International Science and Technology Journal
المجلة الدولية للعلوم والتقنية
Volume 38 العدد
Part 2 المجلد
ISTJ

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

Received	**2026/05/23**	**تم استلام الورقة العلمية في**
Accepted	**2026/06/16**	**تم قبول الورقة العلمية في**
Published	**2026/06/18**	**تم نشر الورقة العلمية في**

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

Haithem Ali Alghazzawi[1], Mohamed Elbeshti[2]
Abdalrahman Alfagi[3], Sami Naji[4]

[1]The Libyan Academy, School of Applied Sciences and Engineering, Libya
210100071@academy.edu.ly, https://orcid.org/0009-0000-4021-008
[2]Department of Information Technology, IT faculty
University of Zawia, Libya
m.elbeshti@zu.edu.ly. https://orcid.org/0009-0005-5897-542X
[3]Department of Information Technology, IT faculty
University of Zawia, Libya
Abd.alfagi@zu.edu.ly, https://orcid.org/0000-0002-7590-2004
[4]STEM Division, American University of Afghanistan (AUAF);
snaji@auaf.edu.af, https://orcid.org/0009-0006-8360-7909

Abstract

Most modern digital services utilize web applications to serve their clients continuously; whereby large amounts of sensitive information are processed in real time. Many cybersecurity threats compromise confidentiality, integrity, and availability, causing financial losses and legal consequences. For instance, Cross-Site Scripting (XSS) is the root cause of 80% of online data breaches. The XSS exploit weaknesses in web applications by injecting malicious scripts into user requests. However, Traditional Web Application Firewalls (WAFs) often rely on static rule-based approaches that may not effectively detect XSS attack patterns in modern web environments. This study proposes an enhanced WAF that integrates a deep learning Long Short-Term Memory (LSTM)

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

model as an intelligent XSS detection layer named XSSLayer positioned alongside the traditional WAF for improving its real-time detection capabilities. The XSSLayer analyzes all incoming requests and identifies abnormal or malicious script patterns before forwarding the analysis results to the WAF, enabling the firewall to forward only valid traffic to reach the web application server. Thus, the LSTM-based detection layer and the WAF collaboratively operate to detect and block XSS attacks efficiently. The integrated system was tested and evaluated using several performance metrics, including accuracy and loss. The obtained results showed that the enhanced Web Application Firewall (WAF), with an integrated Long Short-Term Memory (LSTM), can classify malicious scripts within web requests, whereas the training accuracy reached approximately 98.0% and the validation accuracy was about 99.4%. In addition, the results showed a validation loss of about 0.02, that indicate a high predictive reliability and minimal prediction error. The confusion matrix analysis showed only 25 false positives out of 2,600 benign requests and 11 false negatives out of 2,137 attack instances, which indicates the model's robust discriminative capacity. Furthermore, the obtained results of precision, recall, and the F1-score indicate that the model shows an exceptional capability, with nearly perfect performance in detecting XSS attacks using the LSTM architecture in real-time. However, integrating the XSS model with the ModSecurity firewall introduced additional computational overhead and slightly increased reply time, memory, and CPU usage. However, the findings provide strong evidence that integrating LSTM deep learning models with traditional WAF systems substantially enhances the capability of detecting XSS attacks in real-time compared to conventional WAF.

Keywords: Web Application Firewall, Deep Learning, Attack Detection, Firewall, and XSS.

International Science and Technology Journal
المجلة الدولية للعلوم والتقنية
Volume 38 العدد
Part 2 المجلد
المجلة الدولية للعلوم والتقنية
International Science and Technology Journal
ISTJ

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

كشف هجمات XSSفي الوقت الحقيقي بناء على جدار حماية محسن لتطبيقات الويب من خلال التعلم العميق وإطار عمل MoDالمتكامل مع LSTM

هيثم علي الغازاوي[1]، محمد البشتي [2]، عبد الرحمن الفقي [3]، سامي ناجي[4]

[1]الأكاديمية الليبية، كلية العلوم التطبيقية والهندسة – ليبيا

[2]كلية تقنية المعلومات، قسم تكنولوجيا المعلومات، جامعة الزاوية – ليبيا

[3]كلية تقنية المعلومات، قسم تكنولوجيا المعلومات، جامعة الزاوية – ليبيا

[4]قسم STEM، الجامعة الامريكية الأفغانية

Abd.alfagi@zu.edu.ly

الملخص

تستخدم معظم الخدمات الرقمية الحديثة تطبيقات الويب لخدمة عملائها بشكل مستمر، حيث يتم معالجة كميات كبيرة من المعلومات الحساسة في الوقت الفعلي .العديد من تهديدات الأمن السيبراني تُعَرض سرية ونزاهة وتوافر البيانات للخطر، مما يسبب خسائر مالية وعواقب قانونية .على سبيل المثال، يعد تهديد **(XSS)** السبب الجذري في **80٪** من خروقات البيانات عبر الإنترنت . يستغل **XSS** نقاط الضعف في تطبيقات الويب عن طريق حقن سكريبتات خبيثة في طلبات المستخدمين .غالبا ما تعتمد جدران الحماية التقليدية لتطبيقات الويب **(WAFs)** على أساليب ثابتة قائمة قد لا تكتشف أنماط هجوم **XSS** بفعالية في بيئات الويب الحديثة .تقدم هذه الدراسة نموذج **WAF** محسن يدمج نموذج التعلم العميق للذاكرة طويلة المدى قصيرة المدى **(LSTM)** كطبقة كشف ذكية لتقنية **XSS** تسمى **XSSLayer** موضوعة جنبا إلى جنب مع **WAF** التقليدي لتحسين قدراته على الكشف في الوقت الحقيقي .يقوم **XSSLayer** بتحليل جميع الطلبات الواردة ويحدد أنماط السكريبتات غير الطبيعية أو الخبيثة قبل توجيه إلى **WAF**، مما يمكن الجدار الناري من إعادة توجيه حركة المرور الصالحة فقط للوصول إلى خادم تطبيقات الويب .لذا، تعمل طبقة الكشف المعتمدة على **LSTM** و **WAF** بشكل تعاوني لاكتشاف وصد هجمات **XSS** بكفاءة .تم اختبار النظام المتكامل وتقييمه باستخدام عدة

