







An Ensemble Based Deep Learning Framework to Detect and Deceive XSS and SQL Injection Attacks

Waleed Bin Shahid^(✉) , Baber Aslam, Haider Abbas , Hammad Afzal ,
and Imran Rashid 

National University of Sciences and Technology, Islamabad 44000, Pakistan
{waleed.shahid, ababer, haider, hammad.afzal, irashid}@mcs.edu.pk
<https://www.mcs.nust.edu.pk>

Abstract. Safeguarding websites is of utmost importance nowadays because of a wide variety of attacks being launched against them. Moreover, lack of security awareness and widespread use of traditional security solutions like simple Web Application Firewalls (WAFs) has further aggravated the problem. Researchers have moved towards employing sophisticated machine learning and deep learning based techniques to counter common web attacks like the SQL injection (SQLi) and Cross Site Scripting (XSS). Lately, keen interest has been taken in tackling these attacks through cyber deception. In this paper, we propose an ensemble based deep learning approach by combining Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) models. This detection framework also contains a Session Maintenance Module (SMM) which maintains user state in an otherwise stateless protocol by analyzing cookies thereby providing further optimization. The proposed framework detects SQLi and XSS attacks with an accuracy of 99.83% and 99.47% respectively. Moreover, in order to engage attackers, a deception module based on dockers has been proposed which contains deceptive lures to engage the attacker. The deceptive module has the capability to detect zero-days and is more efficient when compared to other similar solutions.

Keywords: SQL injection attacks · XSS · Deep learning · Deception

1 Introduction

Internet has grown very rapidly in recent years providing outstanding benefits to its users all across the globe. This has enabled businesses, corporations and technically sound people across the world to share data and information in a simple and easy way. This massive data sharing and information dissemination

Sponsored by the Higher Education Commission (HEC), Pakistan through its initiative of National Center for Cyber Security for the affiliated lab National Cyber Security Auditing and Evaluation Lab (NCSAEL), Grant No: 2(1078)/HEC/ME/2018/707.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
N. T. Nguyen et al. (Eds.): ACIIDS 2022, LNAI 13757, pp. 183–195, 2022.
https://doi.org/10.1007/978-3-031-21743-2_15

is mainly carried out through hundreds of millions of websites which are growing day by day [1]. These web applications are connected with back-end databases which contain useful and at times critical organizational data reserved and available only for authorized and legitimate users. Unfortunately, this overwhelming use and resulting ease in data/information sharing and dissemination has come with a price as people with malicious intent do their best to compromise these web applications to serve their illicit purpose. They either subvert web application's security to gain illegitimate access to critical data stored in back-end database or want to disrupt the availability of the website by turning it down or damage it's integrity by defacing it. Another important aim of the attackers is to find vulnerabilities in the target web application to place a malicious script or payload there with an aim to infect everyone visiting that website.

OWASP [2] lists common web attacks based on the ease of exploitation, difficulty in patching and their frequency of occurrence. SQL injection (SQLi) and Cross Site Scripting (XSS) attacks are a very common opportunity for attackers to compromise the website, its back-end database and the visiting benign users. For instance, attackers exploit XSS bugs in order to inject the vulnerable target website with malicious JavaScript so that all normal users visiting the website also get infected and start sending the desired information/data to attacker. Figure 1 explains how an XSS attack takes place.

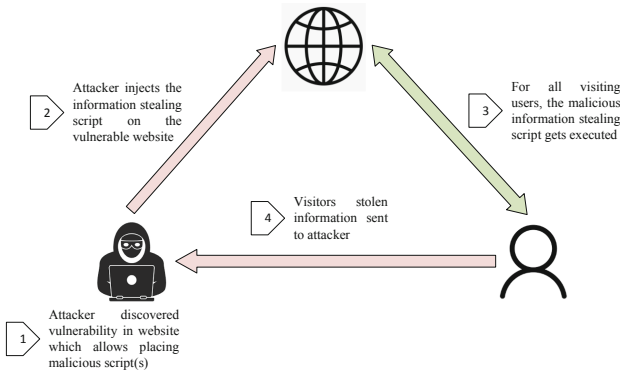


Fig. 1. A cross site scripting attack

SQL injection attacks take place when attackers supply untrusted data or commands to a website especially in the authentication fields. These SQL payloads and statements are executed by the vulnerable website resulting in providing the attacker illegitimate access to the back-end database. Security researchers have used various techniques like Web Application Firewall (WAF) to counter these web attacks [3]. WAFs have many drawbacks as they are unable to deal

