Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.059696

ARTICLE

Utilizing Fine-Tuning of Large Language Models for Generating Synthetic Payloads: Enhancing Web Application Cybersecurity through Innovative Penetration Testing Techniques

Stefan Ćirković[1], Vladimir Mladenović[1], Siniša Tomić[2], Dalibor Drljača[2] and Olga Ristić[1,*]

[1]Faculty of Technical Sciences, University of Kragujevac, Čačak, 32000, Serbia

[2]Faculty of Information Technology, Pan-European University Apeiron, Banja Luka, 78101, Bosnia and Hercegovina

*Corresponding Author: Olga Ristić. Email: olga.ristic@ftn.kg.ac.rs

Received: 15 October 2024; Accepted: 03 January 2025; Published: 06 March 2025

ABSTRACT: With the increasing use of web applications, challenges in the field of cybersecurity are becoming more complex. This paper explores the application of fine-tuned large language models (LLMs) for the automatic generation of synthetic attacks, including XSS (Cross-Site Scripting), SQL Injections, and Command Injections. A web application has been developed that allows penetration testers to quickly generate high-quality payloads without the need for in-depth knowledge of artificial intelligence. The fine-tuned language model demonstrates the capability to produce synthetic payloads that closely resemble real-world attacks. This approach not only improves the model's precision and dependability but also serves as a practical resource for cybersecurity professionals to enhance the security of web applications. The methodology and structured implementation underscore the importance and potential of advanced language models in cybersecurity, illustrating their effectiveness in generating high-quality synthetic data for penetration testing purposes. The research results demonstrate that this approach enables the identification of vulnerabilities that traditional methods may not uncover, providing deeper insights into potential threats and enhancing overall security measures. The performance evaluation of the model indicated satisfactory results, while further hyperparameter optimization could improve accuracy and generalization capabilities. This research represents a significant step forward in improving web application security and opens new opportunities for the use of LLMs in security testing, thereby contributing to the development of more effective cybersecurity strategies.

KEYWORDS: LLM; GPT-2; XSS; SQL injection; command injection; evaluation loss perplexity

1 Introduction

Modern challenges in penetration testing of web applications arise from the limitations and inefficiencies of manually generating attack payloads [1]. While existing research has extensively explored various techniques for penetration testing, the lack of automation in payload generation and the absence of leveraging LLMs for this purpose highlight significant gaps in the literature. Current methods require penetration testers to design payloads manually, which is not only time-consuming but also prone to human error and limited by the tester's expertise. These inefficiencies often result in insufficient coverage of potential vulnerabilities and inconsistencies in test execution.

This research introduces an innovative approach by employing fine-tuned LLMs, specifically the GPT-2 model, to automate the generation of synthetic payloads. This methodology addresses the inefficiencies of manual payload creation and provides a systematic, repeatable, and scalable solution. By training the GPT-2 model on a specially tailored dataset, this study demonstrates how LLMs can generate advanced payloads for critical attack types, including XSS, SQL Injection, and Command Injection, effectively bridging the gap in current penetration testing practices. The unique contribution of this research lies in showcasing the potential of LLMs to not only automate but also enhance the precision and coverage of vulnerability detection, marking a significant advancement in cybersecurity methodologies [2].

1.1 Developed Web Application for Simple Payload Generation

As part of this research, a specialized web application was developed using the Flask framework to facilitate the automatic generation of synthetic payloads for various attack types, such as XSS, SQL Injection, and Command Injection. This application integrates a fine-tuned GPT-2 model trained on a dataset tailored for generating malicious payloads. Through a user-friendly interface, penetration testers and cybersecurity experts can specify the type of attack and instantly receive corresponding payloads [3]. This approach drastically streamlines the penetration testing process, reduces the time required for test preparation, and enhances the accuracy of vulnerability identification.

Unlike traditional tools, this web application leverages advanced language model capabilities, providing a level of automation and precision that has not been previously explored in penetration testing. It was designed exclusively for research purposes and is not publicly available, ensuring controlled use. The methodology section of this paper provides a detailed explanation of the functionality of the application, including an example of the function used to load the model and generate malicious payloads. The figure in this manuscript will illustrate the application's user interface structure, which enables the efficient definition and generation of payloads.

1.2 Contribution of the Research

This paper significantly contributes to the field of cybersecurity by pioneering the application of advanced language models in the context of web application penetration testing. It addresses a critical gap in current methodologies by introducing automation in payload generation, which is traditionally a manual and error-prone process [4]. The methodology presented here for training and fine-tuning LLMs, along with the results confirming their ability to generate synthetic payloads that identify and exploit vulnerabilities, showcases the practical potential of these models to enhance web application security.

Furthermore, this study explores the broader implications of using LLMs for penetration testing, emphasizing how the developed web application empowers penetration testers to conduct more efficient, scalable, and accurate assessments. By automating and optimizing payload generation, this approach not only enhances testing capabilities but also sets the stage for future advancements in cybersecurity tools and methodologies.

GPT-2

GPT-2, introduced by OpenAI in 2019, represents a significant advancement over previous models like BERT due to its unique autoregressive transformer architecture based solely on the decoder. This architecture is specifically designed for tasks such as translation, text generation, and question answering. Unlike BERT, which uses bidirectional context analysis within a sentence, GPT-2 is a unidirectional model, meaning it processes text in one direction—from the beginning to the end. During training, GPT-2 masks future tokens to prevent the model from using context that comes after the current token, ensuring that the prediction of the next word in a sentence is based solely on the preceding words. The GPT-2 architecture includes self-attention layers, attention between encoder and decoder (encoder-decoder attention), and feed-forward layers, enabling the model to efficiently process context and predict the next token in a sequence [5].

According to [6], GPT-2 excels in generating coherent and relevant textual responses, making it a significant tool for various applications in NLP.

To clearly illustrate the difference between the BERT and GPT-2 models, Fig. 1 shows a simplified view of their architectures. Fig. 1 displays BERT's encoder model with layers that enable bidirectional text processing, while the GPT-2 model is based solely on a decoder architecture, which processes text unidirectionally. This figure visually highlights the key differences in how the two models process text and understand context, thereby further explaining their specific applications in NLP tasks.

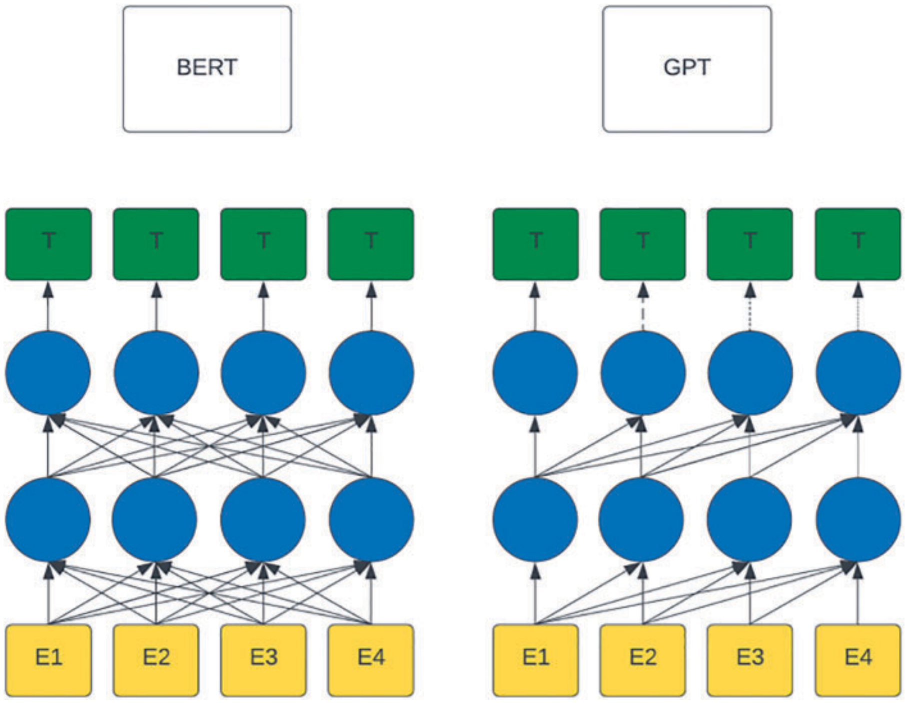


Figure 1: Difference between BERT and GPT-2 models [5]

In this research, the GPT-2 model was used instead of BERT due to differences in the architecture and primary goals of these models. While BERT is optimized for text understanding tasks, such as classification and entity extraction, its bidirectional architecture is not ideal for generating text sequences, which is crucial for the needs of this research.

