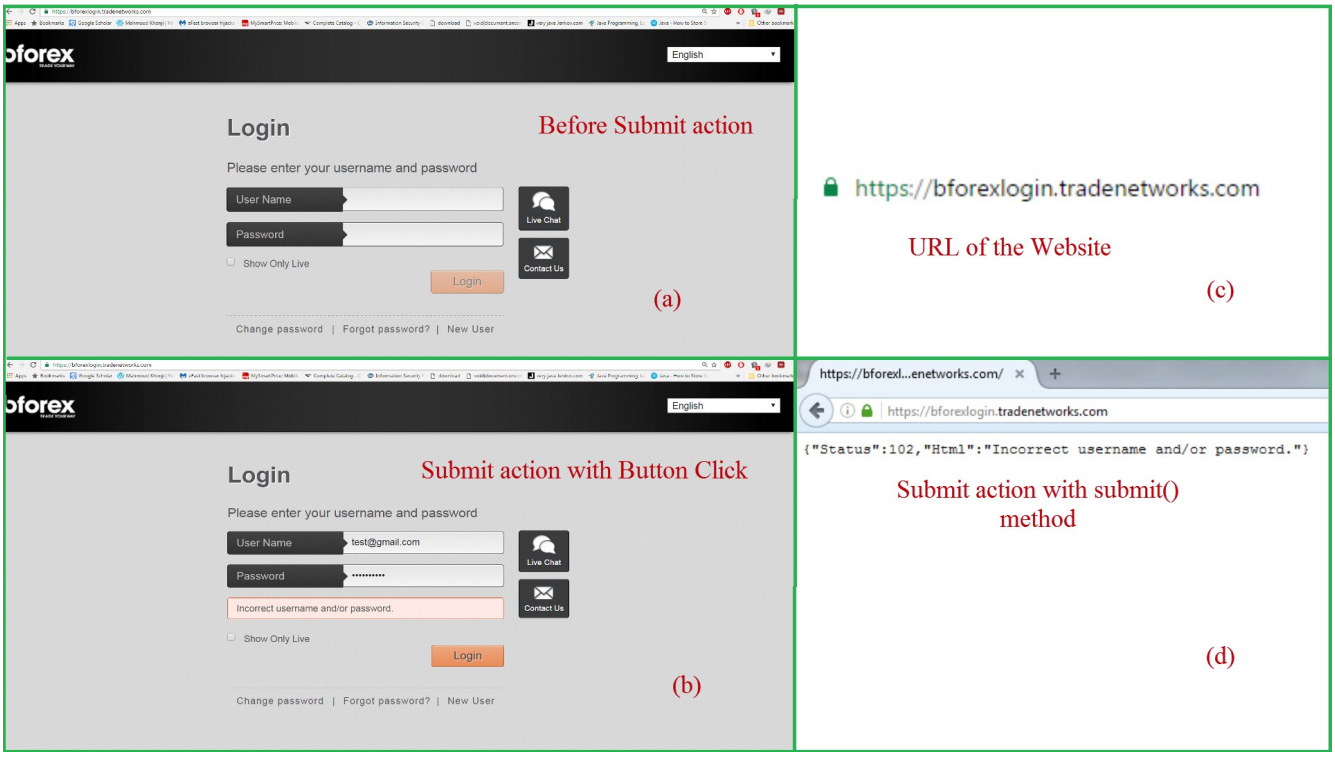

Figure 6: False positive case in our FeedPhish application

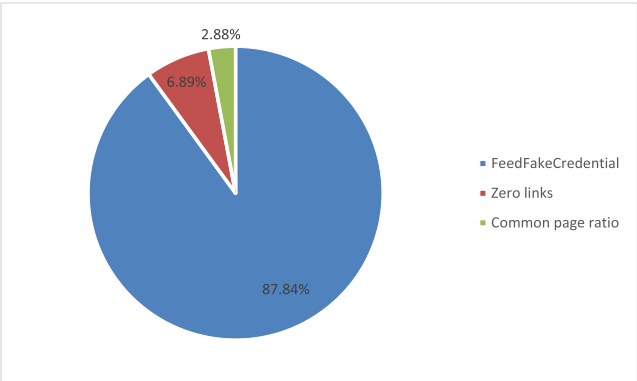

Figure 7: Contribution of each filter in detecting phishing sites

Our application is independent of third-party services like search engine, page ranking and WHOIS. Hence, it is able to detect compromised domains too. Out of 1423 phishing websites, our application found 239 compromised websites. We used WHOIS database to calculate the age of a domain of phishing site such that those websites with high age are manually checked whether the phishing site is hosted on compromised domain or not.

