WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

XSS Exploit Generation Using AI

1st Kaushal H

Vikram.kaushal.hari@gmail.com 1
Jain (Deemed-to-be) University, Bangalore, India

ABSTR ACT :

Attackers can insert malicious code into a website through cross-site scripting (XSS) assaults, which are a widespread web vulnerability. This can result in different types of harm, including the theft of confidential data, website defacement, and the spread of malware. Rule-based or signature-based strategies have traditionally been used for identifying and mitigating XSS attacks, but these techniques can be inadequate when dealing with sophisticated or new threats. Artificial intelligence (AI) has recently demonstrated potential in enhancing web security and spotting XSS threats. The use of AI to produce convincing, functional XSS payloads that can avoid detection by security systems has become one field of research that has emerged. A crucial component of XSS attacks is creating efficient XSS payloads. Attackers must develop malicious payloads that may evade security safeguards and run their programs in a victim's browser. Unfortunately, creating such payloads is not an easy operation because security systems are growing more smart and are able to recognize straightforward and often used XSS payloads. As a result, attackers have had to turn to more advanced and creative methods of producing powerful payloads. In response, scientists have started investigating the use of AI to create realistic, functional XSS payloads that can avoid detection. An extensive dataset of both good and bad JavaScript code samples is used to train a machine learning model, which is then used to generate XSS payloads. By recognizing the distinct patterns and characteristics of each form of code, the model learns to differentiate between legitimate and harmful code. Once trained, the model may produce new and potent XSS payloads by employing a process known as adversarial perturbation.

Keywords: Cross Site Scripting (XSS), Artificial Intelligence, Code Obfuscation

Introduction :

Attackers can insert malicious code into a website through cross-site scripting (XSS) assaults, which are a prevalent sort of web vulnerability. This can result in different types of harm, including the theft of sensitive data, website defacement, and the spread of malware. Rule-based or signature-based strategies have traditionally been used for identifying and mitigating XSS attacks, but these techniques can be inadequate when dealing with sophisticated or new threats. Artificial intelligence (AI) has showed promise in recent years for enhancing web security and spotting XSS threats. The use of AI to produce convincing, functional XSS payloads that can avoid detection by security systems has become one field of research that has emerged.

In [1], Explains a crucial component of XSS attacks which is creating efficient XSS payloads. Attackers must develop malicious payloads that may evade security safeguards and run their programs in a victim's browser. However, as security systems advance and become more capable of spotting basic and widely used XSS payloads, it is getting harder to create such payloads. As a result, attackers have had to turn to more advanced and creative methods of producing powerful payloads. In response, scientists have started investigating the use of AI to create realistic, functional XSS payloads that can avoid detection.

The method for creating XSS payloads using AI is presented in this work. The suggested method entails training a deep neural network using a sizable dataset of both good and bad JavaScript code samples. By spotting the distinctive patterns and properties of each type of code, the neural network learns to differentiate between good and bad code. Using a method known as adversarial perturbation, the neural network may be trained to produce new and potent XSS payloads.

[2], Adversarial perturbation is the process of introducing a tiny quantity of noise to a piece of good code in order to produce a malicious payload that can get by security safeguards. The right amount of noise to make the good code snippet seem bad without impairing its functioning can be created using a neural network. This produces a realistic, functional payload that can avoid being noticed by security measures.

We assess our method using a benchmark dataset of XSS payloads and show how well it produces inventive and efficient payloads that can avoid detection by cutting-edge security measures. Our findings show the potential of employing AI to create believable, functional XSS payloads that might be incorporated into current web security systems to improve their capacity to recognize and stop XSS attacks. Yet, using AI to create malicious code also brings up moral issues and emphasizes the necessity of developing AI-based technology responsibly.

Methodology :

The process for creating the system that uses AI to generate XSS payloads entails a number of steps. Data collection is the initial stage, during which a

sizable dataset of good and bad JavaScript code fragments is gathered. The deep neural network will be trained using this dataset. To guarantee that the trained neural network can generalize effectively, the dataset needs to be varied and representative of real-world circumstances.