The choice of GPT-2 over newer versions of the GPT model is based on the following reasons:

First, GPT-2 is an openly available model that does not require API access, which allows for complete control and privacy during use. This aspect is particularly important in research involving the generation of malicious payloads, where ensuring a high level of security and privacy is essential.

Second, although GPT-2 is not the latest version in the GPT series, its autoregressive decoder architecture allows for the generation of high-quality text sequences. These sequences are sufficiently complex and variable for identifying vulnerabilities in web applications, making it a suitable and practical choice for the purposes of this research. Due to these characteristics, GPT-2 was selected as the most appropriate option for penetration testing of web applications in this study.

2 Literature Review and Comparison of Related Research

The literature review plays a crucial role in laying the groundwork for any research, especially in the field of web application security, where the dynamics of threats and protective measures are constantly evolving. This paper focuses on analyzing security challenges in web applications, with particular emphasis on penetration testing, the security of LLMs, and the use of specific tools and techniques for preventing and detecting attacks. The goal is to identify existing methodologies and highlight the innovation of this approach, which differs from traditional methods of attack detection and classification presented in these studies.

The study of Happe et al. [7] explores the use of LLMs, such as GPT-3.5, in the context of penetration testing. The authors analyze how LLMs can be utilized for high-level task planning during security tests and for detecting lower-level vulnerabilities within a vulnerable virtual machine. Although this paper provides significant insight into the potential of LLMs, it is limited to applications in planning and detection, without exploring the possibility of generating new, sophisticated attacks.

On the other hand, authors in [8] examined the differences between automated and manual approaches to penetration testing, focusing on scenarios where one approach may be more effective than the other. The authors investigate various types of vulnerabilities, including those on the OWASP Top 10 list, to determine the advantages and disadvantages of both approaches. This paper offers a substantial analysis but deals exclusively with the detection of existing threats, whereas this research advances further by exploring the potential for generating new attacks using LLMs.

Paper [9] focuses on the security challenges associated with fine-tuning LLMs, such as Meta Llama and GPT-3.5 Turbo. The authors highlight how even minimal fine-tuning can seriously compromise the security of the model. This paper raises an important question about security during the customization of LLMs, but, like the previous papers, it addresses detection and protection, whereas this research investigates the generation of new attack scenarios.

Paper [10] analyzes the OWASP Security Shepherd technology as a tool for developing manual penetration testing, particularly in the context of protection against XSS attacks. While this research is important for education and prevention of XSS attacks, its focus is exclusively on existing threats, whereas this paper concentrates on creating new payloads using LLMs for attacking web applications.

Paper [11] provides a comprehensive review of methods for preventing and detecting SQL injection attacks, with an analysis of techniques such as input validation, parameterized queries, and intrusion detection systems. This study identifies the most effective combinations of methods for protecting web applications but, like the previous papers, mainly deals with detection, whereas this paper introduces innovation in generating attacks.

Paper [12] of OWASP dedicated to Command Injection attacks provides a detailed description of how attackers can exploit this vulnerability to execute unauthorized commands on a server. The goal is to educate about attack methods and preventive measures that can help protect web applications from such security threats. The page covers various attack techniques, examples, and recommended best practices for mitigation.

Penetration testing serves as an essential approach for uncovering vulnerabilities by mimicking real-life cyberattacks to evaluate the robustness of systems and networks. Although primarily a defensive strategy, some malicious actors exploit open-source tools to conduct pen tests for harmful purposes. The study of Lackhov et al. [13] outlines the stages and methodology of penetration testing, offering guidance and a demonstration of simulated attacks to assist IT administrators and cybersecurity experts in safeguarding their networks.

Karlsen et al. [5] investigate the effectiveness of different LLMs (BERT, RoBERTa, DistilRoBERTa, GPT-2, GPT-Neo) in analyzing log files to improve cybersecurity. The authors show that fine-tuned models, particularly DistilRoBERTa, achieve exceptional results in log analysis, with an average F1-score of 0.998, and introduce a new experimental framework, LLM4Sec, for optimizing and evaluating these models.

The subsequent paper from Yao et al. [14] explores the impact of LLMs such as ChatGPT and Bard on security and privacy. Through a comprehensive literature review, the paper analyzes the positive aspects of LLMs in the field of security, such as improved vulnerability detection in code and data privacy protection, but also potential risks and threats, including the use of LLMs for attacks. The authors identify key areas requiring further research, such as attacks on models and secure instruction tuning, and emphasize the need for continued study of these issues.

Authors Latif et al. [6] investigate the application of fine-tuned ChatGPT (GPT-3.5) for automatic grading of student-written responses in the field of science. Although GPT-3.5 has shown exceptional results in natural language generation, the standard model is not precise enough for grading, as students use specific language different from the material the model was trained on. This study fine-tuned GPT-3.5 using data from specific tasks and expert grading. The results show that fine-tuned GPT-3.5 achieves significantly higher accuracy in automatic grading compared to the BERT model, with an average accuracy increase of 9.1%. This study confirms the effectiveness of fine-tuned GPT-3.5 in education for automatic grading and provides public access to fine-tuned models.

Paper [15] explores the use of LLMs in automated penetration testing, demonstrating that LLMs can efficiently perform specific tasks but struggle with maintaining overall context. It introduces PENTESTGPT, a framework that uses LLMs to enhance the penetration testing process, and shows that PENTESTGPT significantly outperforms previous models with a task success increase of 228.6% and demonstrates effectiveness in real-world challenges.

In [16], the authors investigate the application of LLMs for code review automation using fine-tuning and prompt engineering methods. By studying variations of GPT-3.5 and Magicoder, they found that fine-tuning significantly improves accuracy compared to untrained approaches, while few-shot learning provides substantial improvements when training data is not available. The conclusion is that fine-tuning provides the best performance, while few-shot learning can be useful in situations with limited data.

The paper [17] examines the evolution of vulnerabilities in web applications through a detailed comparative analysis of the OWASP Top 10 lists from 2017 and 2021. It addresses changes in threat rankings, vulnerability descriptions, and identifies new trends, providing organizations, security experts, and developers with insights into contemporary challenges and shifts in web application security.

The study [18] explores advancements in artificial intelligence since the introduction of the Turing test in the 1950s, focusing on the development of language models from statistical to neural models, with particular emphasis on transformers and large language models like ChatGPT. The paper provides an overview of technological achievements and the contributions of LLMs in natural language processing, as well as the transition to large multimodal models that integrate various data types such as text, images, and sound, demonstrating advanced capabilities in understanding and generating content.

The paper [19] introduces an innovative approach to developing customized chatbots that focuses on efficiency and functionality. By combining three key technologies: LangChain, RAG, and LLMs, along with performance-enhancing techniques such as LoRA and QLoRA, the paper enables precise tailoring of chatbots to specific user needs. This approach improves accuracy, user experience, and access to information, allowing chatbots to efficiently process queries and provide useful responses.

In paper [20], authors present that LLMs have demonstrated significant proficiency in software testing. This paper conducts evaluation of LLMs applied to error tracing, test case generation, and bug localization across twelve open-source projects. The advantages and limitations, as well as recommendations associated with utilizing LLMs for these tasks, are delineated. Furthermore, we delve into the phenomenon of hallucination in LLMs, examining its impact on software testing processes and presenting solutions to mitigate its effects. The findings of this work contribute to a deeper understanding of integrating LLMs into software testing, providing insights that pave the way for enhanced effectiveness in the field.

