2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE) | 978-1-7281-9111-9/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCEEE49695.2021.9429609

Defences Against web Application Attacks and Detecting Phishing Links Using Machine Learning

Aya Hashim*, Razan Medani*, Dr.Tahani Abdalla Attia*
*University of Khartoum, Faculty of Engineering
Department of Electrical and Electronic Engineering
Algamaa Street, P.O. Box 321, Khartoum, Sudan
{ayahashim16, razanmedani}@gmail.com

***Abstract*—In recent years web applications that are hacked every day estimated to be 30 000, and in most cases, web developers or website owners do not even have enough knowledge about what is happening on their sites. Web hackers can use many attacks to gain entry or compromise legitimate web applications, they can also deceive people by using phishing sites to collect their sensitive and private information. In response to this, the need is raised to take proper measures to understand the risks and be aware of the vulnerabilities that may affect the website and hence the normal business flow. In the scope of this study, mitigations against the most common web application attacks are set, and the web administrator is provided with ways to detect phishing links which is a social engineering attack, the study also demonstrates the generation of web application logs that simplifies the process of analyzing the actions of abnormal users to show when behavior is out of bounds, out of scope, or against the rules. The methods of mitigation are accomplished by secure coding techniques and the methods for phishing link detection are performed by various machine learning algorithms and deep learning techniques. The developed application has been tested and evaluated against various attack scenarios, the outcomes obtained from the test process showed that the website had successfully mitigated these dangerous web application attacks, and for the detection of phishing links part, a comparison is made between different algorithms to find the best one, and the outcome of the best model gave 98% accuracy.**

***Index Terms*—Cyber security ,PHP and MySQL , Apache access Logs, Server , Web application , Attacks, Phishing ,Machine learning, Classification algorithms, Support vector machine ,Random Forest, Logistic regression,Long short term memory networks.**

I. Preface

The internet is growing very fast and everyone is using it because it can afford all needs online as well as providing a wide range of services. As the use of the internet rapidly increase the requirement of providing confidential and critical data transmissions raises. This has revealed it to a wide range of security attacks, Therefore, it is important to implement web applications security techniques while developing an online web application. Besides these attacks, phishing is one of the most common social engineering attacks that is used by attackers to steal critical and sensitive information by disguising themselves as trustworthy organizations, phishing has compromised millions of users' data. Log files are files that keep a registry of events and activities, therefore they are used to save all the logs of the website in order to keep track of every user attempts to log in. The paper is structured as follows: section II discusses problem description, section III describes common Web Application Attacks, and their mitigations are discussed in section IV. Section V illustrates access logs, and in section VI phishing links detection is discussed using machine learning in section VII and deep learning in section VIII. The results are discussed in section IX. Finally, the conclusion and future work are provided in section X.

II. Problem Description

Nowadays making any online web application without securing it properly makes it vulnerable to cyber-attacks. Hackers all over the world can attack your website and gain secret information or make serious damage to your website. Some attacks, for example, when a user opens a link contains a malicious JavaScript code, it might steal personal information or hijack a web session, and many different hacking scenarios can happen. Hackers will also often insert phishing links into a website, detecting phishing links manually by searching every page in the website is inefficient, time-consuming and requires security experts.

III. Related Work

A. Web Application Attacks

The mitigation techniques used for SQL injection attacks were explained in this study [1] The prepared statement and the real escape function must be used to avoid passing the input directly into SQL queries. This study [2] investigated Cross Site Scripting Attacks and propose an efficient approach to prevent this attack by applying sanitization methods. This study [3] explained Local File Inclusion attack and proposed a solution which is sanitizing input data by checking the directory traversal (../) or its hexadecimal encoding. It also explained Cross Site Request Forgery (CSRF) and proposed a solution by generating random tokens submitted with every request and MD5 hashing algorithm was used. It also explained PHP code injection and proposed a solution which is to sanitize input string for any eval() (PHP function) injections before giving the input as an argument to eval function. This study [4] shows that Sanitization of Input method in order to mitigate RFI command injection respectively, is the most promising method as it allows only the non-vulnerable code to be executed ignoring all the vulnerabilities of it.