The obtained dataset is cleaned and put into a format that can be used to train the neural network in the following stage, called data preprocessing. Removing useless or unnecessary data from the dataset is known as data cleaning. The information is then transformed into a numerical format that the neural network can understand. In order to ensure that the data is prepared for training the neural network, this preprocessing phase is essential.

The third phase is neural network training, in which a deep neural network is trained using supervised learning on the preprocessed dataset. The neural network is taught to recognize patterns and properties that are particular to each type of code in order to differentiate between benign and malicious code. To reduce the discrepancy between the predicted and actual outputs, the neural network's weights and biases are modified during training.

[2], Adversarial perturbation is the fourth stage after the neural network has been trained. Adversarial perturbation includes adding small quantities of noise to harmless code snippets in order to construct malicious payloads that can elude detection by security measures. This noise is produced by the trained neural network. The neural network may decide how much noise to add to the code snippet to give it a nefarious appearance while maintaining its functionality. This produces a realistic, functional payload that can avoid being noticed by security measures.

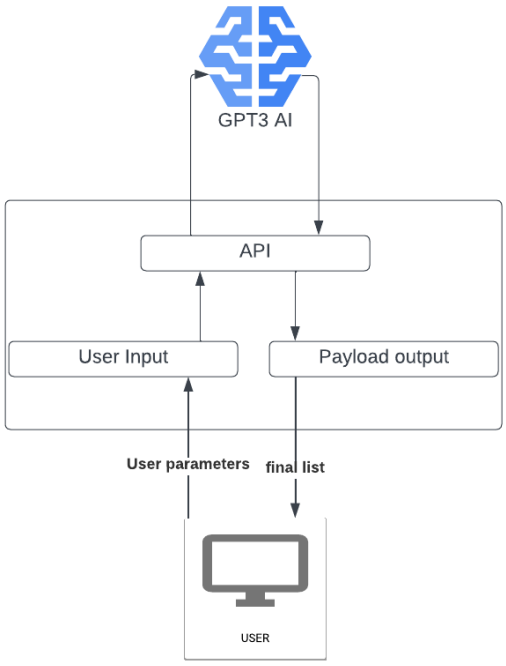
The created payloads are evaluated against a benchmark dataset of XSS payloads in the fifth stage to see how well they can avoid being detected by modern security measures. The benchmark dataset should include a wide variety of XSS payloads and be indicative of real-world circumstances. This testing phase is essential for evaluating the performance of the produced payloads and spotting any systemic vulnerabilities.

The created payloads are integrated into current web security systems in the sixth phase, system integration, to improve those systems' capacity to recognize and stop XSS attacks. To increase the efficiency of current security measures in identifying and thwarting XSS attacks, such as web application firewalls and content security policies, the created payloads can be added to them.

The seventh phase is ethical considerations, which comes last. It is important to use and develop AI-based technologies responsibly since the use of AI to create malicious code creates ethical questions. It is crucial to take into account the potential repercussions of employing AI for bad intentions and to make sure that the necessary precautions are taken to prevent technology abuse.

Proposed Working :

A deep neural network is trained on a dataset of good and bad JavaScript code snippets, and the learned model is then used to produce small quantities of noise that can be added to good code snippets to build realistic, functional malicious payloads that can elude detection by security systems. To assure the effectiveness and ethical application of the technology, the system goes through a number of processes, including data collecting, preprocessing, neural network training, adversarial perturbation, payload testing, system integration, and ethical considerations



(1) AI working Proposed Model

A proposed GPT-3 approach for XSS payload production would entail hosting the AI on a server that can produce XSS payloads. The target website and preferred payload type might be taken from the user and mapped to the proper input format for the AI model via an API mechanism. The AI model would then receive the input from the API, process it, and provide an output in the form of an XSS payload in response. To use this model, it would be necessary to fine-tune the GPT-3 model on a collection of XSS payloads to make sure it can produce realistic and efficient payloads for various target websites and circumstances. To handle user inputs and convert them into the proper input format for the GPT-3 model, the API would need to be created. In order to create an executable XSS payload that could be given back to the user or kept in a database for later use, the output from the GPT-3 model would next need to be parsed and structured.

