DESIGN AND IMPLEMENTATION OF ENFORCEABLE SECURITY POLICIES FOR BROWSER PROTECTION

Kudirat Oyewumi JIMOH [1]*, Ibrahim Kazeem OGUNDOYIN [1], Lawrence Olaleye OMOTOSHO [1] and Abiodun Gabriel AJAYI [1]
[1]Department of Information and Communication Technology, Osun State University, Nigeria
[1*]Kudirat.jimoh@uniosun.edu.ng, [2]ibraheem.ogundoyin@uniosun.edu.ng,
[3]Lawrence.omotosho@uniosun.edu.ng, [4]abiodun.ajayi@cset.uniosun.edu.ng

Keywords: Bhound, Detection, Malicious, Uniform Resource Locator, Web browser, JavaScript.

Abstract: *In this research, an enforceable security policy model which can prevent the browser from visiting malicious URL was designed, simulated, and evaluated. This is with a view to enhancing security performance of browsers. The main aim of this study is to develop enforceable security policies for browsers and specific objectives are to design a malicious detection model, implement the model and evaluate the performance of the model. The model was designed using the Petri-net model and also, in the study, browser hound (Bhound) was designed and simulated using the string-matching algorithm which serves as a control measure to check for URLs similar to the filters stored in the local storage. String matching algorithm was employed for the compromised URL detection model and the model was evaluated using performance metrics; such as accuracy, precision and, recall rate. The results of system simulation were found to have a 91.2% overall detection rate, 90.0% precision, 28.8% false-positive rate, and 86.0% classification rate. The software development model adopted was a scrum, and the implementation was done using JSON, JavaScript, and hypertext mark-up language (HTML). This system developed is capable of enhancing browser security and could as well handle typosquatting which is one of the schemes used by attackers.*

1. INTRODUCTION

A web browser is described as an application for accessing, presenting and navigating web document or information on the internet. The primary purpose of Web browsers is to allow users to locate, retrieve and view content on the Web available in different formats e.g., HTML based Web pages, files, images, videos, and audios. Web browsers are also used as clients to update and maintain Web applications that run on top of the browsers. However, as the amount of sensitivity usage increased, concerns about security fraud and attacks became important. Due to the widespread availability of internet access, it is very easy for attackers to obtain many clients (and even host) connections and addresses, and use them to launch different attacks, both on the networking itself and on other hosts and clients [8]. While surfing the Internet using a known Uniform Resource Locators (URLs), it is easy to visit sites one think are safe but are not due to malicious threat attached to it.

URL is an avenue by which users identify and access resources on the internet through the web browser. [16]. Uniform Resource Locators (URLs) are tools used to locate Web sites and individual Web resources on the internet just as filenames are used to locate files on a local computer. A system that is not protected using appropriate security controls can be infected with malicious software which can allow anyone to breach into other network devices and access information. Malicious URLs and infected websites are the roots of such threats. Any malicious URL can be said to be a compromised one that usually result into downloading of website threats and compromised accounts. URL is an identifier that uniquely allows resources and also as a global locator of documents and other internet resources web [20]. A URL typically contains two parts protocol name and the domain name or IP Malicious URLs account to one-third of total URLs that exist. Naive users of the

internet are the primary targets of such sites. Most frequent attacks tend to be drive-by download, phishing, and social engineering, and spam as these pose threat to the security of cyberspace. Drive-by download is a harmful computer program that is installed on a user's system without the user's consent or permission. Phishing and social engineering attacks use compromised websites to seek personal information by acting as a trustworthy organization. Spam can be a dangerous source of threat and is a part of a phishing scam. Other ways of compromising websites include the installation of malware programs that typically connect to a command and control (C&C) infrastructure [31]. In this fashion, the infected hosts form a botnet, which is a group of internet-connected devices infected by malware that are under the direct control of cybercriminals. The information on the web is compromised as malicious content by the cybercriminals as a means to distributing malicious computer programs. In recent times, drive-by-download are mostly used to compromise the identity of a large number of users [17].