5. LIMITATIONS

This section explains the limitations of our application. Firstly, our application is able to deal login modal window which contains anchor text within the specified keyword set (sign in, signin, login, log in) but it fails to handle if the anchor text is replaced with an icon or an image.

Secondly, we did not consider Single Sign-On (SSO) websites as they use JavaScript functions to handle different login interface for different websites. However, JavaScript actions can be executed through selenium but it increases time complexity of the detection process as they are executed at run time.

6. CONCLUSION

In this paper, we proposed a technique which detects phishing sites based on automation of human behavior for submitting sensitive information. It also identifies target domain of a phishing website. It neither depends on third-party services such as search engines, WHOIS or page rank nor needs any prior knowledge of websites like history, training data, blacklist or whitelist. Due to which, it is able to

Table 4: Calculation of Threshold value

Predicted Phish	Threshold(t)	L	P	FPR(%)	FNR(%)	Misclassification rate(%)	Overall Accuracy(%)
8	1	5	3	0.59	97.33	8.46	95.075
9	0.95	6	3	0.72	97.33	8.57	95.031
12	0.9	7	5	0.84	93.33	8.46	95.075
44	0.85	11	33	1.32	56	5.82	96.121
49	0.8	11	38	1.32	49.33	5.27	96.339
54	**0.75**	**13**	**41**	**1.56**	**45.33**	**5.16**	**96.382**
55	0.7	14	41	1.67	45.33	5.27	96.339
64	0.65	23	41	2.75	45.33	6.15	95.99
68	0.6	27	41	3.23	45.33	6.70	95.772
74	0.55	33	41	3.95	45.33	7.36	95.511
109	0.5	67	42	8.02	44	10.99	94.072
112	0.45	69	43	8.26	42.66	11.09	94.028
130	0.4	86	44	10.29	41.33	12.86	93.33
191	0.3	144	47	17.24	37.33	18.90	90.94
323	0.2	268	55	32.09	26.66	31.64	85.88
514	0.1	454	63	54.37	16	51.21	78.12
910	0	835	75	100	0	91.76	62.04

Table 5: Performance of individual filter

Filters	Phishing instances	Phishing detections	Ratio(%)
FeedFakeCredential	1423	1250	87.84
Zero link	1423	745	52.35
Common page redirection ratio	1423	575	40.41

detect phishing websites which are hosted on compromised domains. It is also able to detect phishing sites containing embedded HTML files which most of the existing techniques fail to address. It also detects phishing websites which use captcha verification in the login page. In the future, we intend to include additional heuristics to detect phishing websites which use JavaScript events on links or input fields. We also would like to address the Single Sign-On phishing websites.

7. REFERENCES

[1] HTTrack Website Copier - Free Software Offline Browser (GNU GPL). https://www.httrack.com/.

[2] Jsoup Java HTML Parser, with best of DOM, CSS, and jquery. https://jsoup.org/.

[3] Selenium. http://docs.seleniumhq.org/download/.

[4] Wget - GNU Project - Free Software Foundation. https://www.gnu.org/software/wget/.

[5] Phishing attack trends reports, 4th quarter 2015. http://docs.apwg.org/reports/apwg_trends_report_q4_2015.pdf, 2015. Accessed: 2016-06-01.

[6] APWG. Phishing attack trends reports, first quarter 2016. http://docs.apwg.org/reports/apwg_trends_report_q1_2016.pdf, 2016. Accessed: 2016-06-01.

[7] Y. Cao, W. Han, and Y. Le. Anti-phishing based on automated individual white-list. In *Proceedings of the 4th ACM workshop on Digital identity management*, pages 51–60. ACM, 2008.

[8] K. L. Chiew, E. H. Chang, W. K. Tiong, et al. Utilisation of website logo for phishing detection. *Computers & Security*, 54:16–26, 2015.