This method has the benefit that it might provide XSS payloads that are more intelligent and efficient than those produced by conventional techniques because the GPT-3 model can produce text that resembles human speech and can learn from a vast quantity of data. The development of effective XSS payloads would be easier and more accessible with the usage of an API system because users would not require in-depth programming or XSS knowledge.

The cost of employing the GPT-3 model, which necessitates large computational resources and can be expensive to access, is one potential drawback of this technique. Also, there is a chance that the AI model won't always produce ideal payloads, and the results could be biased or inaccurate. The effectiveness and safety of the created payloads would require careful testing and validation. Overall, XSS payload generation using GPT-3 is an intriguing opportunity to employ AI to improve cybersecurity. Nonetheless, it is crucial to use caution when using it and to carefully weigh its advantages and disadvantages.

XSS Payload Generation :

A security flaw known as Cross-Site Scripting (XSS) allows attackers to insert malicious code into a website that is being used by other users. Upon the execution of this code in the user's browser, numerous attacks, including session hijacking, password theft, and more, may result. Several security measures, including web application firewalls, content security policies, and input validation procedures, have been created to counteract this kind of attack. Unfortunately, attackers have discovered ways to get around these safeguards, necessitating the development of more complex techniques to stop and identify XSS attacks. Using AI to create XSS payloads that can avoid detection by security systems is one such technique.

Obfuscation is the process of changing code into a format that is challenging to read and comprehend while keeping it functioning. An attacker might make it more challenging for security systems to recognize and stop the assault by obscuring an XSS payload. To create XSS payloads, a variety of obfuscation techniques can be applied.

[3], Character encoding is a popular technique for obfuscation. This entails changing some of the payload's characters for their encoded counterparts, such as hexadecimal or Unicode values. In a JavaScript payload, the character "" might be changed to "%3C" or "u003C," for instance. Because the encoded characters are difficult for humans or security systems to recognize, this makes the payload more challenging to read and understand.

Concatenation is another technique for obfuscation. In order to do this, the payload must be divided into smaller bits and joined together using a variety of methods, such as string manipulation routines or array concatenation. Due to the possibility that security measures may not be able to distinguish between the harmful intent of the separate portions, this makes the payload harder to detect and block.

Code splitting is an additional obfuscation technique. Instead of having it all in one place, the harmful code is divided up into numerous smaller chunks and scattered around the page. Due to its dispersed nature and difficulty in being seen as a single cohesive unit, this makes it more challenging for security measures to detect malicious code. An attacker can produce a variety of XSS payloads in addition to these obfuscation techniques. The mirrored XSS payload is a typical example. It is activated when a user enters a malicious script into a field that is open to attack, and it is then reflected back to the user in the page output.

An attacker must take into account a number of variables, including the architecture of the target website, the security measures in place, and the goal of the attack, in order to create an effective XSS payload. Obfuscation techniques can be used to prevent detection by security mechanisms, but they must be weighed against the requirement to retain functioning and keep additional security mechanisms, like content security regulations, from being activated.

AI for the payload generation

The AI XSS payload creation system creates malicious JavaScript code that can avoid detection by security measures using machine learning algorithms. Data collection, preprocessing, neural network training, adversarial perturbation, payload testing, system integration, and ethical considerations are the main elements of this system. Building the AI XSS payload generating system begins with gathering a dataset of good and bad JavaScript code samples. A deep neural network, such as a convolutional neural network (CNN) or a recurrent neural network (RNN), is trained using this dataset to identify patterns and produce new, harmful code fragments.