This technique involves an attacker who developed malicious scripting programs using client-side technologies usually JavaScript for a certain vulnerability in a web browser or browser's plug-in. This scripting program is injected into the compromised web page or otherwise hosted on the webserver controlled by the hackers. When a malicious web page is visited, the scripting program is executed and the user's computing device is infected with malware. The drive-by-download methods have been extensively used as a means to compromising web pages. Among many other ways of compromising websites is an entrance through the vulnerability of the content management software (CMS) such as Blogger, Joomla, WordPress, or custom software; it can also be through the disposal of administrator credentials to such hackers or attackers [4].

The compromised websites are used for hosting malware, URL redirects, hosting phishing, spam pages, and pornography while some of the compromised websites are targeted at vandalism. This vandalism is carried out by competitors of the site owner with the motive of embarrassing the site owner or for hacktivism. The ones targeted at vandalism are done in such a way that the activity can be easily detected by visitors and the site owner while others are not easily detected because of the motive behind the operation (using legitimate websites to carry out illegitimate activities). To eliminate these criminal acts, many approaches to the solution of these problems have been documented. In this study, an automatic detection of malicious URL model was proposed and the model developed was capable of detecting the malicious URLs by building a browser extension and loading blacklisted and whitelisted sites on it.

The rest of this paper is arranged as follows. Related work described in Section 2. Section 3 describes the materials and methods used to develop the work. Result and discussion were discussed in Section 4. Section 5 concludes the paper.

2. RELATED WORKS

Many approaches to the detection of malicious URLs have been proposed and documented. These approaches are grouped into four which include Blacklists, content-based classification, URL-based classification, and feature engineering approach [13]. The blacklist method uses the details of malicious free URLs to filter the incoming URL. The content-based classification identifies the malicious URL using the layout of the page. The URL-based uses the background information of the URL [22]. The feature engineering approach extracts the features for a specific algorithm [13]. [25][26] employed machine learning techniques to detect phishing websites. The features of the phishing website were blacklisted and the database was generated. The study adopted a rule-based method to classify the URL based on some of the features such as IP address, domain identity, security, and encryption, etc. [21] adopt URL ranking techniques to detect URL phishing. The classification of the URLs was done by categorizing their lexical with online on real-time bases. [29] did not construct a classifier but offered an orderly creation of Malicious URL detection from a machine learning perspective, and then detailed the discussions of existing studies for malicious URL detection, particularly in the forms of developing new feature representations, and designing new learning algorithms for resolving the malicious URL

detection tasks.

[2] worked on the review of malicious web page detection considering classification techniques. The study provides knowledge on malicious website, including its vulnerability. Various feature extraction techniques were highlighted and discussed. In a related work of [10], state of the art survey of the malicious web page was considered. The study grouped the detection techniques into three categories namely; static, dynamic and hybrid approaches. The problem relating to each approach was discussed, and also limitation of various methods were analyzed. In the work of [11] numerous new arrangements were allowed to recognize phishing websites. These arrangements were created based on the URL of the webpage. The accuracy of this method may be increased by using some more features. [1] considered features as a means for classification of phishing website. The work employed Adaptive Neuro-Fuzzy Inference System combined with support vector machine for the detection and protection process respectively.

In the work of [12], the efficiency of machine learning approach was evaluated. The work employed bio-inspired algorithm for feature analysis and use particle swarm optimization for optimizing the URL features. The study evaluated its performance by comparing with other machine learning and concluded that Naïve Bayes and support vector machine have better performance. A phishing-list of popular words that are highly indicative of phishing attacks was also constructed and used. The experimental results reveal that features based on word segmentation, phishing-list, and numerical features (e.g., URL length) perform better than all other features, as measured by misclassification and false-negative rates.

[19] proposed the detection of malicious URLs such as spam, phishing, malware and defacement URLs using different lexical features. A precision value of 92.6% was obtained for multiclass classification of phishing URLs legitimate URLs only. [30] proposed URLDeep, a deep architecture by CNN. The obtained results show a better classification accuracy in using lexical features to analyze the URL. [5] mainly focused on the elimination of blacklist by using Random Forest algorithm for the classification of malicious and benign URL. By eliminating blacklist, database containing existing malicious URL may be generated.