Large language models (LLMs) have demonstrated versatility in various fields but require extensive data and computational resources. Fine-tuning these models, particularly using techniques like LoRA and its quantized version, QLoRA, enables adaptation to smaller datasets. In the paper [21], authors examine the repeatability of fine-tuning four LLMs with QLoRA over seven trials under consistent hardware and software settings, using two public datasets. Results reveal that fine-tuning with QLoRA is not stable, as performance varies significantly across trials on holdout test sets, even when conducted on a single GPU.

In [22], PT simulates hacker attacks to identify vulnerabilities but often relies heavily on human expertise, leading to high costs. AI-based approaches, such as reinforcement learning (RL) and deep reinforcement learning (DRL), offer efficient alternatives but face challenges like large action space dimensions affecting convergence. To address this, authors propose GAIL-PT, which leverages expert knowledge and a GAIL network to guide RL/DRL policy generation more effectively. GAIL-PT uses expert knowledge bases from pre-trained models, feeding them into its discriminator to optimize training and improve policy generation via integrated losses and discounted rewards. Experiments on real and simulated networks show GAIL-PT outperforms state-of-the-art methods like DeepExploit and Q-learning, proving its effectiveness and general applicability to RL/DRL-based PT.

Authors in [23] review over 100 Scopus-indexed studies on ChatGPT, launched in November 2022, which has shown promise across domains like healthcare, education, and creative writing, despite challenges with biases and trust. They classify ChatGPT research, analyze common approaches, and explore its applications in areas such as marketing, financial services, and NLP. Key issues, including biases and trustworthiness, are discussed, along with potential solutions and future research directions. Leveraging ChatGPT's capabilities could drive advancements in conversational AI and its societal impact.

In paper [24], authors give reviews over 100 Scopus-indexed publications, identifying common approaches and proposing a taxonomy of ChatGPT research. Since its launch in November 2022, ChatGPT has gained widespread attention for achievements like passing exams and writing poetry, despite challenges like biases and trust issues. Key applications and issues are explored, along with future research directions and potential solutions. As the first comprehensive review, this work highlights opportunities for advancing ChatGPT's capabilities across diverse fields.

The study [16] evaluates LLM-based code review automation using two approaches: fine-tuning with specific datasets and prompting with explicit instructions. The rapid development of LLMs has sparked interest in automating code reviews. However, resource-intensive approaches limit accessibility for budget-constrained organizations. Authors investigate 12 variations of two LLMs (GPT-3.5 and Magicoder) using fine-tuning and inference techniques (zero-shot, few-shot, and persona learning) and compare them with Guo et al.'s [25] approach and three existing tools (CodeReviewer, TufanoT5, D-ACT). Fine-tuned GPT-3.5 achieves up to 74.23% higher exact match (EM) than Guo et al., while few-shot learning without fine-tuning outperforms zero-shot by up to 659.09%. Key recommendations include fine-tuning for optimal performance and using few-shot learning without a persona for scenarios with limited data. These findings provide practical guidance for deploying LLMs in code review automation.

None of the presented studies directly address the generation of new payloads for testing web application vulnerabilities using LLMs, making this approach innovative and significant for the further development of methodologies in this field. This research focuses on generating specific attacks using LLMs, representing an advancement over studies that deal with the detection or classification of existing threats. This innovation contributes to a better understanding of the potential of LLMs in security research and the enhancement of web application penetration testing.

3 Theoretical Foundations

3.1 Large Language Model (LLM)

LLMs are advanced neural networks trained using vast datasets. The performance of these models largely depends on their complexity, which is typically measured by the number of parameters. Modern LLMs vary in parameter size from several billion, such as LLaMA with 7 billion parameters, to several trillion, such as Wu Dao or GPT-4. Although larger models can exhibit unpredictable behaviors, there is ongoing debate about whether further expansion of models is sustainable due to diminishing returns in scaling efficiency [7].

Training an entirely new LLM is often financially prohibitive for most researchers, but existing models can be fine-tuned or further trained for specific purposes at a more manageable cost. This practice has led to the concept of "base models" [7].

According to recent research, LLMs have undergone significant evolution compared to earlier language models. Earlier models were statistical in nature and laid the groundwork for computational linguistics, while transformers enabled substantial advancements in scalability [14]. Modern LLMs undergo extensive training on massive datasets to understand and generate text that closely mimics human language. These models, with hundreds of billions or even trillions of parameters, have made significant strides in NLP and find applications in various fields, including risk assessment, programming, vulnerability detection, medical text analysis, and search engine optimization [14].

Using LLMs for generating and optimizing prompts can significantly enhance the efficiency of penetration testing by providing opportunities to develop better questions that penetration testers can ask. These models represent a powerful tool with vast potential for improving the security of information systems, but ethical use of LLMs is of fundamental importance [14].

3.1.1 BERT

The BERT model, introduced by Google in 2018, significantly advanced NLP using a bidirectional transformer architecture. The BERT model processes text in both directions—left to right and right to left—enabling it to detect and understand contextual relationships from both sides of a word in a sentence [14]. The model was trained on a large corpus containing over 3.3 billion words, applying techniques such as masked language modeling and next sentence prediction to further enhance text comprehension. BERT's architecture includes 12 layers of transformer encoders, each with 768 hidden states, and utilizes the WordPiece algorithm for tokenization. This powerful framework allows BERT to be fine-tuned for specific domains, providing significant embedding representations for various NLP tasks [5].

3.2 Penetration Testing with LLMs: Generating Payloads

Penetration testing, or "pen-testing," is a critical process in assessing the security of information systems [8]. In the traditional penetration testing process, security professionals analyze the target system using automated tools through five key phases: reconnaissance, scanning, vulnerability assessment, exploitation, and post-exploitation. These phases allow for a comprehensive understanding of the system, identification of vulnerabilities, and their exploitation to gain access to the system [15].

Using LLM for this purpose represents an innovative approach that enables the automation and enhancement of this process. One of the most critical aspects of penetration testing is generating payloads, i.e., malicious code or data used to exploit vulnerabilities in the target system.

LLM models, such as GPT-2 and BERT, provide the capability to generate sophisticated and customized payloads through learning from large amounts of data. These models can be fine-tuned for specific domains, thereby increasing their effectiveness in various security scenarios. This study focuses on testing web applications, which allows for the use of LLM models to automate payload generation and enhance the efficiency of penetration testing.

3.3 Tokenization

Tokenization is a crucial step in preparing textual data for working with LLMs. During the tokenization process, text is broken down into smaller units known as tokens. These tokens can be whole words or parts of words, depending on the specific architecture of the model. This process allows models to accurately analyze and interpret textual information, enhancing learning and language generation efficiency. Tokenization thus plays a key role in enabling models to handle complex linguistic structures and produce relevant results [19].

By correctly applying tokenization, textual data can be transformed into a format that allows LLMs to learn more efficiently. This process contributes to the development of robust and efficient solutions for generating advanced synthetic payloads. As researchers in this field, it is essential to continuously evaluate the effectiveness of these methods and explore innovative approaches to improve data preparation, with the goal of building advanced systems for application security testing.

3.4 Fine-Tuning

Fine-tuning is a method used to adapt pre-trained LLMs to specific tasks and applications. This technique involves directly updating the model's parameters based on smaller datasets, allowing for improved performance on particular tasks. There are many approaches that focus on optimizing the efficiency of this process, aiming to achieve a better balance between quality and speed [16]. This process enhances the model's ability to understand and execute complex textual instructions, significantly increasing the model's flexibility and enabling it to efficiently perform various tasks according to the specific requirements of users [18]. Although methods like context learning and prompt queries do not require parameter changes, fine-tuning is often still preferred as it reduces additional burdens during the inference process and generally provides better and more stable results [9].

In this study, fine-tuning was applied to the GPT-2 model due to the specific advantages it offers. Once the GPT-2 model is pre-trained, it becomes a completely private resource, eliminating the need for API access. This characteristic makes the GPT-2 model particularly suitable for applications that require greater privacy and control over data, as it allows operation without relying on external services.