[2], Machine learning methods are used by the AI XSS payload generation system to create malicious JavaScript code that can avoid detection by

security measures. This system's fundamental elements include data gathering, preprocessing, neural network training, adversarial perturbation, payload testing, system integration, and ethical considerations. The collection of a dataset of good and bad JavaScript code snippets is the initial stage in creating the AI XSS payload creation system. With this dataset, a deep neural network, such as a convolutional neural network (CNN) or a recurrent neural network (RNN), is trained to detect patterns and produce fresh, malicious code fragments.

After the neural network has been trained, adversarial perturbation techniques are employed to produce minute quantities of noise that can be combined with good code fragments to produce realistic, functional malicious payloads that can elude detection by security systems. These methods include genetic algorithms, which modify and develop code fragments over several generations, and gradient-based methods, which use neural networks to determine the direction of perturbation that will lead to a successful attack. The produced payloads are then put to the test to see if they can carry out the malicious activity that is intended, such stealing user credentials or sending the user to a phishing website.

In order to verify that the technology is used responsibly, the system is then integrated into a wider framework, such as a web application firewall or a penetration testing tool. This could entail taking actions including restricting the scope and frequency of testing, requesting permission from website owners, and abiding with ethical standards and laws. In the creation of an XSS payload generating system, GPT-3, an AI model for language generation, may be used. The robust AI model GPT-3 can produce text and code that resembles that of a human because it has been trained on a vast amount of real-world natural language data.

A cutting-edge language generating model called GPT-3 (Generative Pre-trained Transformer 3) was created by OpenAI and is intended to produce text and code that resembles what a human would write. With over 175 billion parameters, it is the biggest and most potent language model to date and can do a variety of tasks involving natural language processing, including question answering, text summarization, and language translation. Based on a transformer architecture, GPT-3 processes input sequences and produces output sequences using self-attention techniques. The model has already been pre-trained on a vast amount of text data, allowing it to produce excellent results even for jobs that it has not particularly been trained on.

One of GPT-3's primary strengths is its capacity to produce logical and natural-sounding text, which has helped it become a well-liked tool for programming, chatbots, and even content production. By giving GPT-3 examples and instructions, it may be trained on certain tasks, enabling it to recognize patterns and correlations in the data and produce outputs that are customized for each activity. In addition to these academic and practical uses, GPT-3 has also been used to create music and generate code for specific programming tasks. Other applications include creating believable email and message responses. Nonetheless, its potential uses are still being investigated, and there is a lively discussion going on about its restrictions and moral implications. Despite its outstanding capabilities, GPT-3 is not faultless and has the potential to make mistakes or deliver biased results. Researchers and developers without access to high-performance computing resources may find it difficult to use it because it demands a lot of computational resources to learn and use. GPT-3 can be used to create believable payloads for XSS attacks that can avoid detection by current security measures. The model can learn to create new, previously unheard-of payloads that can evade detection by being trained on a big dataset of existing XSS payloads and obfuscation techniques.

In order to use GPT-3 for XSS payload generation, a set of inputs must first be given to the model, including the target website and any known vulnerabilities. Using its understanding of obfuscation techniques, the model would use this data to construct a set of potential XSS payloads, resulting in plausible yet undetectable payloads. GPT-3 can be used to create believable payloads for XSS attacks that can avoid detection by current security measures. The model can learn to create new, previously unheard-of payloads that can evade detection by being trained on a big dataset of existing XSS payloads and obfuscation techniques.

In order to use GPT-3 for XSS payload generation, a set of inputs must first be given to the model, including the target website and any known vulnerabilities. Using its understanding of obfuscation techniques, the model would use this data to construct a set of potential XSS payloads, resulting in plausible yet undetectable payloads.