From the above-reviewed works, it was found that greater number of the work had focused and limited to signature-based URL detection systems. The limitation of this technique lies in its inability to handle issue of typo-squatting. In this study, Browser hound (Bhound) was developed using the string-matching algorithm capable of handling typo-squatting efficiently.

3. MATERIALS AND METHODS

This section discussed various materials and methods employed for the implementation of the design of the proposed compromised URLs detection model.

3.1 PETRI NETS MODEL DESIGN

The Petri Nets (PN) was used to design the model operations of the system. The model as shown in Figure 1 represent a Condition or Events nets. The place P_1(token) which is an input to the model are placed on the condition (entered URL) which can either be malicious or benign. The transition state t_1 represents the URL matching process which confirms the status of the URL. The places P_2 and P_3 represent classified URL (malicious or Benign) the activities modelled by the two transitions t_2 and t_3 run concurrently. The transition t_2 represents the event of load condition and t_3 represents the failed condition. Place P_4 loads the condition. A

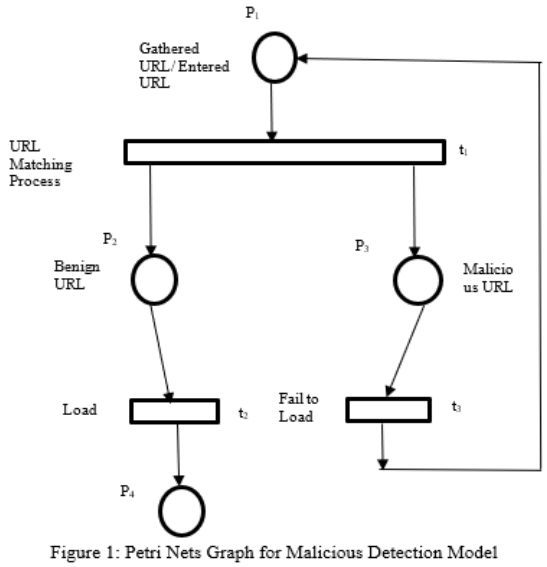


Figure 1: Petri Nets Graph for Malicious Detection Model

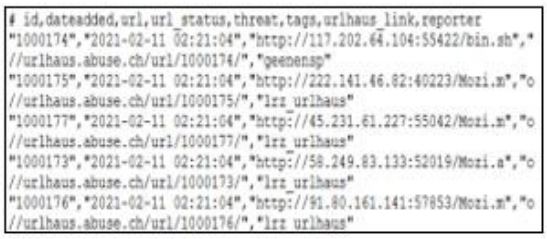
Marked PN is a quintuple (P, T, I, O, M) where:

$P = \{P_1, P_2, P_3 \ldots\ldots P_{np}\}$ is the set of n_p places (drawn as a circle in the graphical representation) $T = \{t_1, t_2, t_3 \ldots\ldots\ldots t_{nt}\}$ is the set of n_t transition (drawn as bars).

I is the transition input relation and is represented using arcs directed from places to transitions.

O is the transition output relation and is represented using arcs directed from transition to places.

$M = \{m_1, m_2, m_3 \ldots\ldots\ldots m_{np}\}$ is the markings.



```
# id,dateadded,url,url_status,threat,tags,urlhaus_link,reporter
"1000174","2021-02-11 02:21:04","http://117.202.64.104:55422/bin.sh","
//urlhaus.abuse.ch/url/1000174/","geenensp"
"1000175","2021-02-11 02:21:04","http://222.141.46.82:40223/Mozi.m","o
//urlhaus.abuse.ch/url/1000175/","lrrz_urlhaus"
"1000177","2021-02-11 02:21:04","http://45.231.61.227:55042/Mozi.m","o
//urlhaus.abuse.ch/url/1000177/","lrrz_urlhaus"
"1000173","2021-02-11 02:21:04","http://58.249.83.133:52019/Mozi.a","o
//urlhaus.abuse.ch/url/1000173/","lrrz_urlhaus"
"1000176","2021-02-11 02:21:04","http://91.80.161.141:57053/Mozi.m","o
//urlhaus.abuse.ch/url/1000176/","lrrz_urlhaus"
```

Figure 2. URLhaus Malicious URL Samples

The generic entry m_i is the number token.