978-1-7281-9111-9/20/$31.00 ©2020 IEEE

B. Phishing Detection

A study was carried out to estimate the performance of the currently used tools for phishing links detection. It exposed that even the most authentic phishing detection tool-bars dropped over 20% of the phishing sites. Another heuristics-based research tests typical phishing site characteristics, such as specific keywords used in URLs. One drawback of these forms of heuristics is that once their technique is detected, attackers can easily bypass it. [5]. Recently the classification of URLs using machine learning techniques has been of great concern, with many latest studies suggesting the use of classification algorithms to detect phishing URLs. These studies are focused on creating features relying on expert experience and lexical analysis of the URL(such as length of host-name and URL length). Then, the extracted features are fed to the model. The model learns to identify patterns and correlations that the inputs must pursue to classify a site as legitimate or phishing [6].

IV. Web Application Attacks

Web applications Attacks become more popular because many organizations and companies move their services online. From attackers perspective, web applications become an effortless victim, that is why web application attacks are one of the biggest threat in the cybersecurity field. These attacks can drive disturbing results such as sensitive information stealing and deleting or editing a database. The most common web application attacks are:

- Structured Query Language (SQL) Injection is a vulnerability that enables the execution of malicious SQL statements. SQL Injection vulnerabilities can be used by attackers to manage the database. [7].
- Cross Site Scripting (XSS) is a class of web application vulnerability occurs when an attacker succeeds to deliver a malicious JavaScript payload to the user. [8].
- Command Injection allows the attacker to execute unwanted system commands and gain unauthorized access to the user's information [9].
- PHP Code Injection is a vulnerability that allows the injection of a code into the server-side. With exploiting this bug, an attacker can insert shell command such as 'shell exec(dir)' -to know entire directory structure -into a special PHP function called 'eval()'. Eval function will execute the shell command. [10].
- Local File Inclusion(LFI) is a vulnerability that occurs when a code includes files from the web server, an attacker will be able to exploit this vulnerability by manipulating the file location parameter and includes a system file. [11].
- Remote File Inclusion (RFI) is a vulnerability that lets the attacker executes a malicious code on the target-machine even though it is not hosted on that machine [12].
- Cross Site Request Forgery (CSRF) Is the process of impersonating a legitimate user, for example, a malicious web page fools the browser of a logged-in user by sending a request to a target website. In this case, the browser will validate the malicious request as if it is assigned by the user [13].
- Brute Force Attack depends on guessing possible combinations of a targeted password until the correct password is revealed [14].
- Enumeration Attack when the web application leaks information about the existence of username or password in the error message, an attacker can use brute-force techniques to either guess or confirm valid users in a system [15].

V. Web Application Attacks Mitigation

The attacks mentioned earlier are mitigated as follows:

- SQL Injection - to mitigate web application from this attack, three mitigation techniques are implemented first is the input Validation in which Regular expression (regex) from open web application security project (OWASP) Core Rule set for this attack is used to detect if the input is an attack or not, the second mitigation is prepared Statements, it allows separation of what is the actual query (code) and what is the data that is being passed to it as statements, the third way to mitigate injections is to add a backslash before any dangerous characters(i.e., Backslash, apostrophe, and semicolon). In PHP, this done by using the mysql-real-escape-string function.
- Cross Site Scripting (XSS) - three techniques are used to mitigate XSS, first is the input validation, second is Escaping any character that can influence the formation of the document for most reliable outcomes the built-in htmlentities() function that PHP offers is used and third is setting HTTPOnly flag to true in the response header so that the cookie cannot be accessed through the client side script.
- Command Injection - To avoid Command injection attack the proposed website will provide input validation by comparing user input with command injection regular expression provided by OWASP Core Rule Set.
- PHP Code Injection- To avoid PHP code injection attack the proposed website provides input validation by comparing user input with PHP code injection regex provided by OWASP Core Rule Set.
- Local File Inclusion(LFI) - the proposed website provides input validation by comparing 'page' value (URL parameters) in URL with LFI regex provided from OWASP Core Rule Set, another mitigation technique is implemented by giving the open-basedir in the configuration file (php.ini) which is associated with windows, apache, MySQL, and PHP (WAMP) server the path to the web application files so the user access is restricted only to the files in the specified path.
- Remote File Inclusion (RFI) - the proposed website provides input validation by comparing 'page' value in URL with RFI regex provided from OWASP Core Rule Set.