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

مقاييس أداء، بما في ذلك الدقة والفقدان .أظهرت النتائج التي تم الحصول عليها أن جدار الحماية المحسن لتطبيقات الويب **(WAF)**، مع ذاكرة طويلة المدى قصيرة المدمج **(LSTM)**، يمكنه تصنيف السكريبتات الخبيثة ضمن طلبات الويب، في حين بلغت دقة التدريب حوالي **98.0**٪ ودقة التحقق حوالي **99.4**٪ .بالإضافة إلى ذلك، أظهرت النتائج فقدان تحقق حوالي **0.02**، مما يشير إلى موثوقية تنبؤية عالية وخطأ توقع بسيط .كما أظهر تحليل مصفوفة الالتباس فقط **25** إيجابية كاذبة من أصل **2,600** طلب حميد و11 سلبيا كاذبة من أصل **2,137** حالة هجوم، مما يدل على قدرة النموذج القوية على التمييز الخاص .علاوة على ذلك، تشير النتائج التي تم الحصول عليها من الدقة، والاستدعاء، ودرجة **F1**إلى أن النموذج يظهر قدرة استثنائية، مع أداء شبه مثالي في اكتشاف هجمات **XSS**باستخدام بنية **LSTM** في الوقت الحقيقي .علماً بأن دمج نموذج **XSSLayer** مع جدار الحماية **ModSecurity** إلى زيادة إضافية في حجم العمليات الحسابية وزيادة طفيفة في وقت الرد والذاكرة المستخدمة وكذلك زيادة استخدام في سعة المعالجة المركزية .ومع ذلك، توفر النتائج أدلة قوية على أن دمج نماذج التعلم العميق **LSTM**مع أنظمة **WAF** التقليدية يعزز بشكل كبير قدرة اكتشاف هجمات **XSS** في الوقت الحقيقي مقارنة بنظام **WAF** التقليدي.

الكلمات المفتاحية :جدار حماية تطبيقات الويب **WAF**، التعلم العميق، اكتشاف الهجمات، جدار الحماية، و**XSS**

1. Introduction

Cross-Site Scripting (XSS) is measured as one of the most prevalent and dangerous threats to web applications, which are the primary sites that run individual or governmental organizations' businesses. The XSS vulnerability is the root cause of 80% of the online data breaches. According to [1] [2], the threats caused by XSS are increasing every day. In addition, the Open Web Application Security Project (OWASP) also confirmed that XSS is consistently ranked in the OWASP Top 10 as one of the most critical web application security risks [3]. In addition, web applications are the main interface between organizations and their customers.

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

Consequently, protecting websites against XSS is a critical business and legal necessity to preserve the Confidentiality, Integrity, and Availability, often referred to as the CIA triad, which is the heart of information security [4]. The importance of strengthening defense systems against XSS can be seen through several key dimensions, including, but not limited to:

- **Protection of Sensitive Operator/users' Data:** The primary goal of most XSS is to steal data by injecting malicious scripts.
- **Prevention of Account Takeover and Fraud (Integrity):** One of the most common payloads in an XSS attack is a script designed to steal raw cookies.
- **Preservation of Brand Reputation and User Trust:** When a user visits a trusted website and is served malicious content or has their data stolen, the trust in that brand is irrevocably damaged. A defaced website account being compromised can lead to negative media coverage and a mass exodus of users.
- **Ensuring Application Integrity and Availability:** XSS attacks undermine the Integrity of the application. An attacker can modify the content displayed to users (website defacement), insert fake forms that look similar to legitimate ones for phishing victims' credentials, or redirect them to malicious domains.
- **Defense Against Malware Distribution:** Sophisticated XSS attacks may be combined with other exploits, mostly known as Drive-by Download scenarios, to deliver malware.
- **Protection of Backend Systems:** the XSS could execute a chain of attacks at the client side to impersonate a privileged user.

Therefore, securing sensitive information of personal, individual, or governmental organizations is a very significant concern and critical issue that most researchers focus on to protect information from compromising, stealing, unauthorized modification, or executing a chain of attacks, which is important to avoid or at least mitigate a

http://www.doi.org/10.62341/istj-vol38-2-hi47

financial loss, legal consequences, and erosion of user trust. According to a recent data breach investigations report [5], every website is vulnerable to cyberattacks that threaten all or one of the CIA's attributes. In addition, [6] confirms that XSS is one of the top 10 attacks. Also, the authors in [7] clearly stated that XSS is one of the most common attacks during the year 2026, despite the ongoing studies, XSS remains one of the web vulnerability that deserve more researchers' effort. Therefore, there is strong evidence for the need to secure the websites against XSS attacks.

2. Background

This section explores the most common type of XSS attack with the aim of analyzing and illustrating how they operate. In addition, the section explains the working mechanism of WAF since it is a defense strategy that the authors aim to improve. Finally, the section explains the long short-term memory LSTM because it is a specialized type of Recurrent Neural Network (RNN) designed to recognize patterns in sequences of data, which is used for enhancing the WAF

2.1. Most Common Types of XSS Attacks

Mainly, XSS vulnerabilities are categorized into three types (Reflected, Stored, and DOM-based) [8]. These three types classify the XSS based on how the malicious payload travels from its source to the browser.

A. Reflected XSS (Non-Persistent)

According to [9], the Reflected XSS is classified as the most common type of XSS that poses a serious threat to the URL or HTTP of the web applications. The attacker injects a malicious script (payload) to be executed in the victim's browser. The attacker's payload is injected into the application. Once the user clicks that malicious link, the victim's browser then executes the injected malicious script, causing an error message or other response. In this type, the injected script is not stored on the target server. Instead, it is reflected off the web server immediately as part of a search result, error message, or any other response that includes some or all of the

http://www.doi.org/10.62341/istj-vol38-2-hi47

input sent to the server as part of the request [10]. Typically, attackers follow a specific sequence involving social engineering exploitation techniques. The steps start by identifying the reflection point, script delivery, and executing the script in the victim's browser once the victim clicks the link. The reflected XSS is often exploited through input that reflects the user's data onto the user's interface, such as form-based input, URL parameters, or other common payloads [11]. The primary impact and risk level of this type lies in the potential of stealing sensitive information such as session tokens, cookies, or personal data. While the risk of reflected XSS is potentially high, its impact is limited to the individual victim who clicks the malicious link.