with sophisticated attackers, custom payloads and scripts, fail against zero-days and are mostly rule-based lacking contextual or behavioral analysis of the incoming HTTP request. This persuaded researchers to come up with machine learning and later deep learning based approaches to counter XSS and SQL injection attacks. Apart from detecting and stopping malicious HTTP requests, researchers are taking ever increasing interest in absorbing these attacks by deceiving the attackers with the help of deceptive lures. This helps in studying and analysing the attack methodologies and attacker behaviour in a lot more detail. The proposed framework achieves considerable success in firstly detecting these two common web attacks and later absorbing them by engaging the attacker with the help of deceptive lures. The main research contribution of this paper is as follows:

1. An ensemble based deep learning framework using CNN and LSTM classifiers has been proposed which detects SQL injection and Cross Site Scripting attacks with very high accuracy.
2. The attack detection module is supported by a State Maintenance Module that helps maintaining the state of attacker resulting in successful attacker profiling and categorization, which strengthens both attack detection and deception.
3. A light-weight high interaction docker based deception module which comprises of a docker daemon that controls and manages SQL injection and XSS attack dockers where deceptive lures have intentionally been placed to engage attackers in real time.

1.1 Background Study

Many research works have used deep learning based approaches in order to counter web application attacks. Luo et al. [4] proposed an ensemble classification model based with three classifiers. The research gives high accuracy and low false positive values. Niu et al. [5] proposed framework to counter web attacks by joining Convolutional Neural Network (CNN) with Gated Recurrent Unit (GRU) in order to yield high accuracy. Both research works used the CSIC data [6] for model training and testing. Tae-Young Kim and Sung-Bae Cho [7] combined CNN and LSTM models to deeply analyze the incoming web traffic with high accuracy. Adem Tekerek [8] used CNN to detect web attacks with high accuracy. Yao Pan et al. [9] used end to end deep learning for detecting XSS and SQLi attacks. Fawaz Mahiuob et al. [10] used an artificial neural network approach for detecting XSS attacks by dynamically extracting traffic features.

All these approaches lack the key feature of analyzing state of the incoming traffic thereby lacking the ability to maintain attacker's profile overtime. Few research works have focused on profiling attackers in order to enhance the attack detection capability [11]. Moreover, these techniques only counter attacks and do not have any deception module to engage the attacker. Nevertheless, many stand alone deception systems have been proposed which are based on docker containers. SMartin Valicek et al. [12] used dockers to create high interaction

honeypots for Linux and Windows. Gaspari et al. [13] also used dockers to propose an architecture which empowers the production system for active defense. Andronikos Kyriakou et al. [14] proposed a docker based honeypot approach by deploying different standalone honeypots.

Most of the deception solutions do not offer customized deception and are static in nature. Moreover, they are not coupled with attack detection modules. Since the proposed technique is based on an ensemble approach using CNN and LSTM models along with a Session Maintenance Module which helps in maintaining the attacker’s state thereby augmenting the performance and efficacy of the deception module.

2 Proposed Detection and Deception Technique

The proposed framework for detecting and deceiving SQLi and XSS attacks is shown in Fig. 2. Here, the attacker’s request first passes through the detection module where the SMM does the profiling and the ensemble based classifier detects and forwards it to the attack specific dockers.

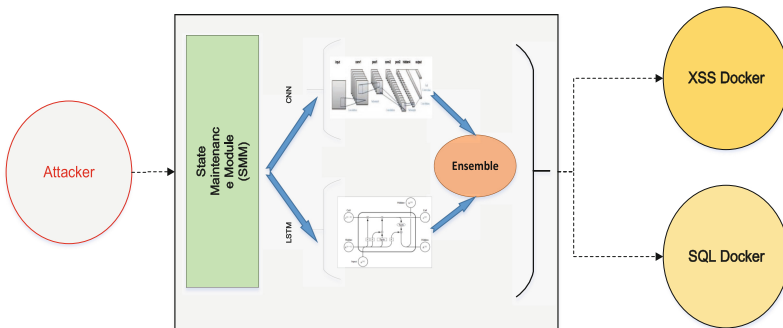


Fig. 2. The proposed framework

For the purpose of training and testing the deep learning classifiers (CNN and LSTM), we employed publicly available benchmark datasets [15, 16] which contain both benign and anomalous XSS and SQL injection request packets.