The Input/output function is given in Equations 1-6. Equation 1 is the list of all places in the model while Equation 2 lists all transition state. In Equation 3, transition t_1 has P_1 as input and output to transition t_1 has P_2P_3. The input to transition t_2 is P_2 and transition t_2 has P_4 as shown in Equation 4. In Equation 5, input to transition t_3 is P_3 and its output is P_1. Equation 6 indicates the available elements in the model.

```
https://158.101.14.207/home/confirm.php?cmd=login_submit&
id=8ec11862eb77fb197db8af7daf8a5f5d8ec11862eb77fb197db8af7daf8a5f5d&
session=8ec11862eb77fb197db8af7daf8a5f5d8ec11862eb77fb197db8af7daf8a5f5d
https://158.101.14.207/home/info.php?cmd=login_submit&
id=f76e8cee65c0ae1f8887e41d912d00a9f76e8cee65c0ae1f8887e41d912d00a9&
session=f76e8cee65c0ae1f8887e41d912d00a9f76e8cee65c0ae1f8887e41d912d00a9
https://158.101.14.207/home/login.php?cmd=login_submit&
id=aedc72db84f5dcf20851b9647906c912aedc72db84f5dcf20851b9647906c912&
session=aedc72db84f5dcf20851b9647906c912aedc72db84f5dcf20851b9647906c912
https://158.101.14.207/home/confirm.php?cmd=login_submit&
```

Figure 3. Openphish Phishing URL Samples

$$P = \{P_1, P_2, P_3, P_4\} \quad (1)$$

$$T = \{t_1, t_2, t_3\} \quad (2)$$

$$I(t_1) = \{P_1\} \quad O(t_1) = \{P_2P_3\} \quad (3)$$

$$I(t_2) = \{P_2\} \quad O(t_2) = \{P_4\} \quad (4)$$

$$I(t_3) = \{P_3\} \quad O(t_3) = \{P_1\} \quad (5)$$

$$M_1 = (1, 0, 0, 0) \quad (6)$$

To evaluate the production of the reachability set of the PN of Figure 1 given the initial marking $M_1 = (1, 0, 0, 0)$. In M_1 the only enabled transition is t_1, firing t_1 removes the token from P_1 and put a token on both P_2P_3 producing the new marking $M_2 = (0, 1, 1, 0)$. In M_2, the transition t_2 and t_3 are both and can fire concurrently. Firing of t_3 leads to $M_3 = (1,1,0,0)$ and subsequent firing of t_2 leads to $M_4 = (1,0,0,1)$. With this, all possible firing has been examined and the reachability set $R(M_1)$ of Figure 1 turns out to contain 4 elements M_1, M_2, M_3, M_4.

3.2 DATA COLLECTION

Three classes of data were collected: URLhaus, OpenPhish, and Moz Trusted URL respectively. In total, three thousand (3000) data were collected. One thousand (1000) datasets were collected from the Urlhahus Malicious URL database which includes the date and the status of the URL. These datasets were loaded in the blacklist filter for further processing. Also, one thousand (1000) datasets with URL names were collected from OpenPhish, which are used for zero-day phishing sites to determine a real-time threat. The dataset was loaded in the blacklist filter for further processing. Likewise, for Moz Trusted URL, one thousand (1000) datasets with trusted sites were also collected and these datasets were loaded in the whitelist filter for further processing. Samples of the URLhaus Malicious URL and Openphish Phishing URL are shown in Figures 2 and 3 respectively.

3.3 DESIGN PROCESS AND BROWSER INTEGRATION

A user needs to enable the Chrome extension to use Browser Hound. If the extension is enabled, it will get the URL and call the detector. The detector is hosted on the Chrome extension store within a directory, which contains all necessary files used to run an extension, such as manifest.json, popup.html, JavaScript file, and so on. Also, the detector in the background processes the URLs, classifies the file as malicious or benign, and returns the response to the Chrome extension. However, if the result is malicious, the user will be notified. Manifest.json file contains all the details required for the extension like name, description, permission required, browser actions, and version, and other necessary details. Once the

developer mode is turned on, it will enable an option to load the extension files. Figure 5 shows that the extension package was fully loaded and integrated. The toggle button can be selected to initialize or stop the process, and use the "details" option for the extension details, such as version number, size, and author.