B. Stored XSS (Persistent)

This is also a common type of XSS that is known as the most damaging type of XSS attacks. In the stored XSS, a malicious script is sent to the server in a single request for the aim of placing a malicious code directly into the target application's database. It follows a distinct path from injection to broad execution. The flow of a typical stored XSS attack is clearly explained by [12], which is firstly started by injecting and storing the malicious payload to a website through common input fields, such as posts' input, blog comments, or users' profiles. The second step operates when the legitimate users request the information stored in that database (such as viewing a blog post or a profile), thus the application serves the page containing the malicious script. Finally, as the application fails to escape the data, the malicious script executes automatically in the browsers of every unsuspecting visitor who loads the affected page. The stored XSS has a unique set of risks compared to Reflected XSS. In the stored XSS, every user who views the compromised page is a potential victim because the script is stored in the database. In addition, this type could cause a critical risk level since it can work as a web-based worm that may affect thousands of users without requiring them to perform a single click. Moreover, the stored XSS recorded a high impact since it relies on the server-side database.

http://www.doi.org/10.62341/istj-vol38-2-hi47

C. DOM-based XSS

This is the third common type of XSS, characterized as a more advanced variant of XSS where the entire tainted data flow remains within the browser's Document Object Model (DOM). The vulnerability of this type resides entirely within the client-side, based on JavaScript code. The mechanism and execution of DOM-based XSS lies in how the web applications' JavaScript handles data[10]. Firstly, the applications' client-side script takes user-controllable data, such as URL or input value, then the data is passed to a sink, which is a function that supports dynamic code execution, such as *eval(), innerHTML,* or *document.write()*. Finally, the JavaScript code handles the data unsafely, where the malicious script is injected into the page's DOM and executed. This type of XSS is distinguished from other types because it is hard and difficult to detect, and the risk consequently caused is considerably high

Table 1 summarizes the differences between the most common types of XSS Attacks. The table covers several features such as occurrence, persistence, location, delivery mechanism, risk level, primary challenge, and simple example. In addition, the table provides brief details for each feature.

Table 1. Differences between the most common types of XSS Attacks

Feature	Reflected XSS (Non-Persistent)	Stored XSS (Persistent)	DOM-based XSS
Occurrence/Status	Most common type.	Most damaging type.	Advanced variant.
Persistence	Non-Persistent; reflected off the server immediately.	Persistent; saved directly into the application's database.	Non-Persistent; exists entirely in the client-side code.
Storage Location	Not stored on the server.	Web application database (e.g., comments, profiles).	Browser's Document Object Model (DOM).
Mechanism	Social engineering (phishing links, deceptive social media).	Automatic delivery when a user views a compromised page.	Malicious data passed to a JavaScript "sink" (e.g., eval ()).
Target	The individual victim who clicks the link.	A mass audience; every visitor to the affected page.	The individual victim who visits a crafted link.

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

Feature	Reflected XSS (Non-Persistent)	Stored XSS (Persistent)	DOM-based XSS
Risk Level	High; requires user interaction.	Critical; can act as a web-based worm.	High.
Primary Challenge	Filtering unique, crafted URLs in server requests.	Identifying malicious payloads before they are saved to the database.	Hardest to detect because the server may never see the payload.
Example Scenario	Malicious script in a search query or error message.	Malicious script in a blog comment or forum post.	JavaScript reading a URL fragment and writing it to the page

2.2. Web Application Firewall (WAF)

The WAF is the first security defense line in web applications, which is a crucial security component to protect web applications from malicious attacks that originate from the application layer of the TCP/IP model. Unlike traditional firewalls that operate at the lower layers of the network, the WAF is designed to operate and detect anomalous traffic, inspect and filter incoming or outgoing packets in order to mitigate several attacks that may harm or affect CIA attributes. Normally, the WAFs depend on pattern-matching approaches to detect or prevent malicious packets based on either black-list or white-list strategies. In the black-list strategy, the traditional WAFs block packets' traffic based on predefined signatures of known attacks, which is generally cost-effective, while in this strategy, the WAF struggles to defend against unknown or variant attacks [2]. In contrast, the WAF's white-list approach allows only packets of traffic that match predefined patterns of normal behavior. However, implementing these strategies may block unknown attacks with the expanse of implementation due to the need for detailed information to define or filter normal traffic, resulting in limitations or blocking legitimate transactions. Since the traditional WAF operates on a set of predefined policies without any intelligence for filtering or packet inspection, the traditional WAF is particularly cost-effective since it needs detailed definitions to achieve its goals successfully [13].

http://www.doi.org/10.62341/istj-vol38-2-hi47

Generally, most traditional firewalls use similar strategies for protecting web applications. They may use defensive techniques such as service, direction, user, and behavior control to dominate access and to enforce the applications' security. Regardless of the defensive techniques in most traditional WAF, they are unable to protect or block novel threats. The authors in [13-17] highlighted several significant limitations and weaknesses that traditional WAFs face in modern cybersecurity. More specifically, the authors discuss the inability of traditional WAF to protect web applications against modern attacks. However, the security failure and functional limitations of traditional WAFs that are discussed in the articles are summarized as follows:

- **Traditional WAFs fail to address modern threats:** Traditional WAFs often struggle to eliminate or mitigate sophisticated attacks such as zero-day attacks, polymorphic malware, and excessive API calls.
- **Traditional WAFs are vulnerable to advanced techniques:** They are commonly useless against complex tactics such as sophisticated injection techniques and a huge traffic bandwidth flooding
- **Traditional WAFs are unable to detect internal threats:** The main limitation of traditional WAFs is their failure to defend against internal threats or the transfer of programs that are already infected with viruses
- **Traditional WAFs provide incomplete protection:** They are unable to guarantee complete protection against threats like malware circulation.

As many articles [13-17] argue that the traditional WAFs are insufficient against modern, sophisticated threats. Therefore, an enhanced Web Application Firewall Security is an active field that needs a robust WAF to protect an organization's web applications and their resources from sophisticated threats and modern attacks. Based on this evidence, the authors in this article use Deep Learning (DL) artificial intelligence (AI) to enhance WAF capabilities. Deep learning techniques have been used by the authors in [13] to provide an enhanced WAF for real-time DDOS detection using Deep

http://www.doi.org/10.62341/istj-vol38-2-hi47

Learning and integrating LSTM models. The authors follow a similar approach for a real-time XSS attack detection based on an enhanced web application firewall through LSTM.

3. Related work

Since XSS is a common vulnerability that is exploited to attack most web applications. The authors in [12] work to address the critical threat of XSS by developing a novel client-side filtering model called XSSFilter. The framework was designed to protect the end-users when the server-side defense is not sufficient. Their proposed method works as an extra layer of protection, focusing on detecting and stopping reflected XSS attacks. The idea is based on comparing the contents of the HTTP requests with the server's response. The framework seeks to detect the vulnerability and automatically modify malicious requests into safe ones in order to allow the user continually browsing without blocking any content. However, the XSSFilter focuses on Client-side filters, which are explicitly defined as a second line of defense where the server-side is not included in the protection. In addition, several protection methods that work on filtering the packets suffer from an unacceptable rate of false positives and false negatives. Moreover, the XSSFilter was able to detect vulnerabilities in 3 out of 5 sites, that indicating the detection rate is about 60%, whereby more improvement is required.