2.1 Data Preparation and Feature Selection

The dataset containing XSS and SQL injection attack packets is in the form of network flows, therefore, it is important to understand the significance of these flows which would facilitate in picking only the necessary and relevant flows for model training. These bi-directional flows reflect quite a lot about the nature of traffic as they carry detailed information about the traffic being sent

to and received from the target website. For that matter, it is not necessary that all network flows will play a decisive role in determining the nature of HTTP request. If a network flow does not relate with the type of attack, there is no point in using it as a feature for model training. For that matter, while selecting/choosing the network flows for training our deep learning classifiers, it was made sure that:

- All flows which had the value *Zero/NULL* were eliminated as they do not participate at all in any sort of classification.
- All network flows which had static information like source and destination socket information were eliminated as it does not play any part in determining either XSS or SQL attacks.
- Network flows which actually carry information about XSS and SQL injection payload and scripts etc. are preferred. For instance, total bytes in the HTTP request, HTTP request body, destination packets and length of the HTTP response are some key network flows which carry immense value while dealing specifically with XSS and SQL injection attacks.

As a result only 29 bi-directional network flows were chosen out of whom 14 carried numerical data. Since LSTM and CNN classifiers deal with numerical data, therefore, the remaining 15 network flows needed to be converted from categorical to numerical form. For that purpose the pre-processing label encoder in *sklearn* [17] was used which labels in a way such that the label values only range from 0 to *total.classes-1*.

2.2 Using the Ensemble Based Deep Learning Classifiers

Ensemble based deep learning approaches help reduce variance of neural networks by combining predictions from various models because the bias added by combining these predictions helps reduce variance of a single employed neural network model [18].

CNN Classifier. For training the convolutional neural network based classifier, the feature set is defined as \mathbf{S} which comprises of features (s_1, s_2, \dots, s_{29}). We then apply the embedding, dropout, convolution layers to give the output as shown in Eq. 1 where A_{ReLU} denotes the activation function, win denotes the window size, W represents the kernel weight and λ represents bias for a particular feature s_i .

$$C_l^1 = A_{ReLU} \left(\sum_{i=1}^{win} W_l^c \cdot s_x^c + \lambda_{sl}^c \right) \quad (1)$$

After convolution, the pooling, dense and activation layers are applied after which the model is projected onto an output layer that has two neurons for two classes by using the *Softmax* activation function. Table. 1 shows the CNN model with all its layers and parameters when applied on the SQL dataset.

Table 1. CNN model output for SQL injection attacks

Layer (type)	Output shape	Parameters
Embedding	(None, 33, 256)	4968192
Dropout	(None, 33, 256)	0
Conv1D	(None, 31, 250)	192250
Pooling	(None, 250)	0
Dense	(None, 250)	62750
Dropout	(None, 250)	0
Activation	(None, 250)	0
Dense	(None, 10)	2510
Activation	(None, 10)	0

LSTM Classifier. The same dataset containing both SQL injection and XSS attack requests was also used to train the LSTM classifier. A standard neural network algorithm typically comprises of an input activation along with the output activation. Both these layers relate with each other through an activation function. In case of LSTM classifier, the input activation after its application gets multiplied by some factor and later added due to the recurrent self-connection. Table 2 shows the LSTM model with all its layers when applied on the XSS dataset.

Table 2. LSTM model output for XSS attacks

Layer (type)	Output shape	Parameters
Input	(None, 10, 28)	0
LSTM	(None, 10, 128)	80384
Flatten	(None, 1280)	0
Output (Dense)	(None, 10)	1280

Both deep learning models have been combined using the ensemble approach in order to come up with a final and optimal classification.