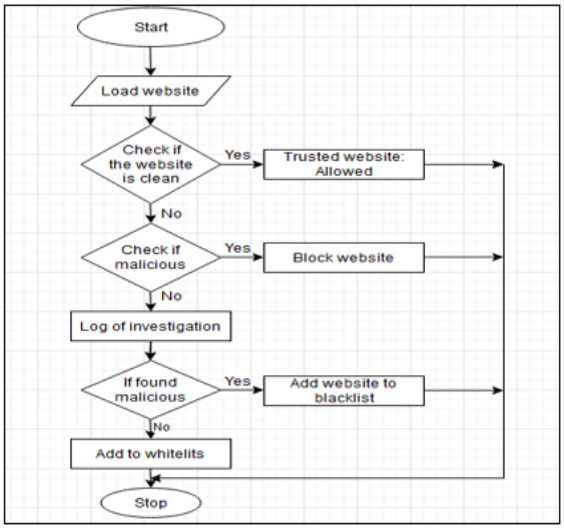


Fig. 4. Flowchart diagram for the proposed design

3.4 MALICIOUS BHOUND DETECTION

The malicious Browser hound detection model is proposed for the detection of malicious URLs in the cyber environment. The model employed Sublime text, JavaScript Object Notation (JSON), Hypertext Mark-up Language (HTML), and Text file. JSON was used for a data interchange format for configuring Bhound, JavaScript, and HTML were used as client-side technologies while Text file was employed as electronic text to store URL datasets with the computer file system.

As shown in Figure 4, the URL of a typical website is an input to the developed model. The model tries to check the status of the URL after comparing it with the template to know if it is clean and if found benign, it will be allowed to launch and added to the Whitelist. But if the URL status is malicious, it will be blocked and added to the Blacklist to update the database. Also, if the status is unknown, it will be subjected to further investigation and after this process, if found benign, it will be added to the Whitelist to update the database. The whitelist filter stores non-malicious uniform resource locators (URLs) on local storage. This feature also allows non-malicious URLs to be manually added to the local storage. Also, the blacklist filter content can be manually updated with malicious URLs. It also allows manual editing of a particular filter in the local storage including removal of a selected filter from the local storage manually. Importing filters from text file helps to input multiple data (filters) from a text file into the database and export filters as text file helps in extracting data (filters) from the local storage. The filters are extracted as text files, which can be used to model another system. Filters can be sorted and the system can be set to select a default action to take for sites in the filters. A blocking time can be scheduled with enabling or disable blocking mode which serves as a switch that enables the system to work on the browser. Password protection can be enabled with automatic enabling on the browser startup.

4. RESULTS AND ANALYSIS

The model was simulated and processed on 8GB of RAM, a 2.3GHz processor machine using the model developed in this study. For the purpose of the implementation of this model, Sublime text, JavaScript Object Notation (JSON), Hypertext Mark-up Language (HTML), and Text file were used as implementation tools. In this study, 70% of the samples of data collected were used for train and 30% were used for testing. The trained datasets were processed and integrated into the Bhound on the web browser. Figure 5 shows the page where Bhound was integrated into the web browser. Figures 6 and 7 show a page when Bhound detection model restricts a URL from launching and allowing a trusted URL to launch respectively. The performance evaluation of the model was done using precision rate, recall rate, and accuracy which are defined in Equations 7, 8, and 9. Table 1 shows the confusion matrix table for the overall detection process. The accuracy of the model was recorded for each malicious detected and the overall detection rate for the selected samples was retrieved using the confusion matrix. The analysis of the result obtained was represented using bar chart as shown in Figure 8.

$$Precision = \frac{T_P}{T_P + F_P} \tag{7}$$

$$Recall = \frac{T_P}{T_P + F_N} \tag{8}$$

$$Accuracy = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \quad (9)$$

i. True positive (T_P): Intrusions that are successfully detected by the IDS. This refers to URLs which are malicious and actually predicted malicious i.e. 160. Examples: www.secure.login.aliexpress.com.coin-balance.com, www.stdyrusschinetwomast.dns.army, www.new-payee-confirm-security.com.