For better detection and defensive techniques, machine learning and deep learning have become an interesting area for enhancing web application security. According to Tekerek and Bay [18] and Kumar and Ponsam[19], the WAFs have a significant attention as both work to strengthen the security of web applications by focusing on advanced WAF models. Tekerek and Bay presented a hybrid technique by integrating two primary strategies, the Signature-Based Detections (SBD) and Anomaly-Based detection (ABD), for detection and preventing XSS. The SBD identifies known attack types such as SQL injection and XSS using a database of pre-established signatures, whereas the ABD is implemented mainly by Artificial Neural Networks (ANN) for the aim of identifying unusual HTTP requests. Similarly, Kumar and Ponsam focused on

http://www.doi.org/10.62341/istj-vol38-2-hi47

inspiring security by mixing a variety of machine learning (ML) models into a WAF, which serves as the primary defense layer. Even though the proposed methods by [18, 19] provided a high level of achievement in enhancing web application security through the use of machine learning and hybrid detection models. However, there are several boundaries and complexities connected with integrating machine learning (ML) models into web applications and enhancing WAFs, such as scalability and performance, since their evaluation is performed on static batch data, which can lead to high latency and dropped requests if it is used for real-time.

Recently, the authors in [20, 21] Both focused on advanced WAF technology based on the use of machine learning by addressing pattern categorization and real-time performance optimization. Teinh focuses on improving how WAFs identify and classify threats based on a deep automated pattern categorization by utilizing logistic regression and Convolutional Neural Networks (CNN) to automate the categorization of attack patterns. On the other hand, Nayar deployed a machine learning security model based on Support Vector Machines (SVM) to conduct a comprehensive performance and scalability analysis for high-traffic environments. Both Teinh and Nayar focused on deep automated pattern categorization, addressed real-time performance, and applied machine learning. However, there are some challenges that impact their practical deployment, including susceptibility to evasion attacks. In addition, both works are vulnerable to adversarial bypass and the extreme operational complexity required to maintain accuracy and low latency in real-time.

In 2025, [9] introduced an innovative generative AI framework called GenXSS, which is designed to enhance WAF defenses against XSS attacks. The GenXSS utilizes Large Language Models (LLMs) to systematize both the generation of complex attack payloads and the creation of self-protective security rules. The GenXSS follows two stages: the first stage generates sophisticated and obfuscated XSS payloads based on manually crafted examples, and then these payloads are validated for correctness against a vulnerable application. Finally, the payloads that successfully bypass a WAF

http://www.doi.org/10.62341/istj-vol38-2-hi47

are grouped using machine-learning clustering techniques, then analyzed to generate new WAF security rules for blocking the identified ones. Their results demonstrate an effective mitigation by generating just 15 new security rules; the framework was able to block 86% of the previously successful attacks that had bypassed the WAF. Despite the success of GenXSS in mitigating XSS attacks, the framework faces challenges, such as computational costs, due to LLMs restricting the generation of attack vectors. In addition, the score of 0.86 indicates that some attacks still managed to bypass the new rules.

Since the XSS vulnerability is the root cause of 80% of online data breaches. In addition, the most recent work that utilizes AI for web application protecting reach about 86% of protection rate, which means XSS remains one of the most concerning vulnerabilities that needs more researchers' effort.

4. Implementation methodology

The authors in this article follow a similar approach to that provided in [13] to enhance WAF utilizing the LSTM neural networks for the aim of Real-Time XSS attack detection. The flowchart shown in Figure 1 illustrates how the authors work to enhance traditional WAF to improve its ability to detect XSS attacks in web applications. The system combines preprocessing, followed by the deep learning training model, and then a classifier (decision maker) that determines whether the request is normal or malicious.

The flowchart works in four stages named dataset collection, pre-processing, deep learning, and decision making. At the first stage, the process starts with the dataset block, whereby the data collected from the primary sources is then labelled as traffic data for a range of scenarios. This study uses a real-world dataset named "Payload All the Things" [22]. The dataset contains a formatted version of the CSIC 2010 dataset with a Start - Id: xxx, where xxx is the ModSecurity unique id of the request, class with a valid or an attack value, full request including headers and body, and the dataset ended

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

with an End - Id: xxx, where xxx is the ModSecurity unique id of the request. In the pre-processing stage, several steps are performed before sending requests into the DL model. Since attackers may hide malicious payloads using encoding techniques, the Decoding step is used to reveal any hidden XSS attacks.

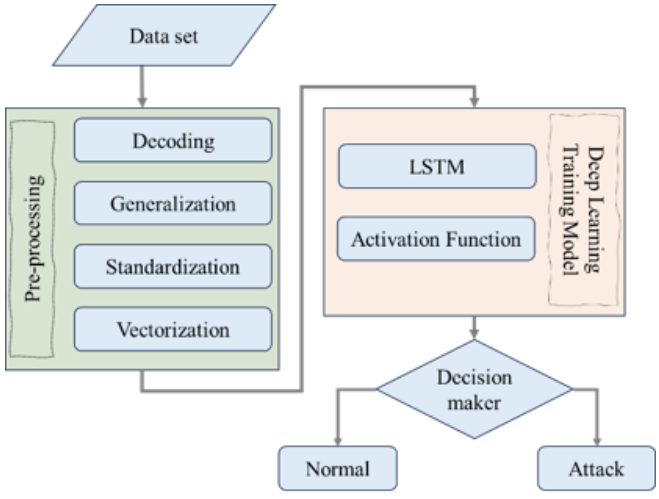

Figure: 1 Learning Implementation Flowchart

While the generalization step is used to convert the HTML into a common format to improve model consistency and detection accuracy. In addition, the standardization steps are a significant step to reduce noise in the dataset and for request cleaning, converting text to lowercase, removing unnecessary spaces, and normalizing symbols. Vectorization is also an important step since the DL model cannot process plain text directly; thus, vectorization transforms web requests into mathematical representations suitable for neural network analysis. After the pre-processing stage, the vectorized data is entered into the DL model, which works as an automatic feature extractor to extract important local features from the request. The dataset is divided into 80% training, 10 % validation, and 10 % testing subsets. During training, the DL model adjusted its internal weights to minimize classification error, while the final model is selected based on its accuracy and ability to identify XSS attack

http://www.doi.org/10.62341/istj-vol38-2-hi47

types. Meanwhile, the LSTM analyzes the sequence and context of the request since the XSS attacks often depend on character order, payload structure, or hidden contextual relationships. The activation function works to support the neural network in making better predictions and to expand learning performance and classification accuracy. The final stage in the flowchart is the Decision Maker, which totally depends on the trained model to classify the request into a normal or a Malicious XSS request.