3.5 OWASP

The OWASP is a global non-profit organization focused on improving software application security. OWASP provides organizations with the resources and tools needed to develop, purchase, and manage secure software. Through a wide range of free and open resources, OWASP grants access to security standards, testing and secure coding books, as well as code analysis [17]. The organization offers reports, video materials, optional security tests, repositories, and support through local chapters around the world.

OWASP is also known for organizing major international conferences and advancing scientific research related to cybersecurity. The goal of OWASP is to educate professionals, including developers, designers, architects, and business leaders, about the risks arising from increasingly common security vulnerabilities in web applications. It is particularly known for its "Top 10" list, which identifies the most dangerous security threats in web applications and provides guidelines for mitigating them [10].

3.5.1 XSS

XSS is a well-known attack on web applications that occurs when malicious web code, usually in the form of a script, is sent and executed through the victim's browser using their web applications. This attack allows attackers to filter personal information or steal cookies from users to hijack session identities. XSS attacks can lead to the theft of confidential data or even take control of other computers. According to data, XSS accounts for 40% of attack attempts on web applications, making it the most common form of attack in this field [10].

3.5.2 SQL Injection

SQL injection is a serious security threat in web applications that use databases. This attack allows attackers to insert malicious SQL code into an application, gaining unauthorized access to confidential data. SQL injection attacks often occur when applications inadequately validate user inputs, allowing manipulation of database queries. The most effective defense includes a combination of prevention methods, such as input validation and parameterized queries, with detection techniques like intrusion detection systems. Despite advanced protective methods, SQL injections remain one of the most common and dangerous forms of cyber-attacks on web applications [11].

3.5.3 Command Injection

Command Injection is a type of attack where an attacker exploits a vulnerable web application to execute arbitrary commands on the operating system. These attacks occur when the application passes unsafe data provided by users, such as form inputs, cookies, or HTTP headers, to the system shell. The attacker sends commands that are then executed with the privileges of the vulnerable application. The main cause of these attacks is the lack of adequate input validation [12].

3.6 Performance Metrics

When fine-tuning a LLM for text generation, such as for creating new payloads for cybersecurity, it's important to use appropriate performance metrics to assess the model's effectiveness. Two key metrics for evaluating the model in this context are evaluation loss and evaluation perplexity. These metrics help assess how well the model generates coherent and relevant text. Evaluation loss measures the model's accuracy in predicting the correct words, while perplexity provides insight into how "natural" the model is at generating text, with lower values suggesting a better ability to produce coherent content.

3.6.1 Evaluation Loss

Evaluation loss measures how much the model's predictions differ from the actual results in the evaluation dataset. In the context of text generation models, this loss indicates how well the model predicts the next word in a sequence based on the previous words. Mathematically, evaluation loss is measured using the cross-entropy loss function, which is defined by the following Eq. (1):

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}\log p\left(y_i|x_i\right), \tag{1}$$

where

N—the number of words in the evaluation dataset.

$p\left(x_i|y_i\right)$—the probability assigned by the model to the correct word y_i given input x_i.

3.6.2 Evaluation Perplexity

Evaluation perplexity measures how "confused" the model is by the evaluation data; lower perplexity indicates that the model is better at predicting words. Perplexity is an exponential function of evaluation loss and provides a better understanding of the quality of the generated text. Mathematically, perplexity is defined as (2):

$$Perplexity = e^{Loss}, \tag{2}$$

where *Loss*—evaluation loss from the previous formula.

Perplexity is often used to compare performance between different models and configurations. Lower perplexity indicates that the model better predicts the next words in a sequence and generally produces more coherent texts.

4 Methodology

In this research, an LLM was used to generate synthetic payloads for attacks such as XSS, SQL Injection, and Command Injection.

4.1 Dataset

The dataset used in this research was obtained from Kaggle [26] and contains a total of 199,797 records in CSV format. This dataset covers three main types of attacks: SQL Injection, XSS, and Command Injection (see Table 1). Each record includes a unique identifier, the text content of the payload, the type of attack, and a binary label indicating whether the payload is an attack or normal. This dataset is carefully designed to enable the analysis and recognition of different attacks, providing support for training machine models and exploring patterns in payloads. By using this dataset, the research focuses on developing methods for automatic detection and prevention of vulnerabilities in web applications, thereby enhancing the security and efficiency of protective systems. In this study, the dataset was used for fine-tuning LLMs with the goal of generating new payloads to enrich the penetration testing process.

Table 1: The dataset on attack types

Sentence	SQL Injection	XSS	Command Injection	Normal
find trunk -type f -exec curl –user user:pass...	0.0	0.0	1.0	0.0
z^n^$h^r3]21%g96$bu{)d1{y+o-x9x`()q),9u7v7l~a...	1.0	0.0	0.0	0.0
It is a story as old as man. The jealousy for...	0.0	0.0	0.0	1.0
I'm racking my brain, but I can't seem to thin...	0.0	0.0	0.0	1.0
MJ7whfind -type f -printf "%s %p\n" | sort -nr...	0.0	0.0	1.0	0.0

(Continued)

Table 1 (continued)

Sentence	SQL Injection	XSS	Command Injection	Normal
<multicol id=x tabindex=1 onbeforedeactivate=a...	0.0	1.0	0.0	0.0
-2992%'))) or 4493 = utl_inaddr.get_ho...	1.0	0.0	0.0	0.0

4.2 Fine-Tuning Process

Fig. 2 shows the comprehensive workflow for fine-tuning a pre-trained LLM to generate synthetic payloads that can be used for penetration testing of web applications. The process involves several key stages, each contributing to the development and implementation of a robust model capable of generating realistic and effective attack vectors [27].

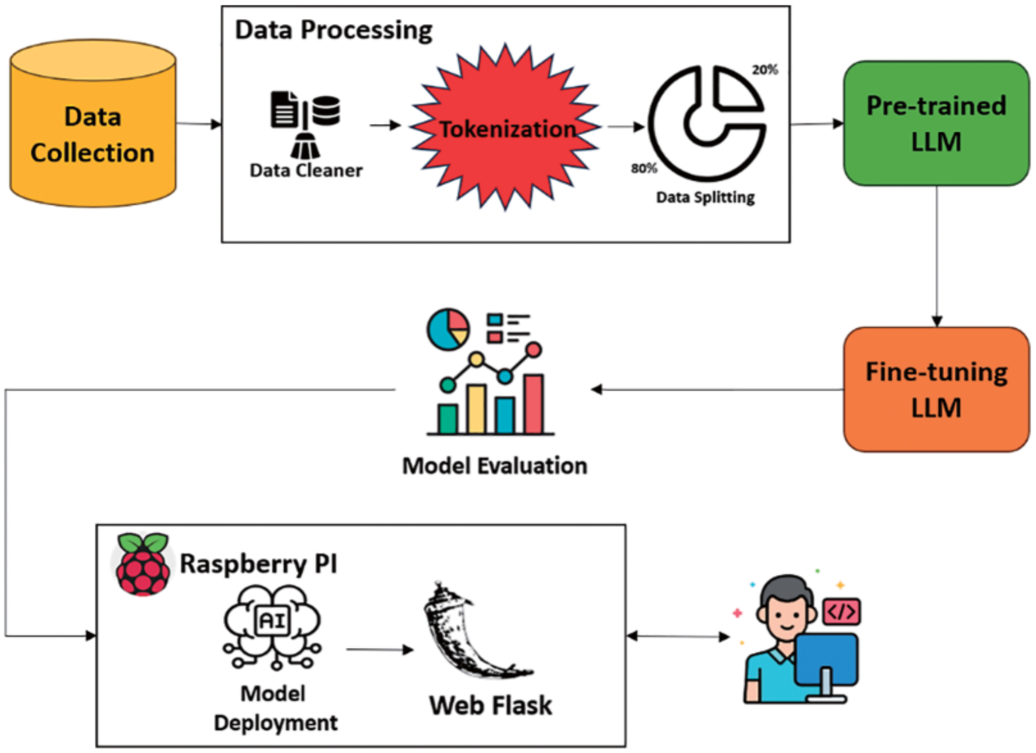


Figure 2: Workflow for fine-tuning a pre-trained language model

- **Data Collection**

This initial phase involves gathering relevant datasets containing examples of payloads for web applications, such as XSS, SQL Injection, and Command Injection. The dataset used in this research is detailed in the previous section.