- Cross Site Request Forgery (CSRF) - Mitigating CSRF requires a random token to be generated and validated on every request. One of the common ways to mitigate CSRF while submitting forms is to attach a random token with each request. This is a unique and random combination of letters and numbers that are generated for each session. The token is produced and then attached in every form request as a hidden input. The website then verifies that the form is valid by matching the token value with the one stored in the user's session variable. An attacker will not be able to forge requests without knowing the token values.
- Brute Force Attack - to prevent brute force Attack limiting login attempts is considered. The website gives the users 5 attempts and if they failed the user IP address is blocked for 3 minutes.
- Enumeration Attack - An effective prevention is to have the server responds with a generic message that does not indicate which field is incorrect, therefore an attacker cannot know whether usernames are valid or not.

VI. Access Logs

The Apache access log saves pieces of information regarding events that happened on your Apache webserver. when a website visitor access the website a log is generated and saved for the web administrator to know certain information like internet protocol (IP) address of the visitor, and the time of the request. Apache access log file has many formats, combined log format is used to save logs.

VII. Phishing Links Detection

In this paper phishing URLs are detected using three classification algorithms as well as long short term memory (LSTM) networks and a comparison is proposed between them.

VIII. Phishing Websites Detection using Machine Learning

To detect phishing URLs using machine learning three steps are proposed:

A. Data Collection

The dataset was gathered from MillerSmiles archive, PhishTank archive and Googleâ€™s searching operators. The dataset consists of 11056 instances, 6157 of it is legitimate and 4898 is phishing, also it has 22 features and one label to indicate if URL features are legitimate or phishing. Values when extracting features are -1, 0, and 1. -1 represents phishing, 0 denotes suspicious and 1 denotes legitimate.

B. Features Extraction

The key characteristics that have proven effective in detecting phishing websites are discussed [16], and they are:

- IP Address - If the domain name of the URL contains an IP address, it is then classified as phishing.
- URL length - Long URLs can contain harmful content. If the length of the URL is greater than the URL's average length, it is then classified as suspicious or phishing.
- TinyURL links -are regarded as phishing because it can forward the user into a fraudulent website.
- Having "@" symbol in URL - The browser ignores the part of the URL that comes before the @ symbol so if any URL contains @ symbol it is considered phishing.
- Using "//" symbol - If the URL has a "//" then the user will be forwarded to an external webpage. If "//" is used after Hypertext Transfer Protocol (HTTP) or Hypertext Transfer Protocol Secure (HTTPS), it is considered legitimate.
- Having "-" in domain name - If the '-' symbol is contained within a domain name, then it is considered phishing.
- Dots in domain - Dot is added whenever a sub-domain is included in the domain name. If the URL contains more than one sub-domain, it is considered suspicious, and if it contains greater than two sub-domains it is considered phishing.
- Domain Registration length –It is found that phishing URLs don't live for a long time. Therefore, if the domain expires in one year it is considered phishing.
- Favorite Icon (Favicon) - Favicon is an image or icon correlated with websites. If the favicon is loaded from an external domain, it might redirect the user to a phishing site.
- "HTTPS" on domain - If the domain name contains "HTTPS", it is considered phishing.
- Request URL - If the external objects (images or videos) in a webpage are loaded from external or other domains, it is regarded as legitimate, suspicious or phishing based on the percentage.
- Using <a >tags - This feature tests if the <a >tags and the webpage domain names are different, it is considered legitimate, suspicious or phishing based on the percentage.
- Links in <meta >,<script >and <Link >tag - This feature tests if these tags domain names are different from the webpage domain name, it is regarded as legitimate, suspicious or phishing based on the percentage.
- Server Form Handler (SFH) - If SFHs contain an empty or blank string, it is regarded as phishing. Also, if the submitted form is directed to an external domain (meaning that SFH domain name is different from the webpage domain name), It is regarded as suspicious.
- Submitting Information to Email – Phishers might make use of web form by directing the submitted information to their email rather than the server, it is considered phishing if the information is directed to an email.
- Abnormal URL – A URL is regarded as phishing if the hostname or identity is not part of the URL.
- Initial Frame (IFrame) Redirection- IFrame is used to display or include a web page into the current one. IFrame tag might be inserted secretly in the web page. Therefore, if Iframe tag is used it is considered phishing.
- Age of Domain - Phishing URLs don't live for a long time. Therefore, if the domain age is greater than or equal