2.3 State Maintenance Module

The state maintenance module is responsible for maintaining the attacker's state by analyzing cookies as they are provided by the website to the visiting users in order to provide a better user experience. Therefore, they contain some data specific to the visiting users. Every time they visit the website, they are being identified through the cookies they bring. If the ensemble based detection module labels an incoming request as malicious, the SMM would log this activity so that in future the user is directly diverted to the deception module by just analyzing the cookie field. This helps in optimizing the proposed framework of attack

deception and deception. Moreover, the SMM is also capable of finding out any mutation, presence of any script or unwanted content in the cookie field.

2.4 Deception Module to Lure/Engage Attackers

After successfully detecting SQL and XSS attacks, attack packets are forwarded to respective dockers via the docker daemon. These dockers contain deceptive lures to engage the attackers. The prime purpose of using the docker approach is the operational flexibility, design simplicity, easy runtime spawning and memory efficiency it provides, unlike virtual machines [19]. Moreover, dockers share the common Operating System (OS) which is not a problem because both SQL and XSS attacks reside at the application layer and do not interfere with the underlying OS.

Docker Daemon. The *docker daemon* which is securely controlled by the administrator and is responsible for integrating the XSS and SQL dockers with the ensemble based deep learning module.

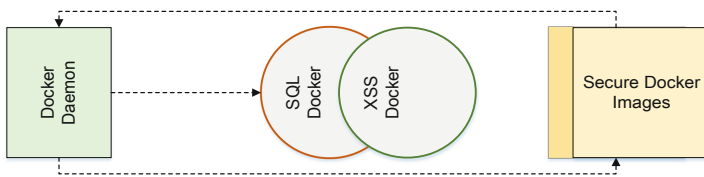


Fig. 3. The Docker Daemon controlling attack specific dockers

The daemon also fetches securely placed docker images when required and performs centralized log management as shown in Fig. 3.

Securing Docker Containers. In order to base the deception module on docker containers, it was important to prioritize the security of all dockers so that attackers fail to compromise security of the docker daemon or the attack specific dockers. For that purpose, all dockers were regularly updated and patched against known vulnerabilities (other than the ones intentionally placed), were run with limited permissions and were enabled with Docker Trust Management.

SQL Injection Docker. This docker is aimed to engage attackers launching SQL injection attacks by providing them the requisite lures. Moreover, any extension in the proposed deception module would connect this docker with other attack and pre-attack dockers further enhancing the deceptive capability. This docker contains few authentication pages which belong to the actual website by removing the SQL related validation and sanitization checks so that whenever

an attacker generates an HTTP request containing an SQL payload, he/she will be routed towards this docker by the docker daemon, lured in and engaged to extract useful attack related information. The SQL injection docker is further explained in Fig. 4.

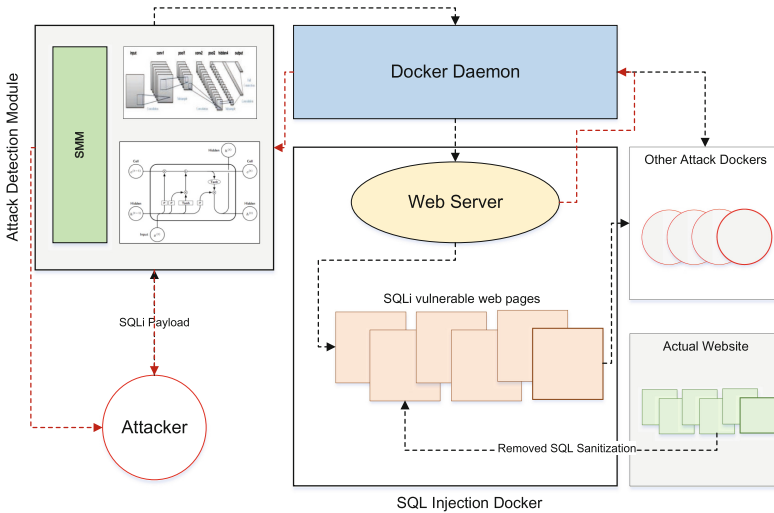


Fig. 4. SQL injection attack docker

XSS Docker. XSS attacks are applicable only on those web pages which basically require user input in some form because malicious JavaScript after execution results in a compromise thereby providing the attacker with victim’s information in an unlawful way. In order to build the XSS docker, the original website, where the entire deception module was deployed, was patched against all cross site loopholes. The difference in this docker is that it contains both XSS vulnerable web pages and replicas of some secure/patched web pages from the actual website. These were added to thwart the possibility of finding the deceptive system by an attacker who could possible perform a comparative analysis of received responses by coming from an altogether different identity. On the other hand, if attackers share links of the vulnerable XSS pages with anyone, he/she will also be diverted to this docker by the docker daemon (Fig. 5).