ii. False positive (F_P): Normal/non-intrusive behaviour that is wrongly detected as intrusive by the IDS. URLs which are not malicious and actually predicted as malicious, i.e. 40. Examples: www.sriglobalit .com, www.sunbayhotel.vn

iii. True Negative (T_N): Normal/non-intrusive behaviour that is successfully labelled as normal/non-intrusive by the IDS. URLs which are not malicious and actually predicted not malicious, i.e. 62. Examples: www.google.com, www.youtube.com, www.yahoo.com

iv. False Negative (F_N): Intrusions that are missed by the IDS, and detected as normal / non-intrusive. URLs which are malicious and actually predicted as not malicious i.e. 38. Examples: www.newpayees-update.com, ww.activacionesdeproductos.com

Table 1. Confusion Matrix

Actual	**Predicted (200)**	**Predicted**	
Attack		Attack 160TP	Normal 38FN
Normal		40FP	62TN

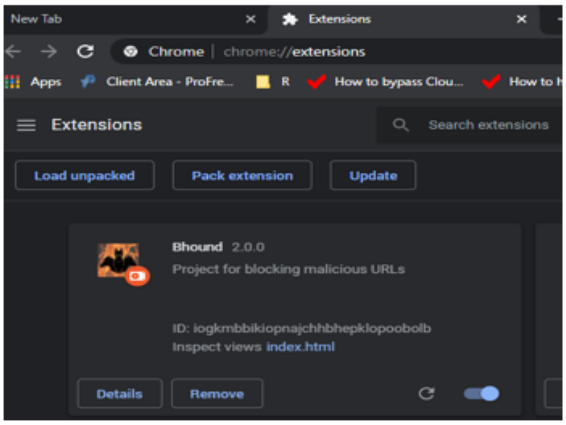


Figure 5. Integrating Bhound into the Browser

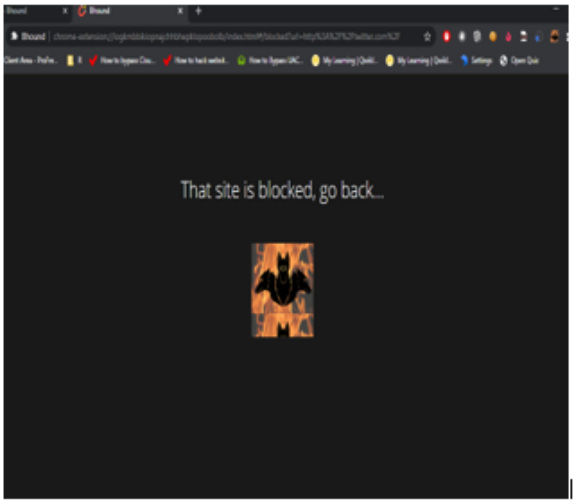


Figure 6. Bhound blocking a Malicious URL

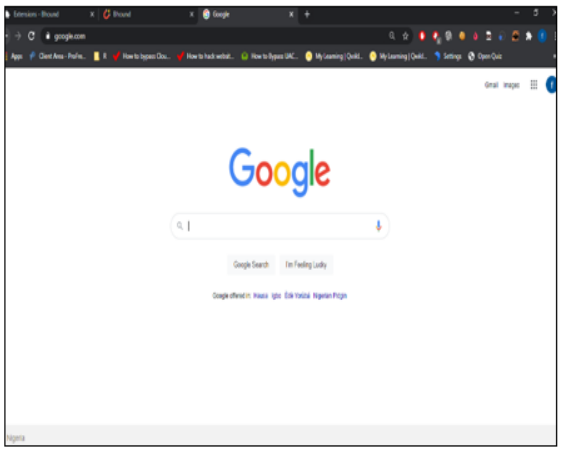


Figure 7. Bhound allowing a trusted URL

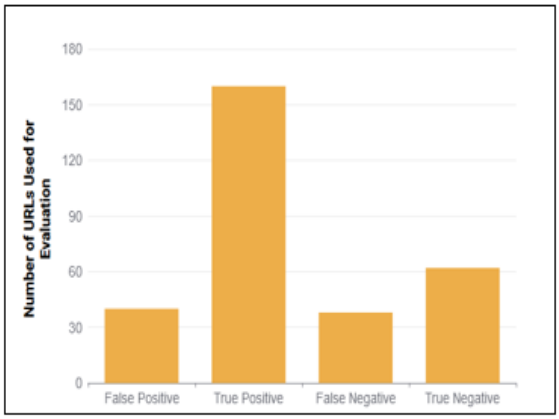


Figure 8. Result analysis of Malicious URL Detection Model

5. CONCLUSION

In this research study, an enforceable security policy model that the browser must follow in its operation in form of malicious URL detection was developed. It uses a string matching algorithm which stands as a measure to check for URLs similar to the filters stored in the local storage. It is enforceable because the model carries out some background mandatory investigations and authorization before executing any instructions from the user. It also a way of making web application and browser more intelligent in averting cyber threat such as SQL command injection and cross-site scripting which could wreck havoc on computer system and information stored on it. This system developed was also capable of handling the issue of typo-squatting efficiently, which is one of the schemes used by attackers. The model employed Sublime text, JavaScript Object Notation (JSON), Hypertext Mark-up Language (HTML), and Text file. JSON was used for a data interchange format for configuring Bhound, JavaScript, and HTML were used as client-side technologies while Text file was employed as electronic text to store URL datasets with the computer file system. The performance evaluation result obtained showed that the model is capable of detecting malicious threats in the web URL. The main benefit of the proposed model is its ability to update the blacklist and whitelist contents, thereby enhancing the efficiency output of the model.