- **Data Processing**

After collection, the raw data is carefully processed to prepare it for model training. This step involves several key processes: cleaning the data to remove faulty or unnecessary information, tokenizing the text to convert it into sequences of tokens, and formatting the data into formats that the model can efficiently use. Data processing ensures high-quality input, enabling precise model training. All normal payloads were removed from the dataset as this work focuses on generating attacks rather than detecting them. The data was split into 80% for training the model and 20% for testing, and placed into two .txt files—one for training and one for testing. Data formatting includes examples such as: SQLInjection -> ‘ OR ‘1’=‘1’;

- **Pre-trained LLM**

In this process, a pre-trained language model, such as GPT-2 in this case, is used. This model was initially trained on a large corpus of general text. This pre-trained model provides a solid foundation with a broad understanding of linguistic patterns and structures.

- **Fine-Tuning**

In this critical phase of the research, the pre-trained GPT-2 model was further fine-tuned on a specific dataset related to web application payloads. This fine-tuning process allows the model to adjust its parameters to more accurately understand and generate text that mimics the structure and content of the given payload examples. The training was conducted on a Huawei Atlas 3010 server, utilizing 36 Intel(R) Xeon(R) Gold 6240 CPU cores at 2.60 GHz, with a total of 125.31 GiB of memory. During the training, 15% of the RAM and 50% of the CPU resources were used. Efficient management of computational resources enabled optimized training with a batch size of 2 and a total of 1 epoch, ensuring an optimal balance between learning depth and model generalization. The training lasted 43 h, with logs recorded every 200 steps, while models were saved every 10,000 steps, with a maximum of two models saved to manage storage efficiently. This systematic approach to fine-tuning allowed the GPT-2 model to generate realistic and precise payloads, leading to improved accuracy and reliability in penetration testing.

- **Model Evaluation**

After fine-tuning, the model undergoes evaluation to assess its performance. Metrics such as Evaluation Loss and Evaluation Perplexity are crucial for assessing the LLM’s effectiveness in generating realistic and diverse payloads. The evaluation ensures that the model meets the desired standards before implementation. The obtained evaluation results are presented in the chapter on results and interpretations.

- **Model Deployment**

The fine-tuned model is implemented on a Raspberry Pi 5 platform, which serves as the server, making it available for generating synthetic payloads in real time. This phase involves integrating the model into a web application framework, such as Flask, which enables interaction with the model through a user interface.

- **Flask-Based Web Application**

The application was developed using the Flask framework, which provides a flexible and simple way to implement web applications. Flask serves as the backend for integrating with the fine-tuned GPT-2 model, allowing users to generate synthetic payloads through an intuitive interface. The payload generation function utilizes the GPT-2 model by first loading the model and the corresponding tokenizer, which converts the input prompt into a format the model can process. The function then generates multiple text sequences representing potential payloads, using sampling methods with top-k and top-p parameters to ensure diversity and coherence in the results. The generated sequences are decoded and returned as a list of possible malicious texts, enabling users to obtain appropriate payloads in real time.

Python code

```
# Loading the model and tokenizer
model_path = "./fine_tuned_gpt2"
model = GPT2LMHeadModel.from_pretrained(model_path)
tokenizer = GPT2Tokenizer.from_pretrained(model_path)
# Payload generation function
def generate_payload(prompt, model, tokenizer, max_length=60,
num_return_sequences=5):
    inputs = tokenizer(prompt, return_tensors="pt")
    outputs = model.generate(
        inputs["input_ids"],
        max_length=max_length,
        num_return_sequences=num_return_sequences,
        do_sample=True,
        top_k=50,
        top_p=0.95,
        temperature=0.7,
    )
    return [tokenizer.decode(output, skip_special_tokens=True) for
output in outputs]
```

The frontend shown in Fig. 3 is designed to be user-friendly, making the process of generating malicious payloads simple and efficient. The user interface is directly connected to the model, allowing users to input the desired attack type (XSS, SQL Injection, Command Injection) and receive the corresponding payloads in real time. This functionality enables penetration testers and cybersecurity experts to effectively use the tool in their work without needing deep technical knowledge about the model or its training.

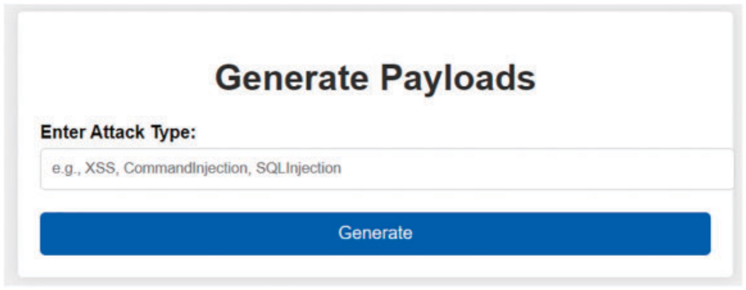


Figure 3: Display of the malicious payload generation application

- **User Access Web Application**

End users, including penetration testers and cybersecurity experts, access this web application. Fig. 3 shows a simple and intuitive application that allows the generation of malicious payloads for three specific types of attacks: XSS, SQL Injection, and Command Injection. The application uses the GPT-2 model, which has been fine-tuned on a dataset specific to these attacks. This dataset includes various examples of malicious payloads, enabling the model to create new, potentially dangerous payloads based on user-input parameters.

Following the steps in the methodology shown in Fig. 2, the fine-tuned language model can effectively generate synthetic payloads that mimic real attacks. This process not only enhances the accuracy and reliability of the model but also provides a practical tool for cybersecurity experts to improve web application security. This methodology and its systematic implementation highlight the significance and potential of advanced language models in the field of cybersecurity, demonstrating their usefulness in generating high-quality synthetic data for penetration testing.

5 Results and Interpretation

5.1 Training Results

During the model fine-tuning process, performance was evaluated using two key metrics: Evaluation Loss and Evaluation Perplexity, calculated based on the mathematical expressions shown in Eqs. (1) and (2).

Evaluation Loss, shown in Fig. 4, is calculated using the formula in Eq. (1), which defines the loss function as the negative logarithmic probability of the model's output relative to the actual data. This metric measures the accuracy of the model's predictions, with lower values indicating better performance. In this experiment, Evaluation Loss ranges from 1.42 to 1.58, suggesting that the model achieved a stable and reliable level of accuracy. The lack of significant variations in loss indicates stable model performance during training, which is crucial for avoiding overfitting and ensuring good generalization to new data. Although current results are satisfactory, further hyperparameter optimization could potentially reduce the loss value, thereby improving the model's performance.

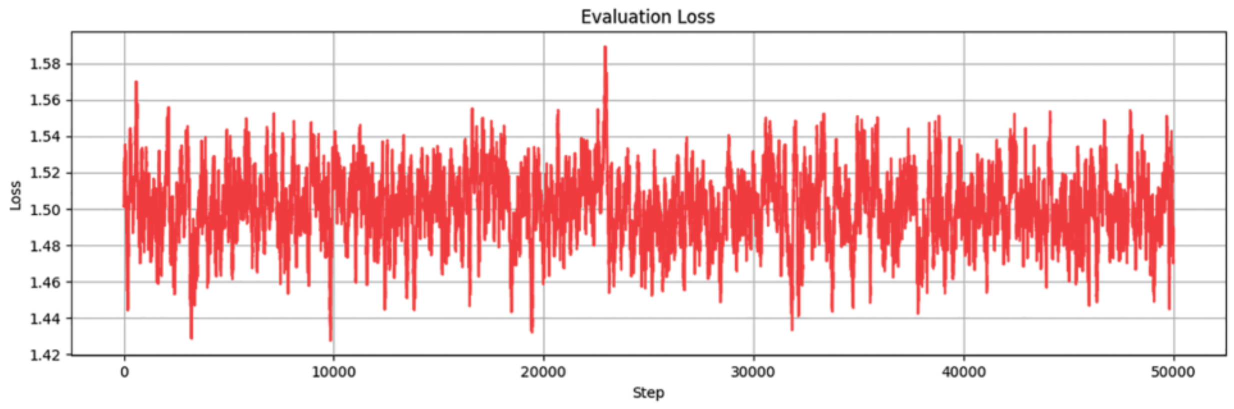


Figure 4: Evaluation loss

Perplexity, calculated according to Eq. (2), represents the logarithmic measure of the average number of possible outputs the model considers when generating sequential data. As shown in Fig. 5, Evaluation Perplexity ranges from 47 to 53, indicating stable performance with room for improvement. Lower perplexity values, closer to the lower end of this range, suggest that the model has a better ability to predict sequences, which is crucial for tasks such as language modeling and text generation. The stability of perplexity within these bounds can be considered acceptable for many applications, but further model optimization could potentially reduce these values, thereby enhancing the accuracy and reliability of the model's predictions.