3 Discussion, Performance Analysis and Testing

In order to test the proposed deep learning models we made use of the testing dataset which was separated earlier from the training dataset. Both the CNN

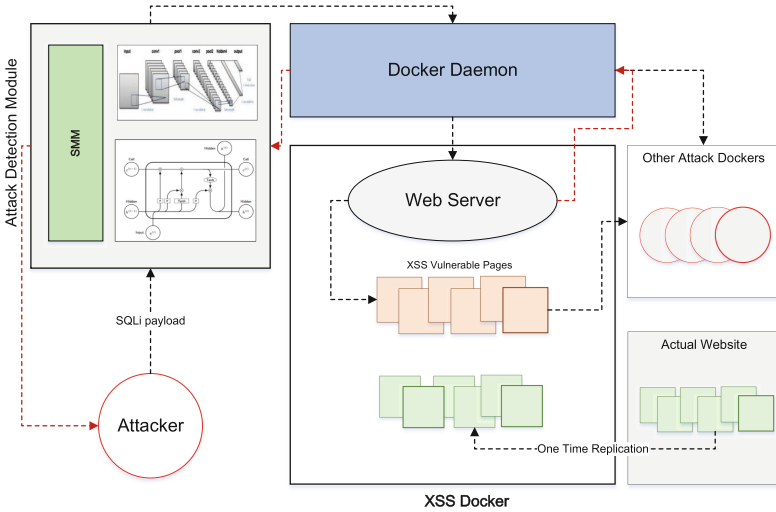


Fig. 5. Cross site scripting docker

and LSTM models were executed on this testing dataset comprising of SQL and XSS traffic (attack and benign requests) for 10 epochs in order to maximize accuracy and minimize validation loss. Later the ensemble classifier made the final classification decision by combining the CNN and LSTM classifications. Figures 6 and 7 show the validation accuracy and validation loss in detecting XSS attacks by both classifiers. It is evident that the LSTM based deep learning classifier outperforms the CNN classifier in accurately detecting cross site scripting attacks. Similarly, both deep learning models were tested on the SQL injection dataset comprising of both benign and malicious requests in order to yield a very high accuracy and minimized validation loss as shown in Figs. 8 and 9. Here, the CNN classifier yields better performance as compared to LSTM in accurately predicting SQL injection attacks, thereby contributing more in the ensemble based decision.

As far as the performance of the deception module is concerned, it was observed that by using the docker based approach, no major delay was observed in response time as HTTP responses by both the attack dockers were very minutely greater than response time of the actual website as shown in Eq. 2, thereby not letting the attacker notice about presence of any deception module.

$$T_d \leq T_w \tag{2}$$

It was observed during testing, that for 141 malicious SQL requests, the SQL attack docker successfully engaged the attacker for 89 requests by reaching a maximum engagement level of 5.