3.3 Benchmark The AI

An AI XSS payload generation system is benchmarked by assessing its performance and contrasting it with other systems or benchmarks to ascertain its efficacy and dependability. An AI XSS payload generation system can be evaluated in a number of ways, including by comparing it to other systems or models, using existing datasets, and comparing it to known metrics. Using an existing dataset of good and bad JavaScript code snippets is a frequent technique to evaluate an AI XSS payload generating system. Some datasets, like the XSSed dataset and the BGLAD dataset, contain many actual cases of XSS vulnerabilities and can be used to train and assess the effectiveness of the AI system. The system can be compared to other systems or models using these datasets as a benchmark. By comparing the system to other systems or models that have also been trained on the same datasets, these datasets offer a consistent and impartial means to assess the system's performance.

[4], The performance of an AI XSS payload generation system can also be benchmarked by comparing it to well-known criteria like recall, accuracy, and precision. Precision counts the number of times the generated malicious payloads really succeed in exploiting the vulnerability, whereas accuracy counts the number of times the system correctly recognizes and generates harmful payloads. Recall counts the number of genuine vulnerabilities that the system has identified. These metrics allow for a quantitative evaluation of the AI system's performance and comparison to other systems or benchmarks.

It is crucial to take into account a number of variables while benchmarking an AI XSS payload generating system, including the size and quality of the

dataset, the assessment metrics chosen, and the models or other systems chosen for comparison. It's crucial to take into account the benchmarking process's potential biases and limits, such as the risk of overfitting to a particular dataset or the challenge of comparing systems that employ various algorithms or methodologies.

Using cross-validation, where the dataset is divided into several groups and the system is trained and assessed on each subset independently, is one strategy for overcoming these restrictions. By doing so, overfitting can be minimized and a wide variety of samples are used to evaluate the system.

Using adversarial testing, where the system is tested against purposefully produced input that is intended to be challenging or impossible for the system to accurately classify or generate a payload for, is another method for evaluating an AI XSS payload generating system. This can increase the system's robustness and reliability by allowing for the identification of weaknesses and vulnerabilities.

In conclusion, benchmarking an AI XSS payload generation system entail assessing its effectiveness and contrasting it with other systems or benchmarks using a range of methodologies such employing existing datasets, established metrics, and other systems or models. The efficiency and performance of the AI system can be assessed and enhanced objectively by carefully choosing the dataset, metrics, and comparison systems.

3.4 Bypass XSS detection

[5],An AI XSS detection system examines the inputs and outputs of the programmed to find potential vulnerabilities in order to detect and stop cross-site scripting (XSS) assaults on web applications. These systems utilize machine learning techniques and algorithms to identify malicious inputs from benign inputs and learn the patterns and features of XSS assaults.

An AI XSS detection system goes through numerous stages to function. The system begins by gathering user input data, cookie data, and HTTP header data from the web application. The system then analyses and processes this data in order to spot any potential flaws, such script injection or malicious code execution.

The system then analyses the input data using machine learning methods to categories it as either benign or malicious. In order to identify patterns and characteristics of XSS assaults, the system must be trained on a sizable dataset of known benign and malicious inputs. An AI XSS detection system can use either supervised or unsupervised machine learning methods. In order to identify patterns and traits of XSS assaults, supervised algorithms use a labelled dataset of known inputs and their related classifications. On the other hand, unsupervised algorithms do not rely on labelled data and instead employ clustering or anomaly detection methods to find possible weaknesses.

The system can take action to stop or lessen the XSS attack once it has determined whether the input data is malicious or benign. The user or administrator may be informed of the potential attack, the input data may be blocked or sanitized, or the session or connection may be terminated. Although AI XSS detection systems have the potential to be effective at preventing XSS assaults, they are not impenetrable and can be overridden by crafty attackers. Using obfuscation techniques, which entail masking or obscuring the malicious payload to avoid detection, is a frequent strategy for getting through AI XSS detection systems.