to six months, it is considered legitimate, otherwise, it is considered phishing.

- Domain Name System (DNS) record - If a website doesn't have a DNS record, it is considered as a phishing site.
- Website Traffic – The rank of a website depends on traffic and number of visitors. If the website doesn't have traffic it is considered phishing.
- Google Index – if a website isn't in Google index and doesn't appear on search results, it is considered phishing.
- Statistical-Reports Based Feature - If the webpage host refers to Top of phishing IP's or domains, it is considered phishing.

C. Detection using Machine Learning

First, the dataset has been processed to get mature data in the desired format and to extract features, then it is divided into two sections 80% training and 20% testing. The experiment has been carried out using python installed on windows 10. The experiment has two phases:

- Training phase: Training data is fed to three classification algorithms. These include Support Vector Machine (SVM), Random Forest and Logistic regression. In this phase, each model learns how to detect phishing URLs and hyper-parameters in each algorithm are tuned using a grid search to give the best performance.
- Testing phase: Testing data is fed to the same three classification algorithms to assess each model performance (accuracy).

IX. Phishing Websites Detection using LSTM

To detect phishing URLs using deep learning three steps are proposed:

A. Data Collection

Phishing URLs were collected from Phish-Tank, Openphish and others, while legitimate URLs were collected from Alexa top websites. The dataset consists of 119313 instances, 49144 of it is legitimate and 70168 is phishing. It consists of URLs and labels to indicate if a URL is legitimate or phishing.

B. Data Processing

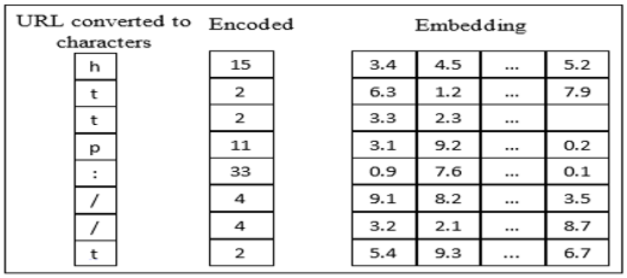
In the previous machine learning section, a series of features were extracted from URL and fed to a classification model to predict whether a URL is phishing or not. But here, Instead of manually extracting the features, three steps are proposed:

- Each URL is tokenized using char-level tokenization method, the total number of characters, numbers and symbols found in training data was 90.
- Each character is encoded by an integer so that each URL is represented by a list of integers.
- In order to give all URLs same length which is 2083 integers, longer URLs are truncated and smaller URLs are padded by '0' integer.