[9] N. Chou, R. Ledesma, Y. Teraguchi, J. C. Mitchell, et al. Client-side defense against web-based identity theft. In *NDSS*, 2004.

[10] A. Y. Fu, L. Wenyin, and X. Deng. Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd). *IEEE transactions on dependable and secure computing*, 3(4):301–311, 2006.

[11] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*, pages 1–8. ACM, 2007.

[12] M. Hara, A. Yamada, and Y. Miyake. Visual similarity-based phishing detection without victim site information. In *Computational Intelligence in Cyber Security, 2009. CICS'09. IEEE Symposium on*, pages 30–36. IEEE, 2009.

[13] M. He, S.-J. Horng, P. Fan, M. K. Khan, R.-S. Run, J.-L. Lai, R.-J. Chen, and A. Sutanto. An efficient phishing webpage detector. *Expert Systems with Applications*, 38(10):12018–12027, 2011.

[14] Y. Joshi, S. Saklikar, D. Das, and S. Saha. Phishguard: A browser plug-in for protection from phishing. In *Internet Multimedia Services Architecture and Applications, 2008. IMSAA 2008. 2nd International Conference on*, pages 1–6. IEEE, 2008.

[15] H. Kazemian and S. Ahmed. Comparisons of machine learning techniques for detecting malicious webpages. *Expert Systems with Applications*, 42(3):1166 – 1177, 2015.

[16] P. Mensah, G. Blanc, K. Okada, D. Miyamoto, and Y. Kadobayashi. Ajna: Anti-phishing js-based visual analysis, to mitigate usersâĂŹ excessive trust in ssl/tls.

[17] M. Moghimi and A. Y. Varjani. New rule-based

phishing detection method. *Expert systems with applications*, 53:231–242, 2016.

[18] Y. Pan and X. Ding. Anomaly based web phishing page detection. In *Proceedings - Annual Computer Security Applications Conference, ACSAC*, volume 6, pages 381–392, 2006.

[19] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta. Phishnet: Predictive blacklisting to detect phishing attacks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5. IEEE, 2010.

[20] G. Ramesh, I. Krishnamurthi, and K. S. S. Kumar. An efficacious method for detecting phishing webpages through target domain identification. *Decision Support Systems*, 61:12 – 22, 2014.

[21] R. S. Rao and S. T. Ali. A computer vision technique to detect phishing attacks. In *Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on*, pages 596–601. IEEE, 2015.

[22] R. S. Rao. and S. T. Ali. Phishshield: A desktop application to detect phishing webpages through heuristic approach. *Procedia Computer Science*, 54:147–156, 2015.

[23] H. Shahriar and M. Zulkernine. Trustworthiness testing of phishing websites: A behavior model-based approach. *Future Generation Computer Systems*, 28(8):1258–1271, 2012.

[24] C. L. Tan, K. L. Chiew, et al. Phishing website detection using url-assisted brand name weighting system. In *2014 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 054–059. IEEE, 2014.

[25] C. L. Tan, K. L. Chiew, K. Wong, and S. N. Sze. Phishwho: Phishing webpage detection via identity keywords extraction and target domain name finder. *Decision Support Systems*, 88:18 – 27, 2016.

[26] T. Van Goethem, F. Piessens, W. Joosen, and N. Nikiforakis. Clubbing seals: Exploring the ecosystem of third-party security seals. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 918–929. ACM, 2014.

[27] G. Varshney, M. Misra, and P. K. Atrey. A phish detector using lightweight search features. *Computers & Security*, 62:213 – 228, 2016.

[28] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng. Detection of phishing webpages based on visual similarity. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1060–1061. ACM, 2005.

[29] G. Xiang, J. Hong, C. P. Rose, and L. Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security (TISSEC)*, 14(2):21, 2011.

[30] J. Zhang, P. A. Porras, and J. Ullrich. Highly predictive blacklisting. In *USENIX Security Symposium*, pages 107–122, 2008.

[31] Y. Zhang, J. I. Hong, and L. F. Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648. ACM, 2007.