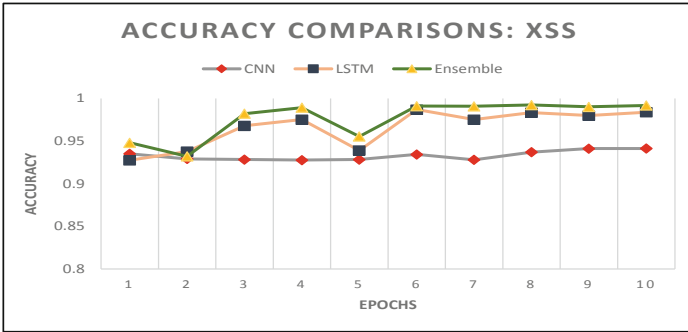


Fig. 6. Validation Accuracy (XSS): CNN, LSTM, Ensemble

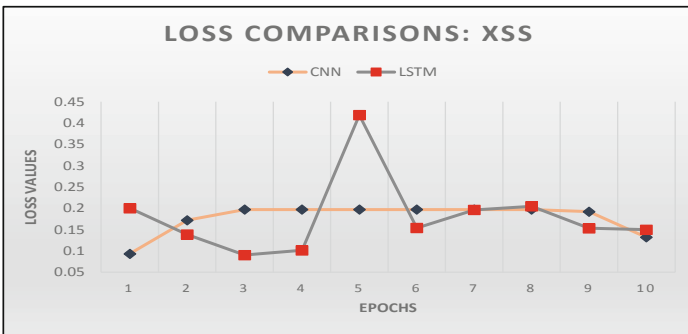


Fig. 7. Validation Loss (XSS): CNN vs. LSTM

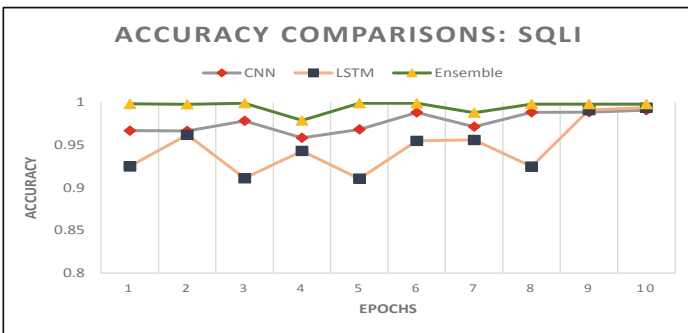


Fig. 8. Validation Accuracy (SQLi): CNN, LSTM, Ensemble

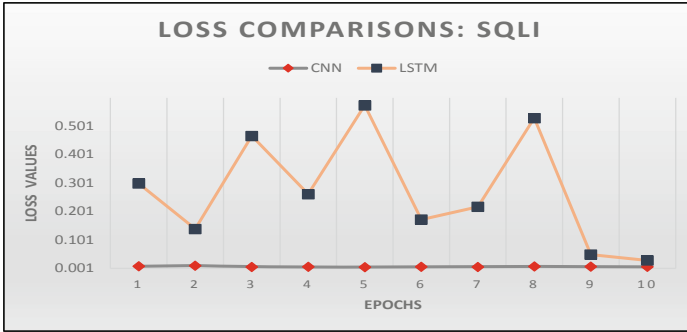


Fig. 9. Validation Loss (SQLi): CNN vs. LSTM

3.1 Comparative Analysis

The proposed deception framework was compared with Glastopf [20], another honeypot solution for countering web application attacks, in Table 3. It was observed that our technique is more optimized because of the State Maintenance Module which helps in recording the attacker profile and then dealing with all future requests from the attacker without much of processing. This feature is the hallmark of the proposed framework and helps it to detect zero-day attacks because attackers send a lot of reconnaissance and scanning packets to the target website before injecting it with zero-day attack payloads. Therefore, once detected in the pre-attack phase, the SMM would help detect the unknown zero-day attacks without much problem.

Table 3. Proposed deception module vs. Glastopf

Functionality	Glastopf [20]	Proposed framework
Maintaining attacker' state	No	Yes
Effectiveness of deceptive lures	Low	High
Runtime Docker spawning capability	No	Yes
Zero day attacker engagement	No	Possible
Centralized control	No	Yes
Customization	Difficult	Easy

4 Conclusion and Future Work

The rapidly evolving threat landscape has motivated security researchers to protect websites which carry useful data along with the back-end systems holding

critical organizational data. Researchers have focused on using advanced techniques to counter these attacks that target web applications. This research work proposes an ensemble based deep learning approach comprising of CNN and LSTM classifiers along with a State Maintenance Module for detecting XSS and SQL injection attacks with very high accuracy. Deception is being carried out with the help of a full-fledged deception framework based on docker containers. In this module, the highly secure docker daemon controls the attack dockers that have lures to engage the attacker. In future, we intend to enhance the deception framework by adding more pre-attack and attack specific dockers and expose the entire detection and deception solution to live data in a production environment.