C. Detection using Deep Learning

The dataset has been processed by converting each URL to a list of integer of length 2083, then it is separated into three sections: 50% training, 25% validation, and 25% testing. The experiment has been carried out using python and Keras library. After converting each URL to a list of integer, each list is fed to Embedding layer in Keras In order to know relationships or similarity between these URLs. As in Figure 1, the Embedding layer is initialized with random weights and can learn an embedding vector for all of the characters in the training dataset, thus each character in the URL is represented by 64 embedding vector. Thus each URL is converted to (2083, 64). The sequential patterns of characters are important for this reason RNN model must be used because it is able to model sequential patterns. One disadvantage of general RNNs is that they can not learn the association between elements separated by more than 5 or 10-time steps, in other words, RNNs have short-term memory. A model that solves this problem is LSTM that is why it is used in this work. The experiment has two phases:

- Training phase: Training and validation data is fed to the Embedding layer, three LSTM layers each layer has 32, 64,128 units respectively, fully connected (FC) layer with 20 neurons (units), and final FC layer with one unit to compute the output result. The model is trained using 18 epochs. In the last LSTM, only the last sequence is fed to the first FC layer. Dropout is used before the final layer to drop 20% of all hidden units to reduce overfitting or model complexity. Then the model is created by model.compile(), which contains argument parameters (loss, optimizer, metrics) required for compiling the model. The loss function is required to compute the quantity that a model should seek to minimize during training, and the 'binary crossentropy' is used to calculate the loss, the optimizer tries to minimize or maximize some attributes (i.e. weights and biases) of the program and the class used for optimizer function is 'adam', the metric function judges the performance of the model and the class used is 'accuracy' which calculates how often predictions equal labels.
- Testing phase: Testing data is fed to the same model to assess the model performance(accuracy).



URL converted to characters	Encoded	Embedding			
h	15	3.4	4.5	...	5.2
t	2	6.3	1.2	...	7.9
t	2	3.3	2.3	...	
p	11	3.1	9.2	...	0.2
:	33	0.9	7.6	...	0.1
/	4	9.1	8.2	...	3.5
/	4	3.2	2.1	...	8.7
t	2	5.4	9.3	...	6.7

Fig. 1. Character Conversion to 64-Dimension Embedding .

X. Results

In this work, the Web application is tested against previously mentioned attacks and the mitigation techniques are proved to be effective in securing the website from external breaches. For the detection of phishing links data was collected from UCI (University of California, Irvine) machine learning Repository. Random forest, logistic regression and SVM algorithms have been implemented in the collected dataset, the confusion matrix was applied on tested and predicted data set, and then the results in terms of Accuracy, Recall, Precision and F - measure have been calculated using specific formulas. Confusion matrix calculates the following values :

- True Positive(TP): model predicts positive and it is true.
- True Negative(TN): model predicts negative and it is true.
- False Positive(FP): model predicts positive and it is false.
- False Negative(FN): model predicts negative and it's false.

Once these values are calculated performance measurements can be calculated using the following formulas :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{4}$$

A comparison between three classification algorithms is considered as in Table I, it is shown that SVM has the highest accuracy and F-measure, for this reason, it is regarded as the best algorithm when extracting features manually.

TABLE I
Compares Three Classification Algorithms

Classification Algorithm	Accuracy	Precision	Recall	F-measure
Random forest	0.94103	0.907926	0.95939	0.932949
Logistic Regression	0.88639	0.856685	0.887967	0.872046
SVM	0.94139	0.922338	0.946589	0.934306

Data was collected from Phishtank, Openphish, and others. Those were URLs data, not features, preprocessed using natural language processing techniques, and then fed to the LSTM network. Table II shows an accuracy comparison of SVM and LSTM, and it appears that the LSTM model predicts better, has better accuracy, and a better f-measure. Thus LSTM model is considered the most robust way to detect phishing URLs using a character sequence method.

TABLE II
Comparing SVM and LSTM Accuracy

Algorithm	Accuracy	Precision	Recall	F-measure
SVM	0.94139	0.922338	0.946589	0.934306
LSTM	0.986556	0.989089	0.988135	0.988611

LSTM model learning curve shows that in just 18 epochs, validation accuracy converges training accuracy, increasing to over 98%, as in Figure 2, and Figure 3 shows training and validation loss.