6. REFERENCES

[1] Adebowale, M, Lwin, K, Sanchez, E and Hossain, A.. "Intelligent Web-Phishing Detection an Protection Scheme using
integrated Features of Images, Frames and Text. *Expert" Systems with Applications*. 115(2019), 300-313, 2018.

[2] Agrawal Jitendra, Agrawal Shikha, Awathe Anurag, Sharma Sanjeev, "Malicious Web Page Detection through Classification Technique: A Survey", International journal of Computer Science an Technology, 8(1), 74-79, 2017

[3] Atharva Deshpande, Omkar Pedamkar, Nachiket Chaudhary, Swapna Borde "Detection of Phishing Websites using Machine Learning", International Journal of Engineering Research & Technology. 10(5), 2527-2531, 2021.

[4] Commtouch Full compromised web, report retrieved January 2, 2021, https://www.slideshare.net/CYREN/commtouchhacked-website-report 2012.

[5] Deebanchakarawarthi G, Parthan A. S., Sachin Lal, Surya A. **"**Classification of URL into Malicious or Benign using Machine Learning Approach", *International Journal of Advanced Research in Computer and Communication Engineering*. 8 (2), 245-248, 2019.

[6] Eric A. Fischer. "Cybersecurity Issues and Challenges: In Brief" *Congressional Research Service* pp. 4-6, 2016.

[7] Garera, N. Provos, M. Chew, and A. D. Rubin . A Framework for Detection and Measurement of Phishing Attacks. *InWORM*. ACM, Alenxandria. 2007

[8] Gopinath P., Sangeetha S., Balaji R., Sanjay, S. G., Bindhumadhava B. S. Malicious Domain Detection Using Machine Learning on Domain Name Features, Host-Based Features and Web-Based Features. *Procedia Computer Science* 171 (2020) 654–661, 2020.

[9] Gunikhan, S., and Kuppusamy, K S. Machine Learning Based Accessible Anti-Phishing Model with Heterogeneous Features and Multifarious Filters. PhD Dissertation, 2019.

[10] Jayakanthan N. and Ramani A.' Malicious Web Page Detection: A state of art Survey. *International Journal of Science Technology and Management*. 5(3), 359-371, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. IEEE *Conference proceedings on*

Computer Vision and Pattern Recognition. Vol.1 pp. 770-778, 2016.