The evaluation results provide a basis for further analysis and optimization to achieve even better model performance, including considering different training strategies, modifications to the model architecture, or hyperparameter optimization to achieve significant improvements in Evaluation Loss and Perplexity metrics.

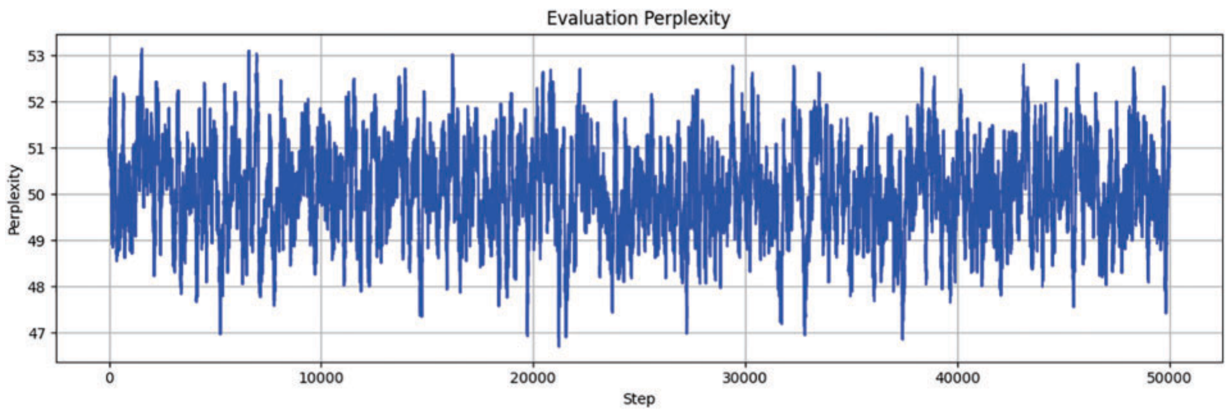
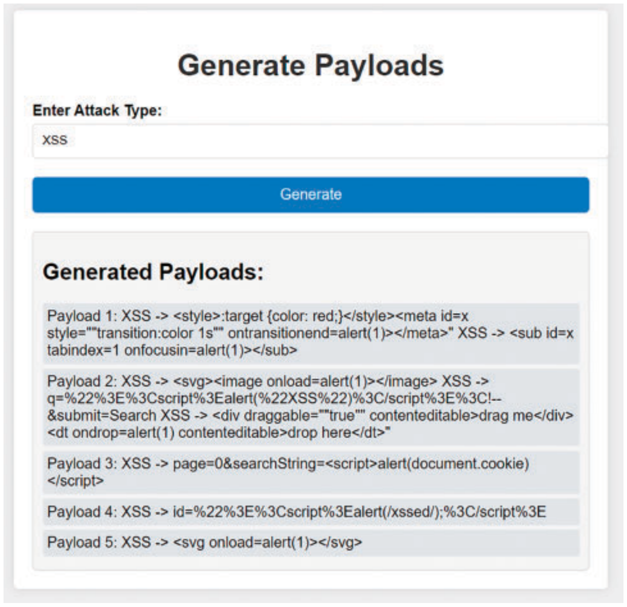


Figure 5: Perplexity

5.2 Generating Payloads Using the Web Application

In the current iteration of the application shown in Fig. 6, the model generated five XSS payloads. Each of these payloads contains characteristic XSS elements such as <script>, <svg>, and onload events, which are known for their ability to trigger the execution of malicious code on the client side (in the web browser).



```
Payload 1: XSS -> <style>:target {color: red;}</style><meta id=x style=""transition:color 1s"" ontransitionend=alert(1)></meta>" XSS -> <sub id=x tabindex=1 onfocusin=alert(1)></sub>

Payload 2: XSS -> <svg><image onload=alert(1)></image> XSS -> q=%22%3E%3Cscript%3Ealert(%22XSS%22)%3C/script%3E%3C!--&submit=Search XSS -> <div draggable=""true"" contenteditable>drag me</div><dt ondrop=alert(1) contenteditable>drop here</dt>"

Payload 3: XSS -> page=0&searchString=<script>alert(document.cookie)</script>

Payload 4: XSS -> id=%22%3E%3Cscript%3Ealert(/xssed/);%3C/script%3E

Payload 5: XSS -> <svg onload=alert(1)></svg>
```

Figure 6: Generated payloads

Analysis of Generated Payloads

- **Payload 1:** This payload combines CSS and HTML elements with JavaScript code executed via the *onfocusin* event. The *onfocusin* event is a JavaScript event that triggers when an element, such as a text input field, receives focus, meaning when a user clicks on it or selects it in some other way. The payload structure demonstrates how attacks can be sophisticated by using diverse technologies like CSS to deceive systems. The use of the *alert(1)* function here is symbolic and used solely for educational and ethical purposes. In a real attack scenario, this payload could be expanded to send stolen data, such as cookies, to a malicious server.

- **Payload 2:** The payload uses the onload event within an SVG tag to execute JavaScript code. The use of *alert(1)* serves as a harmless proof of concept, while a real attack could use this vector to perform much more harmful actions, such as redirecting user data to a hacker's server.
- **Payload 3:** This payload targets *document.cookie* through a script within *the searchString* parameter, demonstrating the potential for cookie theft. However, in this instance, *alert(document.cookie)* is used instead of potentially more dangerous code that could send these cookies to a malicious server. This approach is applied to maintain ethics and safety during testing.
- **Payload 4:** The payload is minimalist, focusing on the direct execution of a script. The use of the alert function here is deliberately chosen to avoid causing actual harm, although in an attack scenario, such a payload could be used for much more destructive actions.
- **Payload 5:** A simple payload that uses SVG to execute the alert function. This payload demonstrates how code execution can be achieved with straightforward techniques, while in real conditions, this payload could be exploited for unethical purposes.

The use of the alert function in the generated payloads is designed to demonstrate the concept of XSS attacks in an ethical manner, without actually compromising security. In practice, these payloads could be modified to cause significant harm, such as stealing cookies or redirecting sensitive information to malicious servers.

5.3 Testing of Payload Vectors on the Vulnerable DVWA Application

DVWA is a web application specifically designed for educational purposes and testing vulnerabilities in a controlled environment. One of its key advantages is the ability to set different security levels (Low, Medium, High), allowing users to test various payloads in simulated scenarios that mimic real-world security challenges [28].

In this study, DVWA was chosen over OWASP Juice Shop as the research platform because it enables dynamic testing of generic payloads without the limitations imposed by predefined challenges. While OWASP Juice Shop is an extremely useful educational tool, it relies on specific CTF scenarios, which may complicate the validation of universal payloads. This tool expects precisely defined payloads, which are intentionally set in line with particular challenges, whereas DVWA offers flexibility and freedom in testing a wide range of vulnerabilities, making it ideal for simulating real-world attacks and assessing the effectiveness of various security mechanisms.