4.1. Experimental Environment

This section illustrates the experimental environment for the enhanced Web Application Firewall (WAF). The environment consists of virtual components within a controlled network architecture, including a server setup: an Apache web server running on a Linux platform (Ubuntu 24.4.02 LTS). In addition, ModSecurity is installed and enabled as the main WAF, with PHP support for the web interface. Meanwhile, a bridge script (ml_waf_analyzer.sh) and a Python engine (waf_ml_integration.py) have been developed because ModSecurity does not support the AI models. The ModSecurity captures incoming HTTP requests and forwards them to the Python engine through the shell script. Also, the experimental environment installed Python 3.x and libraries such as TensorFlow, Keras, and NumPy to train the LSTM Model. Moreover, Python and Streamlit are utilized to design a web interface that allows manual data entry, CSV batch testing, and real-time visualization of classification results. The interface uses tab-based navigation to separate the pages. The LSTM network traffic model tester loads its respective LSTM model, which is stored in .h5 format, and provides real-time classification results. The experimental environment can be implemented using a PC with Intel Core i5/i7, 8–16 GB RAM, 256 GB SSD or higher storage with a Virtual Network configuration. The experimental process follows seven steps. Firstly, the clients send requests to the enhanced WAF, then the WAF preprocesses and normalizes the request data. After that, features are extracted from URLs, headers, parameters, and payloads. Then the AI-based detection engine analyzes the request

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

so the system can classify traffic as a legitimate pattern to be forwarded safely to the web server or an XSS attack that is blocked instantly.

5. The Enhanced WAF

Normally, a traditional WAF is the first line of defense between clients and the internet server, analyzing requests and responses for abnormal behavior. In the enhanced WAF shown in Figure 2, an extra layer named XSSLayer is added to the WAF structure to support its functionality. This layer works to extract the HTTP header and body parts to analyze the presented characteristics of XSS attacks based on a deep learning model. The layer is designed to act as a sensitive detection layer to analyze requests against pre-existing capabilities of XSS patterns in real-time.

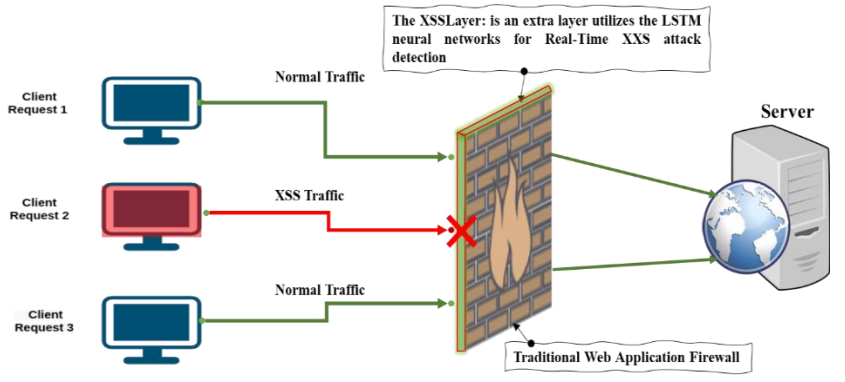

Figure: 2 The enhanced Web Application Firewall Security for real-time XSS attack detection

As shown in the figure, in normal operation, users send HTTP requests to the web application through the enhanced WAF. Normally, these requests may contain user login data, search queries, form submissions, URL parameters, JavaScript code fragments, or SQL statements embedded in inputs. All incoming traffic must pass through the enhanced WAF layer rather than directly reaching the server. The enhanced WAF, specifically XSSLayer, captures each request and accomplishes preprocessing

http://www.doi.org/10.62341/istj-vol38-2-hi47

operations such as packet inspection, URL decoding, input normalization, removal of unnecessary symbols, and tokenization of request parameters. These steps convert raw traffic into a structured format that is suitable for intelligent analysis.

Unlike traditional WAFs that depend only on static signatures and predefined rules, the enhanced WAF integrates a machine learning model that analyzes the sequence and behavior of request patterns in real time. The XSSLayer model detects injected XSS scripts intended to steal session cookies, execute malicious JavaScript, Hijack user sessions, or manipulate webpage content. This intelligent layer improves detection accuracy, especially against obfuscated payloads and previously unseen attack patterns. This layer forwards legitimate requests securely to the server, while malicious requests are identified and blocked immediately. In the figure, the green requests represent legitimate traffic, whereas the red requests represent an attack attempt that is denied by the enhanced firewall. The enhanced WAF continuously observes the incoming traffic behavior and updates detection patterns automatically to provide real-time detection, reduce false positives, and improve web application availability and confidentiality

6. Testing and result evaluation

The enhanced WAF system was experimentally evaluated to test its functionality and classification performance in real-time. In section 6.1, the evaluation is based on two main measurements, namely model accuracy and model loss. The model accuracy measures the proportion of correctly classified requests among all requests, while the model loss measures the prediction error. Section 6.2 presents the confusion matrix analysis, which offers a comprehensive evaluation of the model's ability to distinguish between legitimate (benign) web requests and malicious XSS attacks. In addition, the proposed model's performance by class is considered and presented in Section 6.3 based on Precision, Recall, and F1-Score. Finally, Section 6.4 examines the impact of integrating the proposed LSTM-based detection model with the ModSecurity firewall by evaluating

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

system performance and resource utilization before and after deployment

6.1. The Model Accuracy and the Model Loss

The model's accuracy is used to evaluate classification performance by measuring the proportion of correctly classified requests among all requests. The equation is given below, where TP (True Positive) represents the attack correctly detected as an attack, TN (True Negative) represents the normal traffic correctly identified as normal, FP (False Positive) represents normal traffic incorrectly classified as an attack, and FN (False Negative) represents an attack incorrectly classified as normal.

$$Accuracy = \frac{True\ Positives\ (TP) + True\ Negatives\ (TN)}{Total\ Number\ of\ Instances\ (TP+TN+FP+FN)} \quad [8]$$

Figure 3 illustrates the model accuracy over ten training epochs. The accuracy graph shows the performance of both Training Accuracy (Train Acc) and Validation Accuracy (Val Acc) over the learning process. As clearly seen, the training accuracy improves from approximately 91.7% in the first epoch to around 98.0% by the final epoch. In the same way, the validation accuracy increases from about 94.3% to approximately 99.4% over the same period.

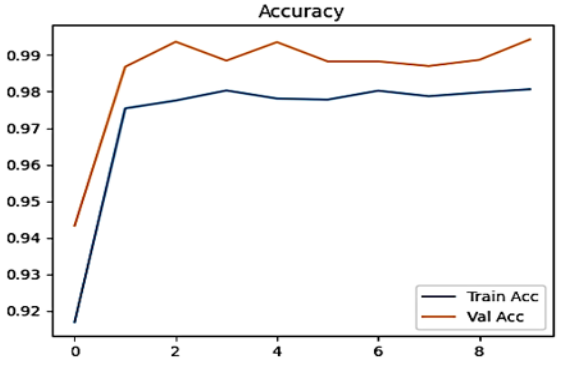

Figure: 3. Training Accuracy (Train Acc) and the Validation Accuracy (Val Acc) of the proposed LSTM model over 10 training epochs

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

From the graph, it can also be observed that after the second epoch, both curves become relatively stable, with only minor fluctuations, suggesting that the model has reached convergence. This behavior indicates that the model has a consistent performance over training epochs. Furthermore, the small gap between the two curves indicates that the model has good capability and high learning stability. Overall, testing model accuracy confirms that the model can provide reliable and accurate predictions in real-time environments.