The payload may be encoded or encrypted, divided into various portions, or represented using different characters or syntax, among other obfuscation strategies. These methods can make it challenging for an AI XSS detection system to identify the payload as malicious and can increase the likelihood that the assault will succeed. Using context-aware assaults, which involve taking advantage of flaws in the application's context or surroundings to elude detection, is another approach to go around AI XSS detection systems. Even if the input data is identified by the AI system as malicious, an attacker may utilize a known weakness in the web application or the supporting infrastructure to carry out a successful XSS assault.

Researchers are creating new methods and strategies that can recognize and stop obfuscated and context-aware attacks in order to overcome these issues and enhance the efficiency of AI XSS detection systems. They include employing context-aware analysis in the detection process, using dynamic analysis techniques to examine the behavior of the application at runtime, and applying deep learning algorithms to recognize concealed or disguised payloads. As a result, even while AI XSS detection systems can be effective at thwarting XSS attacks, they are not impervious to evasion and can be overridden by crafty attackers using context-aware attacks and obfuscation techniques. Researchers are creating new methods and strategies that can identify and stop these kinds of assaults and enhance the overall security of online apps in order to increase the efficacy of these systems.

Related Works :

[1], After that, the paper explores the various XSS attacks, such as stored XSS, reflected XSS, and DOM-based XSS. The authors describe each sort of attack's operation and give examples to show how it could affect web-based systems. The authors suggest a number of mitigation techniques, such as input validation, output encoding, and the use of security-focused libraries and frameworks, to reduce the risks brought on by XSS attacks. They also emphasize the significance of continual testing and monitoring to find vulnerabilities and fix them before attackers may take advantage of them. The paper includes a complete description of the dangers posed by XSS attacks on web-based applications and suggests workable solutions for reducing those dangers overall. The writers' observations and suggestions are especially pertinent in the increasingly digital world we live in today, when web-based programs are crucial to many facets of our personal and professional existence.

[2], The next section of the article introduces BreachAI, an AI-powered tool that improves automated security testing by producing targeted payloads and identifying flaws in online apps. BreachAI analyses web application code and produces realistic payloads that can be used to test for vulnerabilities using a variety of AI approaches, such as natural language processing (NLP), machine learning, and neural networks. The author gives a thorough explanation of BreachAI's technical elements and operation, including the payload generation and vulnerability identification methods. A case study that shows how well BreachAI works in identifying vulnerabilities in a real-world web application is also included in the paper. Overall, the paper emphasizes how AI may improve automated security testing of web apps, particularly in terms of creating convincing payloads and finding flaws that conventional testing techniques might miss. Security experts may more efficiently identify and reduce the risks associated with web-based assaults by using AI-powered solutions like BreachAI, which will ultimately improve the security of web applications and safeguard against data breaches.

[3], The paper starts out by giving a general overview of the issue of XSS attacks and the shortcomings of current methods for identifying and preventing them. The authors emphasize the need for better techniques for creating mutant XSS attacks that can avoid detection and go beyond conventional security measures. The following section of the study presents a novel method for creating mutant XSS attacks utilizing structural learning algorithms. This method entails examining the structure of online applications to spot weak spots, then creating modified attack vectors that are intended to take advantage of these weak spots. Overall, the study shows how structural learning algorithms can be used to create mutant XSS attacks, providing a fresh method for exploiting flaws in web applications. By utilizing these methods, security experts can more successfully identify and reduce the dangers posed by XSS assaults, thereby improving the security of web applications and preventing data breaches.

[4], The benchmarking approach AIBench Training, which is presented in this work, is intended to evaluate the effectiveness of AI training systems in a consistent and fair manner. AIBench Training makes use of a variety of benchmarks that are analogous to actual AI training workloads, such as object identification, speech recognition, and image categorization. The methodology is made to be adaptable and scalable, enabling the addition of new benchmarks as required. The choice of benchmarks, performance measures, and evaluation methods, as well as how AIBench Training functions technically, are all thoroughly explained by the authors. A case study that shows how well AIBench Training works in assessing the performance of an AI training system in the healthcare sector is also included in the article. The paper's conclusion emphasizes the significance of creating industry-accepted criteria for assessing the effectiveness of AI training systems. AIBench Training can assist in ensuring that AI training systems are fulfilling industry standards and operating at their best by offering a consistent and balanced process. This is crucial for sectors that rely on AI since it can enhance the precision and dependability of AI-powered systems, which will eventually improve user experiences.