Table 2 presents the payloads that were tested on the DVWA application. Each payload is marked with the corresponding security levels (Low, Medium, High) and indicators showing whether the payload passes through the application at each of these levels. This table provides an overview of the effectiveness of various payloads under simulated conditions of different security configurations.

Table 2: Results of payload testing on DVWA application

Payload ID	Payload	Low	Medium	High	Description
XSS 1	<style>:target {color: red;} </style><meta id=x style="transition:color 1s" ontransitionend=alert(1)></meta><x><sub id=x tabindex=1 onfocusin=alert(1)></sub>	✓	×	×	CSS transitions and events like onfocusin for an XSS attack.
XSS 2	<img onload=alert(1)> </image> XSS -> g="%22%3E%3C/script%3Ealert('XSS')%3C/script%3E%3C!–&submit=Search XSS –> <div draggable=true contenteditable>drag me</div> <dt ondrop=alert(1)>)	✓	×	×	XSS using the onload and ondrop events.
XSS 3	page=0&searchString=<script>alert (document.cookie)</script>	✓	×	×	Cookie theft using the document.cookie.
XSS 4	id="%22%3E%3C/script%3Ealert('XSS')%3C/script%3E	✓	✓	×	Encoded XSS payload to bypass basic security mechanisms.
XSS 5	<svg onload=alert(1)> </svg>	✓	✓	×	SVG payload with onload for XSS.
SQL Injection 1	'UNION SELECT null,username, password FROM users –	✓	✓	×	Authentication bypass and displaying user data from the database.
SQL Injection 2	' OR '1'='1	✓	×	×	Classic SQL Injection payload for authentication bypass.
SQL Injection 3	'; DROP TABLE users; –	✓	✓	×	SQL injection that attempts to delete the users table from the database.
SQL Injection 4	' AND (SELECT COUNT(*) FROM information_schema.tables) > 0 –	✓	✓	×	Checking the number of tables in the database using SQL injection.
Command Injection 1	; ls -la	✓	✓	×	Command injection attempting to display files in the current directory.
Command Injection 2	; cat /etc/passwd	✓	✓	×	Attempts to execute a command to display user data.
Command Injection 3	; touch /tmp/exploit; chmod 777 /tmp/exploit	✓	✓	✓	Creates a file with all permissions and allows the attacker to manipulate it.

In this chapter, various payload vectors were tested on the DVWA application to assess their effectiveness under different security levels (Low, Medium, High). The results showed varying success rates of payloads across different security levels, further emphasizing the importance of implementing appropriate security measures to protect against common attacks such as XSS, SQL injections, and Command Injection attacks.

By analyzing the results of the testing within the context of the DVWA application, we gained insights into the performance and vulnerabilities of the application under simulated conditions. This testing provided crucial information on how different security settings affect the ability of payloads to bypass the application's defenses. The success of several payloads at lower security levels confirms the need for more robust security measures to ensure adequate protection against the most common and dangerous types of attacks.

5.4 Inference Time and System Resource Usage for Web Application Payload Generation

This section presents the results of measuring the time required to generate synthetic attacks on web applications using a fine-tuned model on a Raspberry Pi 5 device. Measuring inference time, or the time it takes for the model to generate a payload, as well as analyzing system resource usage (CPU and RAM), is crucial for assessing the model's performance, especially in resource-constrained environments.

Testing was conducted on a Raspberry Pi 5 device, a hardware-constrained platform, to evaluate the speed of attack generation after completing the fine-tuning process. The total time required to generate a single synthetic attack was 10.5231 s. These results confirm that the model is optimized for rapid payload generation, even in environments with modest hardware capacities, making its practical application feasible.

The Fig. 7 illustrates system usage during the payload generation process. The top chart shows CPU usage, where all four cores of the Raspberry Pi 5 processor were engaged at different stages of the process. A significant increase in CPU load is observed, with occasional spikes to nearly maximum capacity, indicating intensive processing during attack generation.

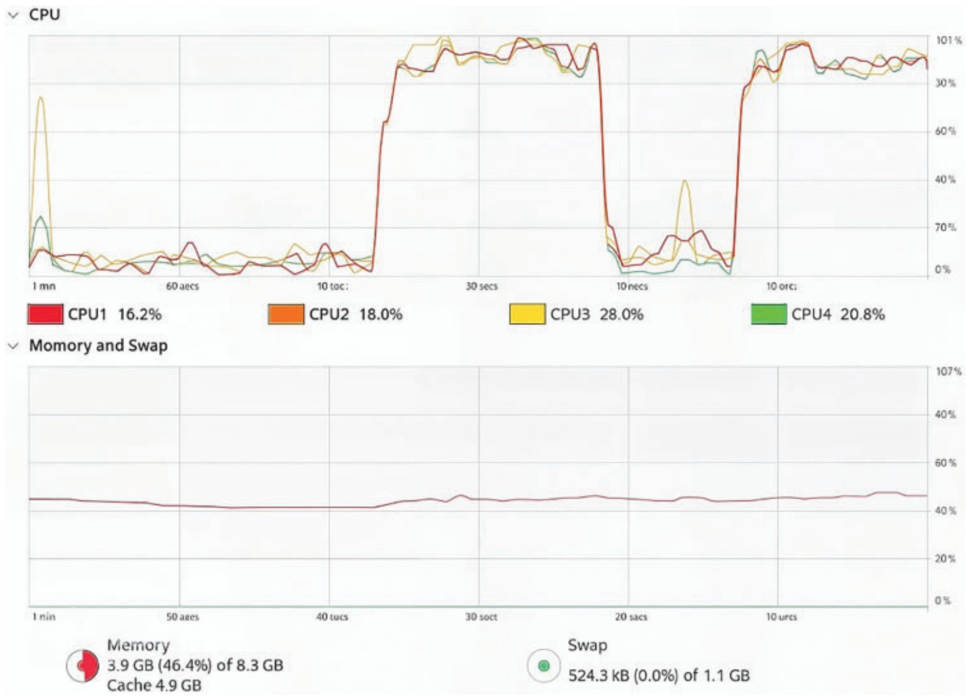


Figure 7: System resource usage during payload generation on Raspberry Pi 5

The bottom chart in the Fig. 7 displays RAM usage, which remained stable throughout the process, with an occupancy of 46.4% of the total 8.3 GB capacity. This data indicates that while the model is CPU-intensive, it does not require a large amount of memory for its operations, further emphasizing its efficiency in environments with limited available memory.

These results provide valuable insights into how the fine-tuned model performs in real-time on a resource-limited platform such as the Raspberry Pi 5, further confirming the practicality and efficiency of this approach in the context of advanced cybersecurity for web applications.

6 Ethical Aspects of Fine-Tuning LLMs

The use of LLMs to automate payload generation in penetration testing offers significant benefits for advancing cybersecurity but also carries potential ethical risks, particularly the misuse of such technologies for malicious purposes. To mitigate these risks, this study strictly complies with the General Data Protection Regulation (GDPR) of the European Union, which mandates data privacy and minimization [29]. Furthermore, the research adheres to the OECD Principles on Artificial Intelligence, emphasizing human autonomy, transparency, and accountability [30], as well as the IEEE Ethically Aligned Design standard, which prioritizes human well-being and responsible use of technology [31]. The model was fine-tuned on a specific dataset with strict limitations on its universal applicability, further reducing the risk of misuse.

Technical measures include authentication and authorization to restrict access to authorized users, activity monitoring through logging systems to ensure accountability, and encryption protocols to protect data during payload generation. Transparency in how payloads are generated is embedded into the application, enabling better user understanding and fostering trust. All tests were conducted under strictly controlled laboratory.

In addition to technical safeguards, this study proposes an ethical framework encompassing legal compliance with global standards such as GDPR and NIST, transparent application of the model for educational and research purposes, and the development of advanced security protocols for automatic misuse detection [32]. This approach addresses challenges identified in the literature, such as the risk of security compromise during model fine-tuning [9], and contributes to the development of practical solutions to enhance trust in AI technologies [14].

In conclusion, this ethical framework not only addresses potential misuse but also lays the groundwork for the responsible application of AI technologies in cybersecurity, contributing to global efforts to improve digital security.