The model's loss is used to measure the prediction error of the model during training. In this test, the equation given below is used to evaluate the model classification problems (model loss). Where: N is the number of samples, y_i represents the actual label, and $\tilde{y}_i$ represents the predicted probability. If the loss is close to zero means the model provides an excellent prediction, whereas a higher value for L means there is higher prediction errors in the model.

$$L = -\frac{1}{N}.\ \sum_{i-1}^{N} y_i \log\left(\tilde{y}_i\right) + (1 - y_i) \log\left(1 - \tilde{y}_i\right) \quad [8]$$

The result of testing the model loss is plotted in Figure 4.

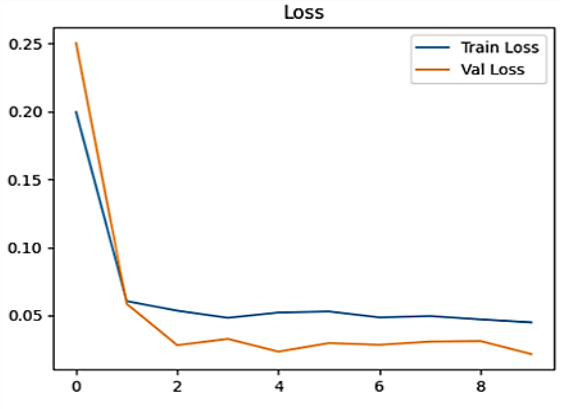

Figure: 4 The training loss (Train Loss) and validation loss (Val Loss) of the proposed LSTM model over 10 training epochs

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

The figure demonstrates the training loss (Train Loss) and validation loss (Val Loss) over 10 training epochs. As clearly seen, during the initial training process, both the training and validation loss values are relatively high. The training loss started at approximately 0.20 while the validation loss started at approximately 0.25. Noticeably, both loss curves are stabilized at low values, approximately 0.045 and 0.02 for training loss and validation loss, respectively. Since both values are close to zero means the model provides an excellent prediction

6.2. Confusion matrix

The confusion matrix, often referred to as the error matrix, assists in the easier interpretation of results through data visualization. It summarizes the performance of the proposed LSTM model in distinguishing between the normal (Benign) and the malicious XSS traffic (Attack). The matrix is structured as in Figure 6.

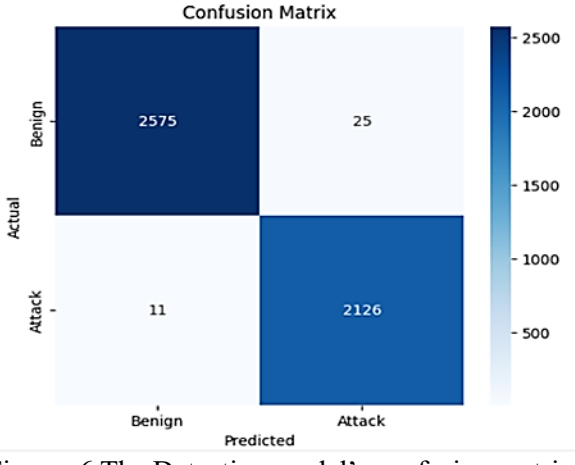

Figure: 6 The Detection model's confusion matrix

The y-axis represents the actual class, while the predicted is displayed on the x-axis. The total number of benign traffic was 2600. The model correctly classified 2575 requests as normal traffic, while only 25 requests were wrongly classified as attacks,

http://www.doi.org/10.62341/istj-vol38-2-hi47

which is considered a low false alarm rate. Meanwhile, the total requests from a malicious XSS traffic attack were 2137; only 11 requests were misclassified as benign. Overall, the model is good at recognizing benign traffic with about 96%, and about 0.5% of attacks, which is 11 out of 2,137 are misclassified. These percentages provide evidence that
the proposed LSTM model is reliable and suitable for real-world XSS detection.

6.3. Breakdown of Model Performance by Class: Precision, Recall, and F1-Score

From the confusion matrix, a set of typical arithmetic systems of measurement was plotted to evaluate the model's ability in distinguishing between normal traffic and XSS attack for each class. Similar to [13] The authors considered three common parameters to interpret results, namely precision, recall, and the F1-score. Precision is the rate at which the samples classified into a given class are really a true example of that class. Whereas the Recall (also known as sensitivity or true positive rate) implies how many of the truly positive samples are detected by the model, while the F1-score gives an overall balanced measure between precision and recall as its harmonic mean of precision and recall.

As presented in Figure 7, the proposed XSS model reached exceptional results for all classes. As shown, in the benign class, the model reached a precision of 99.6%, a recall of 99.0%, and 99.3% for the F1-score. Whereas for the XSS attack class, the model attained a precision of 98.8%, a recall of 99.5%, and 99.2% for the F1-score. Consequently, these results testify that the model is able to detect XSS attacks using the proposed model in real-time.

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

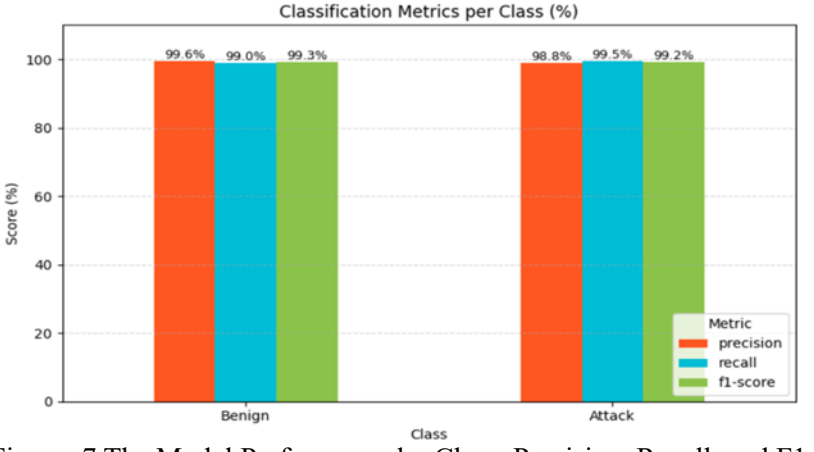

Figure: 7 The Model Performance by Class: Precision, Recall, and F1-Score

6.4. Integrating the XSS model with the ModSecurity firewall

This analysis aims to study the proposed model's impact on the system performance and resource consumption. The assessment was based on several key performance metrics such as throughput, average latency, 95th percentile latency, CPU load, and memory usage. Comparing these metrics across both scenarios allows the study to quantify the computational cost of the integration process, while ensuring that improvements in detection capabilities have no significant effects on network performance or resources. All performance tests were obtained by using the custom WAF's performance in Python (performance_test.py), which sends concurrent HTTP requests to the web application under test, logs response times, computes throughput and latency, and produces numerical reports of the firewall's activity before and after integration. The metrics that cut across the performance evaluation of the WAF are used by the generate_report() function to produce a report showing how the baseline state differed from that of AI integration. The report contains averages, P95 latency, throughput, error rate, as well as fictitious security effectiveness indicators. In addition, the CPU load and memory usage were monitored by using the *htop* tool in the Ubuntu environment, as shown in Figure 8.