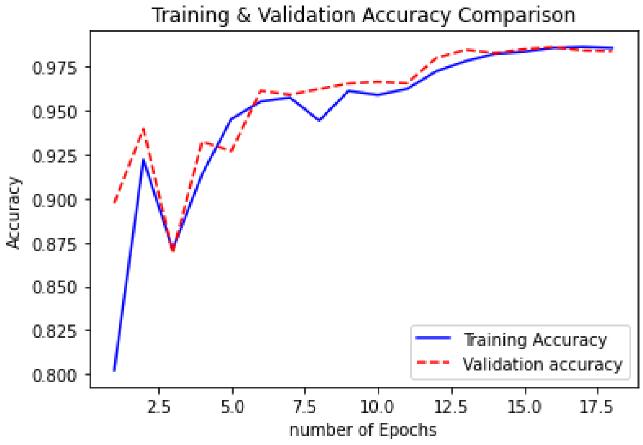


Fig. 2. LSTM Training and Validation Accuracy.

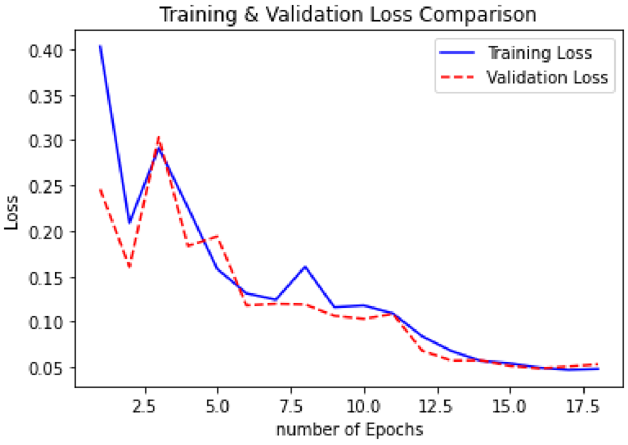


Fig. 3. LSTM Training and Validation Loss.

Also, to be more accurate in comparing SVM and LSTM because they were trained in different datasets, random samples of URLs were taken and trained in LSTM and SVM. The sample numbers are 729, 1458, and 2187. First, each sample's features were extracted manually and fed into the SVM, then URLs from the same sample were tokenized (converted to character sequences), encoded, and fed into the LSTM, as illustrated in Figure 4 the LSTM accuracy is better than the SVM.

In comparison between SVM and LSTM, each algorithm has its pros and cons as in Table III, but in general, LSTM is better than SVM.

XI. Conclusion

Throughout the project's research, development and implementation, it was evident that there was a real need for adopting a solution for the web application security problems

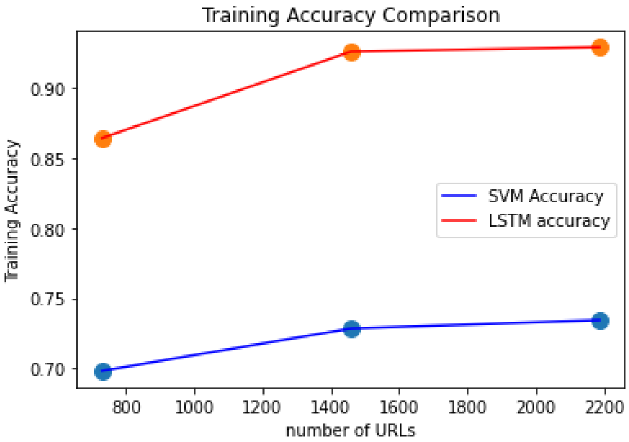


Fig. 4. Comparing LSTM and SVM Training Accuracy .