[12] Lee Ong Vienna, Heryanto Ahmad, Ab Razak Mohd Faizal, Anis Farihan Mat Raffei, Danakorn, Shahreen Kasim,]Tole Sutikno. A malicious URLs detection system using optimization and machine learning classifiers. *Indonesian Journal of Electrical Engineering and Computer Science* 7(3), 1210-1214. 2020.

[13]Li, Tie Gang Kou, Yi Peng "Improving malicious URLs detection via feature engineering: Linear and non-linear Space transformation methods", Information Systems, 91(2020), 101494,2020.

[14]Markopoulou, Le,A.and Faloutsos, M. Technical Report: "Phish Def:URL Names Say It All". http://www.ics.uci.edu/~anhml/publications.html. Also onarxiv:1009.2275, Sep. Moz Database Dump. https://moz.com/top500.

[15] Ma, L. K. Saul, S. Savage, and G. M. "Beyond blacklists: learning to detect malicious Web sites from suspicious URLs" Conference: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 2009

[16]Ma, L. K. Saul, S. Savage, and G. M. Voelker. "Identifying Suspicious URLs: An Application of Large-Scale Online Learning". In Proceeding of *International Conference on Machine Learning*, Califonia, 2017.

[17] Marco, Canali, Giovanni Cova, Kruegel, V. C. "Prophiler: A Fast Filter for the Large-scale Detection of Malicious Web page" International World Wide Web Conference Committee (IW3C2) 2011.

[18] Masanori Hara, Akira Yamada, and Yutaka Miyake, "Visual similarity-based phishing detection without victim site information". In Computational Intelligence in *Cyber Security*.CICS'09. IEEE Symposium on. IEEE, 30–36, 2009.

[19] Mamun, M., Rathore, M., Habibi L., Stakhanova, N., and Ghorbani, A. "Detecting Malicious URLs Using Lexical Analysis" 9955. 467-482. 10.1007/978-3-319-46298-1_30, 2016.

[20] Michael Cross. "Social Media Security", First Edition. Elsevier.

[21] Mohammed, N. and Susan, M. Phishing URL Detection Using URL Ranking. 2015 International Congress on Big Data. Pp. 635-638.

[22] Moshchuk, A. Bragin, T. Deville, D. Gribble, S. D. Levy, H.M. SpyProxy: Execution-based detection of malicious web content, in: Proc. of the USENIX Security Symposium, Boston, MA2007.

[23] Rakesh Verma, Devin Crane, and Omprakash Gnawali. "Phishing During and After Disaster: Hurricane Harvey.In *Resilience Week* (RWS). IEEE, 88–94.

[24]RLhaus Database Dump. Retrieved from https://urlhaus.abuse.ch/downloads/csv_online on 18th April, 2021.

[25] Rudra Nath, Katja Hose, Torben, 2018. Pedersen, "Towards a programmable semantic extract transform-load framework for semantic data warehouses", *Information System*. 68 pp. 17–43, 2017.

[26] Sudha M., Jaanavi R. V., Blessy Ida Gladys S., Priyadharshini, "A Review on Phishing Website Detection sing Machine Learning", Journal of Critical Review, 7(19), 4847-4853, 2020.

[27] Tayyab Saad and Masood Asad "A Review: Phishing Detection using URLs and Hyperlinks Information by Machine Learning Approach", *International Journal of Computer Science and Mobile Computing*, 8(3), 345-351, 2019.

[28] URLhaus Database Dump Retrieved from. https://openphish.com/feed.txt 18th April, 2021

[29] Sahoo, Doyen, Liu, Chenghao and Hoi, Steven. "Malicious URL Detection using Machine Learning: A Survey", ACM, 1(1): 1-37, 2019'

[30] Wanda, Putra & Jie, Huang. "URLDeep: Continuous Prediction of Malicious URL with Dynamic deep learning in Social work, *International Journal of Network Security*. 21(6), 971-978, 2019.

[31] Zhauniarovich ,Yury, Khalil Issa, Yu Ting, and Dacier ,Marc. "A Survey on Malicious Domains Detection through DNS Data Analysis", *ACM Computing Survey*. 51 (67), 1-36, 2018.