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

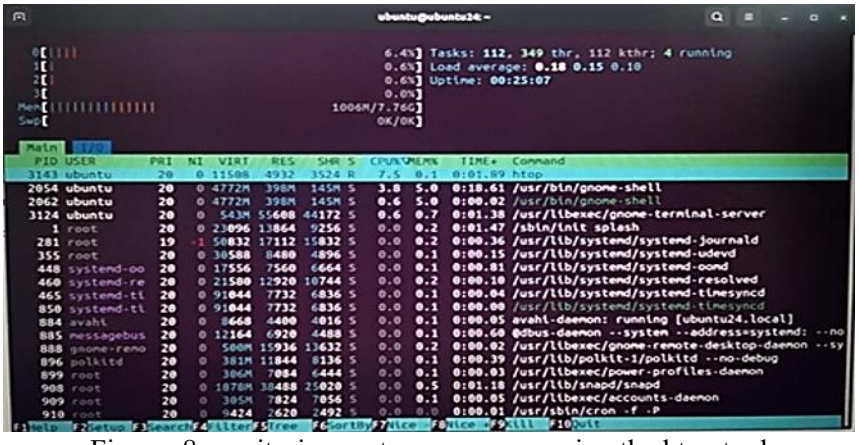

Figure: 8 monitoring system resources using the htop tool

The Performance evaluation before and after integration is summarized in Table 2. The table compares system/application performance before and after integrating a Machine Learning-based Web Application Firewall (ML-WAF).

Table 2. Performance evaluation (Before and After Integration)

Metric	*Before Integration (Baseline)*	*After ML-WAF Integration*	*Average Before*	*Average After*	*Change (%)*
Throughput	500-800 req/s	430-660 req/s	(500+800)/2 = 650	(430+660)/2 = 545	≈-16%
CPU Usage	10-15%	25-45%	(10+15)/2 = 12.5	(25+45)/2 = 35.0	≈+180%
Avg Latency	2-5 ms	7-17 ms	(2+5)/2 = 3.5	(7+17)/2 = 12.0	≈+240%
95th Percentile Latency	8-12 ms	20-40 ms	(8+12)/2 = 10	(20+40)/2 = 30	≈+200%
Memory Usage	50-80 MB	190-300 MB	(50+80)/2 = 65	(190+300)/2 = 275	≈+275%

The table focuses on evaluating five key system metrics measured under a baseline configuration when there is no ML-WAF and after the integration of an ML-based system with the ML-WAF. In each

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

row, the table reports the observed range, the calculated arithmetic mean of the range endpoints labelled Average Before/After, and the resulting relative change expressed as a percentage. Throughput that represents the requests per second is decreased from a baseline average of 650 req/s to 545 req/s post-integration, which indicates about a 16% reduction. This indicates that the additional processing introduced by the enhanced ML-WAF bounds the request-handling capacity. However, this could be a minor percentage to protect servers against XSS attacks. By looking at the second row, CPU usage increased from 12.5% to 35%, approximately a +180% of an increase, reflecting the computational cost of feature extraction, model inference, and the additional rule for matching and testing each request. Similarly, the memory usage increased due to the extra storage required by the ML model parameters, feature vectors, rule caches, and associated data structures required for real-time inference.

Generally, the enhanced WAF is considered an improved model to provide better security in detecting XSS threats, even though it introduces significant overhead in throughput, latency, and resource consumption. The column chart presented in Figure 9 shows a comparative analysis of firewall performance before and after integrating Machine Learning to enhance the WAF performance.

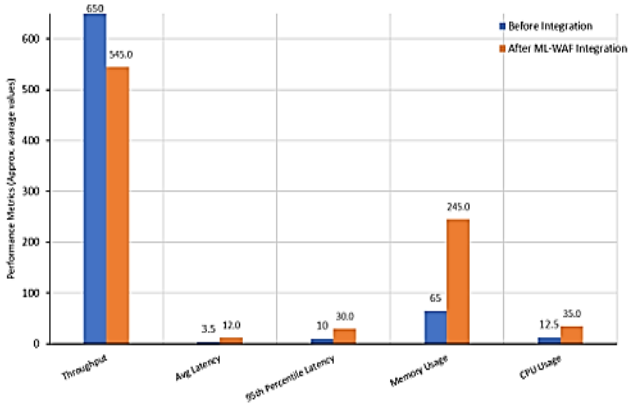

Figure: 9 A firewall performance comparison before and after the ML-WAF integration

http://www.doi.org/10.62341/istj-vol38-2-hi47

Despite the increase in resource utilization, the experimental results reveal that the enhanced WAF based on the LSTM model provides a valuable trade-off between performance and security. The additional computational cost is justified by the improved capability of the firewall to detect sophisticated threats of XSS attacks in real time. As stated earlier, the traditional firewalls mainly rely on static signatures and predefined rules; the proposed enhanced WAF using ML can perceptively classify unfamiliar and obfuscated patterns with higher accuracy against XSS attacks. Overall, the results indicate that integrating artificial intelligence into the traditional firewall architecture can enhance the security performance and increase the WAF capabilities in detecting threats, specifically XSS. In addition, the results show acceptable working efficiency with an accepted trade-off in performance. Moreover, the proposed model successfully balances real-time attack detection and system performance, making it a reliable solution for protecting modern web applications against XSS threats.