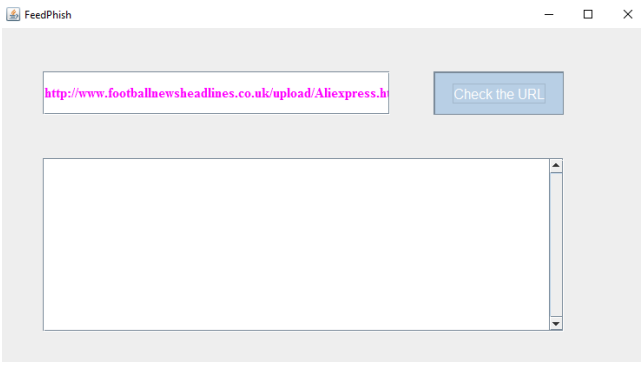

Figure 8: Interface of our FeedPhish application

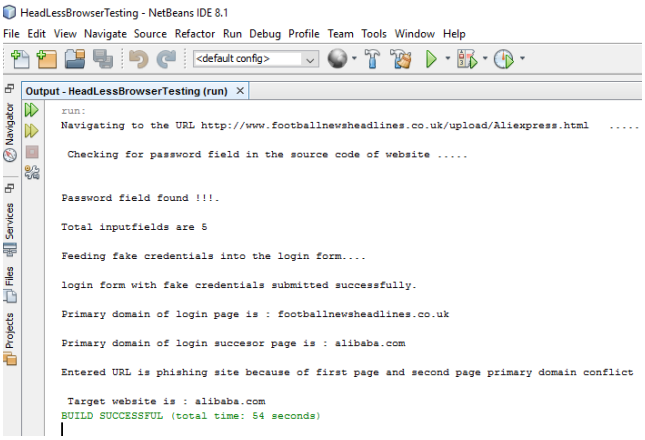

Figure 9: System Output

APPENDIX

A. PHISHING EXAMPLE

In this section, we present the demonstration of detection of phishing site with an example. Figure 8 shows the interface of our application which takes input as suspicious URL and outputs the status of website as legitimate or phishing. We consider a live phishing site from PhishTank database and is fed to the FeedPhish application. The suspicious website undergoes filtering process to be classified as a legitimate or phishing site. In this example, on submitting the fake values to a login form, the login successor page redirects to target legitimate site. Hence, the website is classified as phishing site based on the domain conflict of websites. We can also find the primary domain of login page and login successor page in Figure 9.The output interface of our application is shown in Figure 10. The phishing site of our example can be seen in Figure 11.