References

1. Lindsay Liedke.: 100+ Internet Statistics and Facts for 2020. <http://www.websitehostingrating.com/internet-statistics-facts/>. Accessed 29 Mar 2021
2. The Open Web Application Security Project.: OWASP Top Ten. <http://owasp.org/www-project-top-ten/>. Accessed 25 Mar 2021
3. Clincy, V., Shahriar, H.: Web application firewall: network security models and configuration. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), vol. 01, pp. 835–836 (2018)
4. Luo, C., Tan, Z., Min, G., Gan, J., Shi, W., Tian, Z.: A novel web attack detection system for internet of things via ensemble classification. *IEEE Trans. Ind. Inform.* **01**, 1 (2020). [https://doi.org/10.1109/TII.2020.3038761\(2018\)](https://doi.org/10.1109/TII.2020.3038761(2018))
5. Niu, Q., Li, X.: A high-performance web attack detection method based on CNN-GRU model. In: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), vol. 01, pp. 804–808 (2020). <https://doi.org/10.1109/ITNEC48623.2020.9085028>
6. Giménez, C.T., Villegas, A.P., Marañón, G.Á.: HTTP DATASET CSIC 2010. <http://www.isi.csic.es/dataset/>. Accessed 8 Nov 2021
7. Kim, T.-Y., Cho, S.: Web traffic anomaly detection using C-LSTM neural networks. *Expert Syst. Appl.* **106** (2018). <https://doi.org/10.1016/j.eswa.2018.04.004>
8. Tekerek, A.: A novel architecture for web-based attack detection using convolutional neural network. *Comput. Secur.* **100**, 102096 (2021). <https://doi.org/10.1016/j.cose.2020.102096>
9. Pan, Y., et al.: Detecting web attacks with end-to-end deep learning. *J. Internet Serv. Appl.* **10** (2019). <https://doi.org/10.1186/s13174-019-0115-x>
10. Mokbal, F.M.M., Dan, W., Imran, A., Jiuchuan, L., Akhtar, F., Xiaoxi, W.: MLPXSS: an integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique. *IEEE Access* **7**, 100567–100580 (2019). <https://doi.org/10.1186/s13174-019-0115-x>
11. Shahid, W.B., Aslam, B., Abbas, H., Khalid, S.B., Afzal, H.: An enhanced deep learning based framework for web attacks detection, mitigation and attacker profiling. *J. Netw. Comput. Appl.* **198**, 103270 (2022)
12. Valicek, M., Schramm, G., Pirker, M., Schrittwieser, S.: Creation and integration of remote high interaction honeypots. In: 2017 International Conference on Software Security and Assurance (ICSSA), pp. 50–55 (2017). <https://doi.org/10.1186/s13174-019-0115-x>

13. De Gaspari, F., Jajodia, S., Mancini, L.V., Panico, A.: AHEAD: A New Architecture for Active Defense, pp. 11–16. Association for Computing Machinery (2016). <https://doi.org/10.1145/2994475.2994481>
14. Kyriakou, A., Sklavos, N.: Container-based honeypot deployment for the analysis of malicious activity. In: 2018 Global Information Infrastructure and Networking Symposium (GIIS), pp. 1–4 (2017). <https://doi.org/10.1109/GIIS.2018.8635778>
15. The TON IoT Datasets. <http://research.unsw.edu.au/projects/toniot-datasets>. Accessed 7 Oct 2021
16. Stratosphere Lab: A labeled dataset with malicious and benign IoT network traffic. <http://www.stratosphereips.org/datasets-iot23>. Accessed 4 Oct 2021
17. sklearn.preprocessing.LabelEncoder. <http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>. Accessed 8 Nov 2021
18. Polikar, R.: Ensemble Machine Learning, pp. 1–34. Springer, New York (2012). <https://doi.org/10.1007/978-1-4419-9326-7>
19. Shahid, W.B., Aslam, B., Abbas, H., Afzal, H., Khalid, S.B.: A deep learning assisted personalized deception system for countering web application attacks. *J. Inf. Secur. Appl.* **67**, 103169 (2022)
20. Mphago, B., Mpoeleng, D., Masupe, S.: Deception in web application honeypots: case of Glastopf. In: International Journal of Cyber-Security and Digital Forensics, vol. 6, pp. 179–185. The Society of Digital Information and Wireless Communications (2017)