Conclusion

There is no doubt that web applications are essential for providing several online services such as e-banking, e-learning, and governmental services. Securing sensitive private and organizational information has become a serious fear due to the increasing number of web application attacks, principally Cross-Site Scripting (XSS) attacks. This study intended to enhance Web Application Firewall (WAF) systems by integrating a trained Long Short-Term Memory (LSTM) model for real-time XSS attack detection. The experimental results prove that the enhanced WAF using the LSTM model implicitly enhanced the detection capabilities compared with traditional WAFs that work basically on signature-based methods. The proposed enhanced model combines the traditional WAF architecture with an extra layer named XSSLayer based on AI to enhance the detection ability. The obtained results showed that the enhanced Web Application Firewall (WAF), with an integrated Long Short-Term Memory (LSTM) can classify malicious scripts within web requests, whereas the training

http://www.doi.org/10.62341/istj-vol38-2-hi47

accuracy reached approximately 98.0% and the validation accuracy about 99.4%. In addition, the results showed a validation loss of about 0.02, that indicate a high predictive reliability and minimal prediction error. The confusion matrix analysis showed only 25 false positives out of 2,600 benign requests and 11 false negatives out of 2,137 attack instances, which indicates the model's robust discriminative capacity. Furthermore, the obtained results of precision, recall, and the F1-score indicate that the model shows an exceptional capability, with nearly perfect performance in detecting XSS attacks using the LSTM architecture in real-time. However, integrating the XSS model with the ModSecurity firewall introduced additional computational overhead and slightly increased reply time, memory, and CPU usage. Overall, this study establishes that embedding deep learning-based classification into conventional firewall architectures yields a reliable, real-time defensive mechanism against XSS attacks, positioning the proposed system as a practical advancement for securing modern web applications. With extra efforts for minimizing response delay, memory, and CPU usage

Author Biography

Haithem Ali Alghazzawi holds a Master's degree in a computing-related field and is affiliated with the School of Applied Sciences and Engineering at The Libyan Academy, Libya. His academic interests include information technology, intelligent systems, and applied engineering solutions. He actively participates in collaborative research focusing on emerging technologies and their practical applications.

Dr. Mohamed Elbeshti is a faculty member in the Department of Information Technology at the University of Zawia, Libya. His research interests include information technology, cybersecurity, and modern computing solutions. He has contributed to several academic and research activities in the field of information systems and network technologies.

Dr. Abdalrahman Alfagi is affiliated with the Department of Information Technology at the University of Zawia, Libya. He is a

http://www.doi.org/10.62341/istj-vol38-2-hi47

remote Adjunct Instructor at the American University of Afghanistan (AUAF). His research interests include web application security, artificial intelligence, machine learning, and cybersecurity. His work focuses on developing intelligent solutions for enhancing the security and performance of modern information systems.

Dr. Sami Naji is Co-Founder of Eurobillr and an AI Researcher in Gent, Belgium. His academic and research interests include machine learning, computer vision, and natural language processing. He has been involved in interdisciplinary research and international academic collaborations in technology-related fields.

Authors Collaboration

This research paper is the result of a collaborative effort between researchers from The Libyan Academy, the University of Zawia, and the American University of Afghanistan. The authors worked together in developing the research concept, conducting the literature review, designing the methodology, analyzing the results, and preparing the manuscript. This collaboration brought together expertise from different academic institutions and disciplines, contributing to the quality and significance of the research outcomes.

References

[1]. Kaur, J., U. Garg, and G. Bathla, *Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review.* Artificial Intelligence Review, 2023. **56**(11): p. 12725-12769.

[2]. Aslan, Ö., et al., *A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions.* Electronics, 2023. **12**(6): p. 1333.

[3]. Verizon, *2026 Data Breach Investigations Report*. 2026, Verizon Enterprise Solutions.

[4]. Stallings, W., *Cryptography and Network Security Principles and Practice*. 8 ed. 2023: Pearson Education Limited.

Real-Time XSS attack detection based on an enhanced web application firewall through Deep Learning and LSTM-Integrated MoD Framework

http://www.doi.org/10.62341/istj-vol38-2-hi47

[5]. Verizon, *2025 Data Breach Investigations Report*. 2025, Verizon Enterprise Solutions.
[6]. Tanveer, F., et al., *Towards Secure APIs: A Survey on RESTful API Vulnerability Detection.* Computers, Materials, & Continua, 2025. **84**(3): p. 4223.
[7]. Decker, P. and F. Hantke, *VDPCollect: Vulnerability Disclosure Programs as a Complement to Web Security Measurements.* 2026.
[8]. Alghazzawi, H.A., *Enhancing Web Application Firewall Security with a Multi-Layered Deep Learning Approach for DDoS, XSS, and SQL Injection Detection.* 2025.
[9]. Babaey, V. and A. Ravindran. *GenXSS: An AI-driven framework for automated detection of XSS attacks in WAFs*. in *SoutheastCon 2025*. 2025. IEEE.
[10]. Hapugala, H., et al., *An analysis of xss vulnerabilities and prevention of xss attacks in web applications.* An analysis of xss vulnerabilities and prevention of xss attacks in web applications, 2023.
[11]. Ni, Y., Z. Li, and X. Li, *Security Analysis of Web Applications Based on Gruyere.* ArXiv, 2025. abs/2509.14706.
[12]. Alenzi, K.F. and O.A.B. Abbase, *A defensive framework for reflected XSS in client-side applications.* Journal of Web Engineering, 2022. **21**(7): p. 2209-2229.
[13]. Alghazzawi, H.A., M. Elbeshti, and A. Alfagi, *An enhanced Web Application Firewall Security for real-time DDOS detection using Deep Learning and integrating LSTM models into Modsecurity Firewall.* Academy Journal for Basic and Applied Sciences, 2025. **7**(2).
[14]. Diekmann, C., L. Hupel, and G. Carle. *Semantics-preserving simplification of real-world firewall rule sets*. in *International Symposium on Formal Methods*. 2015. Springer.
[15]. Goss, K. and W. Jiang, *Distributing and Obfuscating Firewalls via Oblivious Bloom Filter Evaluation.* arXiv preprint arXiv:1810.01571, 2018.

http://www.doi.org/10.62341/istj-vol38-2-hi47

[16]. Krishna, V.A. and T.A.A. Victoire, *Analysis of Firewall Policy Rules a Comparative Study.* Analysis, 2013. **6**(5): p. 112-118.

[17]. Zhaikhan, A., et al., *Safeguarding the IoT from malware epidemics: A percolation theory approach.* IEEE Internet of Things Journal, 2020. **8**(7): p. 6039-6052.

[18]. Tekerek, A. and O.F. Bay, *Design and implementation of an artificial intelligence-based web application firewall model.* Neural Network World, 2019(4).

[19]. Kumar, H. *Securing web application using web application firewall (waf) and machine learning*. in *2023 First International Conference on Advances in Electrical, Electronics and Computational Intelligence (ICAEECI)*. 2023. IEEE.

[20]. Trinh, C.-V., et al. *An Efficient Machine Learning-Based Web Application Firewall with Deep Automated Pattern Categorization*. 2023. Singapore: Springer Nature Singapore.

[21]. Nayar, V., et al. *Optimizing Real-Time Performance in ML-Based Application Layer Firewalls*. 2024. Singapore: Springer Nature Singapore.

[22]. Council, S.N.R. *CSIC 2010 HTTP Dataset*. [Dataset] 2010; Available from: https://gitlab.fing.edu.uy/gsi/web-application-attacks-datasets/-/tree/master/csic_2010?ref_type=heads.