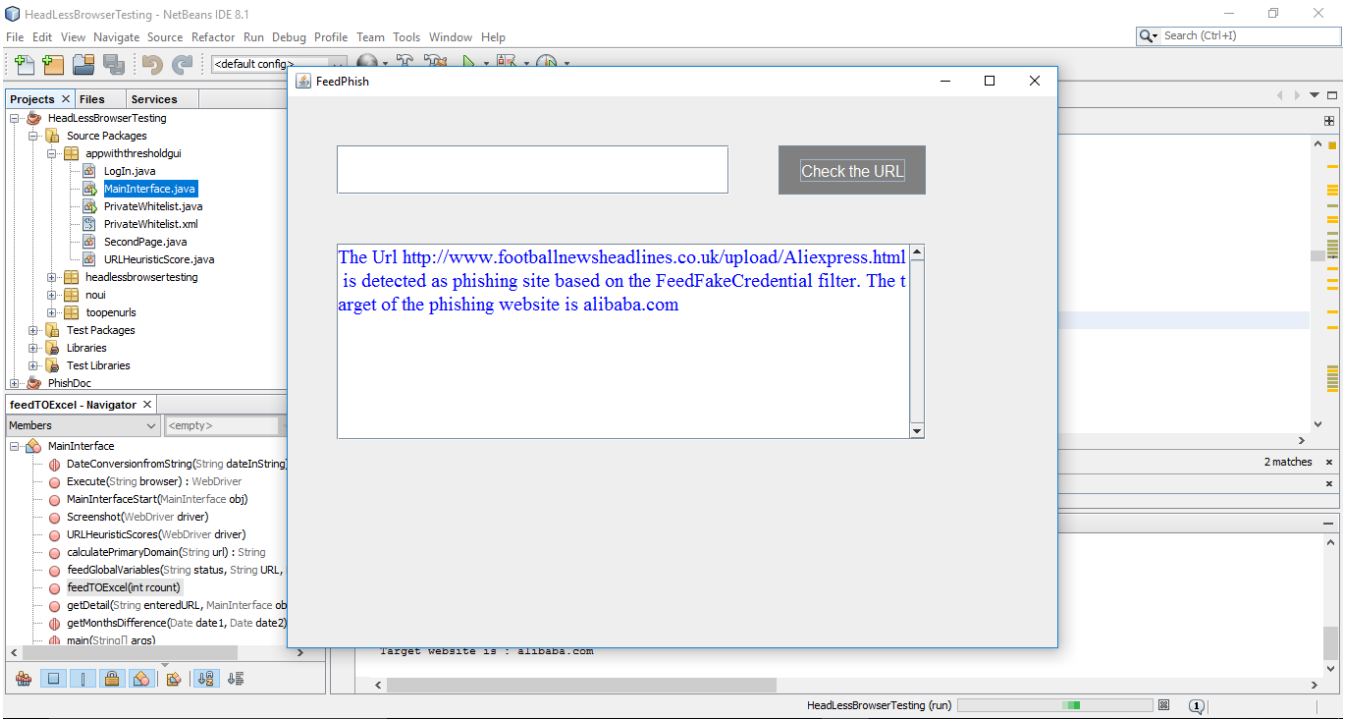

Figure 10: Output interface of our FeedPhish application

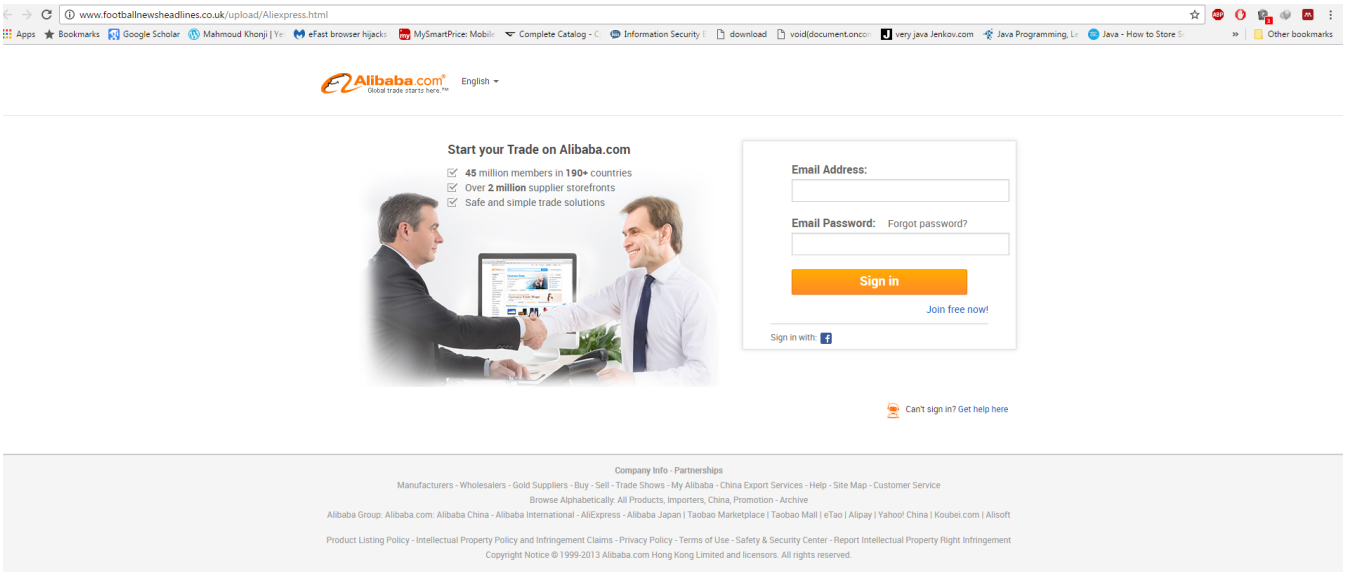

Figure 11: Phishing site targeting Alibaba website