TABLE III
COMPARING SVM AND LSTM ALGORITHMS

LSTM	SVM
Long time for training, but once it is trained the model can be saved.	Short time for training.
Size of data is large.	Size of data is small.
High accuracy and better performance than SVM.	Medium accuracy
Features learned through training.	Features extracted manually.

and phishing links detection, this was because millions of websites are hacked each year and fall victim to many attacks. The project has developed mitigation techniques for various web application attacks in which each attack considered has one to three mitigation techniques implemented. This project aims to enhance the detection of phishing websites using machine learning technology. It also took into considerations developing the best technique for detection in which a comparison is made between three machine learning algorithms and deep learning using Long short term memory, 98% detection accuracy was achieved using LSTM, and hence it is chosen, the project also saves the user logs in a file according to apache log format. Finally, the most important conclusion in web application security is never trusting user input hence the first golden rule of user input is, all input is bad until proven otherwise. Typically, the moment you forget this rule is the moment you are attacked. For our website to be more secure and hence protecting the users' data and privacy, it is recommended to use a combination of security protocols (such as HTTPS, which is an encryption method) and security technologies (such as firewalls, intrusion detection, and prevention systems). There is a need for more investigation and analysis of web application attacks because new attacks and hacking techniques emerge continuously. Further enhancement of phishing URLs detection could be increasing the training data. A combination or hybrid machine learning and deep learning algorithm can also be implemented to improve accuracy. Also, log files can be used for further work in analyzing logs to detect anomaly behavior and dangerous attacks. Finally, the proposed solution should have the ability to be deployed on different platforms. This will enhance its adaptability and maximize the benefits derived through it.

REFERENCES

[1] S. Kumar, R. Mahajan, N. Kumar, and S. K. Khatri, "A study on web application security and detecting security vulnerabilities," in *2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. IEEE, 2017, pp. 451–455.

[2] I. Yusof and A.-S. K. Pathan, "Preventing persistent cross-site scripting (xss) attack by applying pattern filtering approach," in *The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*. IEEE, 2014, pp. 1–6.

[3] M. M. A. Baig, "Security vulnerabilities in php applications," Ph.D. dissertation, San Diego State University, 2012.

[4] V. Yerram and G. V. R. Reddy, "International journal of research in computer applications and robotics," 2014.

[5] R. Das, M. Hossain, S. Islam, A. Siddiki *et al.*, "Learning a deep neural network for predicting phishing website," Ph.D. dissertation, Brac University, 2019.

[6] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, "Classifying phishing urls using recurrent neural networks," in *2017 APWG symposium on electronic crime research (eCrime)*. IEEE, 2017, pp. 1–8.

[7] R. A. Katole, S. S. Sherekar, and V. M. Thakare, "Detection of sql injection attacks by removing the parameter values of sql query," in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. IEEE, 2018, pp. 736–741.

[8] J. Garcia-Alfaro and G. Navarro-Arribas, "A survey on cross-site scripting attacks," *arXiv preprint arXiv:0905.4850*, 2009.

[9] A. Rahman, M. M. Islam, and A. Chakraborty, "Security assessment of php web applications from sql injection attacks," *Journal of Next Generation Information Technology*, vol. 6, no. 2, p. 56, 2015.

[10] I. Papagiannis, M. Migliavacca, and P. Pietzuch, "Php aspis: using partial taint tracking to protect against injection attacks," in *2nd USENIX Conference on Web Application Development*, vol. 13, 2011.

[11] M. S. Tajbakhsh and J. Bagherzadeh, "A sound framework for dynamic prevention of local file inclusion," in *2015 7th Conference on Information and Knowledge Technology (IKT)*. IEEE, 2015, pp. 1–6.

[12] A. Begum, M. M. Hassan, T. Bhuiyan, and M. H. Sharif, "Rfi and sqli based local file inclusion vulnerabilities in web applications of bangladesh," in *2016 International Workshop on Computational Intelligence (IWCI)*. IEEE, 2016, pp. 21–25.

[13] Z. Mao, N. Li, and I. Molloy, "Defeating cross-site request forgery attacks with browser-enforced authenticity protection," in *International Conference on Financial Cryptography and Data Security*. Springer, 2009, pp. 238–255.

[14] K. Apostol, "Brute-force attack," 2012.

[15] N. Singh, K. Singh, and R. S. Raw, "Analysis of detection and prevention of various sql injection attacks on web applications," *Int. J. Appl. Inf. Syst*, vol. 2, pp. 20–26, 2012.

[16] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Phishing websites features," *School of Computing and Engineering, University of Huddersfield*, 2015.