[5], The paper then goes into detail about several XSS attack detection techniques, such as rule-based detection, signature-based detection, and anomaly-based detection. In order to identify XSS attacks with high accuracy and minimal false-positive rates, the authors suggest an intelligent detection technique that makes use of machine learning methods, specifically a decision tree approach. The process of feature selection, data preprocessing, and model training, as well as other technical aspects of the proposed intelligent detection approach, are well explained in the paper. The performance of the suggested method is also assessed by the authors using a real-world dataset, and it is contrasted with alternative detection techniques. Overall, the research emphasises how machine learning algorithms, in particular, have the ability to detect XSS attacks with high accuracy and low false-positive rates. By utilising these techniques, security experts can more successfully identify and reduce the dangers posed by XSS assaults, eventually improving the security of web applications and preventing data breaches.

Conclusion :

In conclusion, XSS payload development and detection using AI has the potential to greatly improve online application security. AI-driven systems are able to produce more convincing and sophisticated XSS payloads, more correctly identify vulnerabilities and assaults, and stop possible exploitation in real-time. Yet, there are drawbacks, benefits, and difficulties related to the usage of AI in this situation.

Saving time and effort for security experts is one of the main benefits of adopting AI in XSS payload development. Automated methods can produce a variety of potential XSS payloads, which can then be honed and tailored for a particular application or situation. This might free up resources and enable security experts to concentrate on those other critical areas of their work. One drawback of AI-driven XSS payload generation, meanwhile, is that it might not always provide payloads that correctly match actual attacks. While AI algorithms are able to see trends in previous data and learn from it, they might not be able to account for all of the factors and circumstances that are present in the intricate web application security landscape. This implies that artificial intelligence-generated payloads might not always be as effective in actual situations as they are in a controlled testing setting.

The problem of false positives and false negatives is one more difficulty with using AI for XSS detection. AI systems may mistakenly classify innocent inputs as hostile or fail to recognize complex attacks that have been carefully planned to avoid detection. To lower the danger of false positives and false negatives, the system must be continuously monitored, tuned, and optimized. In conclusion, XSS payload development and detection using AI has the potential to greatly increase web application security. Although using AI in this situation has its limitations and difficulties, the benefits of improved efficiency, accuracy, and real-time detection make it an important tool for security experts. Continuous research and development in this field will be essential to keep ahead of new threats and guarantee the security and integrity of web applications as the threat landscape continues to change.

REFERENCES:

[1] Aliga, A. P., John-Otumu, A. M., Imhanhahimi, R. E., & Akpe, A. C. (2018). Cross Site Scripting Attacks in Web-Based Applications. *Journal of Advances in Science and Engineering*, *1*(2), 25-35. https://doi.org/10.37121/jase.v1i2.19

[2] Soong, J. M. (2018). BreachAI: an artificial intelligence approach to enhance automated security testing of web applications.

[3] Wang, Y. H., Mao, C. H., & Lee, H. M. (2010). Structural learning of attack vectors for generating mutated xss attacks. *arXiv preprint arXiv:1009.3711*.

[4] F. Tang *et al*., "AIBench Training: Balanced Industry-Standard AI Training Benchmarking," *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Stony Brook, NY, USA, 2021, pp. 24-35, doi: 10.1109/ISPASS51385.2021.00014.

[5] Stency, V. S., & Mohanasundaram, N. (2021, February). A study on XSS attacks: intelligent detection methods. In Journal of Physics: Conference Series (Vol. 1767, No. 1, p. 012047). IOP Publishing.