7 Conclusion

This research represents a pioneering effort in the field of cybersecurity, with a specific focus on the application of LLMs for the automated generation of malicious payloads in penetration testing processes. The developed methodology and tool enable more efficient and precise detection of vulnerabilities in web applications, significantly advancing existing industry practices. Using fine-tuning techniques on a specific dataset, the GPT-2 model was successfully trained to generate advanced payloads for three key types of attacks: XSS, SQL Injection and Command Injection.

Expanding the dataset to include a broader range of attacks could enhance the practical application of this approach and allow the model to generate new attack vectors, such as those listed by OWASP. Through fine-tuning the model on extended datasets, future research could further improve the versatility and efficiency of this model.

Experimental results confirm that this approach not only reduces the time required for test preparation but also significantly improves the coverage and accuracy of vulnerability detection compared to traditional methods. A web application, implemented on the Raspberry Pi platform and integrating this model, provides penetration testers with a powerful tool for quickly and effectively creating attack scenarios. This tool enables testers to use a private model without relying on external APIs, which is crucial in terms of data privacy and security.

Data privacy and security remain key priorities in this research. The use of the GPT-2 model, which allows local implementation without reliance on external APIs, demonstrates how the study successfully addresses data protection challenges, contributing to greater reliability and control during payload generation.

Although more advanced models, such as GPT-3.5 and GPT-4, are available, their exclusive API-based access and limited privacy did not align with the objectives of this research. Successfully implementing GPT-2 on Raspberry Pi 5, a computationally constrained platform, is a significant achievement, as running such a large model with billions of parameters on a small device represents a challenge. The ability to generate payloads in just a few seconds highlights the model's potential for real-time applications, while on more powerful servers, generation could be nearly instantaneous, enabling rapid and efficient security assessments of web applications.

The generated payloads were further tested on the DVWA to validate their effectiveness using metrics such as Evaluation Loss and Perplexity. This testing framework was chosen over OWASP Juice Shop due to the latter's design for CTF competitions, where some valid payloads might not execute in that specific context.

While GPT-2 demonstrates excellent results in generating synthetic attacks, relying on a single model introduces certain limitations, particularly in terms of the diversity and robustness of the generated payloads. This approach may face challenges in recognizing new and more sophisticated attacks, as it lacks the breadth of models with varying architectures. Limited diversity in the model could lead to scenarios where the generated payloads fail to address all types of attacks that may occur in real-world conditions.

Future research should address these limitations by expanding the ensemble with multiple models and diverse architectures. Including models such as T5 and RoBERTa, as well as generative models with different techniques, could significantly improve the robustness and precision of the generated attacks and expand the range of attacks that can be simulated.

This study contributes to the academic community by providing a new methodology that can serve as a foundation for future research in the fields of cybersecurity and artificial intelligence. Plans for the future include further improving the model through the application of the latest NLP techniques and expanding experimental scenarios to various types of applications and systems. Additionally, opportunities to integrate this approach into existing automated testing tools will be explored, further increasing its practical value and industrial applications.

Acknowledgement: The authors would like to thank the anonymous reviewers for their helpful comments to improve the technical quality of the paper.

Funding Statement: This study was supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia, and these results are parts of Grant No. 451-03-66/2024-03/200132 with the University of Kragujevac–Faculty of Technical Sciences Čačak.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Stefan Ćirković, Vladimir Mladenović, Olga Ristić; methodology: Siniša Tomić, Dalibor Drljača; data collection: Stefan Ćirković, Siniša Tomić, Dalibor Drljača; analysis and interpretation of results: Stefan Ćirković, Vladimir Mladenović, Olga Ristić; draft manuscript preparation: Stefan Ćirković, Vladimir Mladenović, Olga Ristić. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The corresponding author can provide data supporting the study's conclusions upon an adequate request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

Abbreviations

LLM	Large Language Model
XSS	Cross-Site Scripting
GPT	Generative Pre-trained Transformers
NLP	Natural Language Processing
OWASP	Open Web Application Security Project
BERT	Bidirectional Encoder Representations from Transformers
OECD	Organization for Economic Co-operation and Development
PT	Penetration Testing
GDPR	General Data Protection Regulation

RAG	Retrieval Augmented Generation
GAIL-PT	Generative Adversarial Imitation Learning-based Penetration Testing
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
DVWA	Damn Vulnerable Web Application
CTF	Capture The Flag

References

1. Hance J, Milbrath J, Ross N, Straub J. Distributed attack deployment capability for modern automated penetration testing. Computers. 2022;11(3):33. doi:10.3390/computers11030033.
2. Altulaihan EA, Alismail A, Frikha M. A survey on web application penetration testing. Electronics. 2023 Mar;12(5):1229. doi:10.3390/electronics12051229.
3. Kaur G, Bharathiraja N, Singh KD, Veeramanickam MRM, Rodriguez CR, Pradeepa K. Emerging trends in cybersecurity challenges with reference to pen testing tools in Society 5.0. In: Artificial intelligence and Society 5.0. USA: Chapman and Hall/CRC; 2023 Dec. p. 196–212. doi:10.1201/9781003397052-18.
4. Auricchio N, Cappuccio A, Caturano F, Perrone G, Romano SP. An automated approach to web offensive security. Comp Comm. 2022 Aug;195(6):248–61. doi:10.1016/j.comcom.2022.08.018.
5. Karlsen E, Luo X, Zincir-Heywood N, Heywood M. Benchmarking large language models for log analysis, security, and interpretation. J Netw Syst Manag. 2024 Jul;32(3):1–10. doi:10.1007/s10922-024-09831-x.
6. Latif E, Zhai X. Fine-tuning ChatGPT for automatic scoring. Comp Educ: Artif Intell. 2024 Jun;8(2):1–10. doi:10.1016/j.caeai.2024.100210.
7. Happe A, Cito J. Getting pwn'd by AI: penetration testing with large language models. In: Proceedings of 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE); 2023 Dec 3–9; San Francisco; CA, USA. New York: ACM; 2023. p. 1–10. doi:10.1145/3611643.3613083.
8. Singh N, Meherhomji V, Chandavarkar BR. Automated versus manual approach of web application penetration testing. In: Proceedings of 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT); Kharagpur, India. New York: IEEE; 2020. p. 1–6. doi:10.1109/icccnt49239.2020.9225385.
9. Qi X, Zeng Y, Xie T, Chen P-Y, Jia R, Mittal P, et al. Fine-tuning aligned language models compromises safety, even when users do not intend to!. In: Proceedings of International Conference on Learning Representations (ICLR); 2024 May 7–11; Vienna, Austria. doi:10.48550/arXiv.2310.03693.
10. Wibowo RM, Sulaksono A. Web vulnerability through cross site scripting (XSS) detection with OWASP security shepherd. Indonesian J Inf Syst. 2021 Feb;3(2):1–10. doi:10.24002/ijis.v3i2.4192.
11. Abdullayev V, Chauhan AS. SQL injection attack: quick view. Mesopotamian J Cyb Sec. 2023 Feb;1(1):1–10. doi:10.58496/mjcs/2023/006.
12. Zhong W. Command injection. OWASP [Online]. [cited 2024 Aug13]. Available from: https://owasp.org/www-community/attacks/Command_Injection.
13. Lachkov P, Tawalbeh L, Bhatt S. Vulnerability assessment for applications security through penetration simulation and testing. J Web Eng. 2022 Oct;21(7):2187–208. doi:10.13052/jwe1540-9589.2178.
14. Yao Y, Duan J, Xu K, Cai Y, Sun Z, Zhang Y. A survey on large language model (LLM) security and privacy: the good, the bad, and the ugly. High-Conf Comp. 2024 Jun;12(2):1–10. doi:10.1016/j.hcc.2024.100211.
15. Deng G, Liu Y, Mayoral-Vilches V, Liu P, Li Y, Xu Y, et al. PentestGPT: an LLM-empowered automatic penetration testing tool. arXiv:2308.06782. 2024 Aug.
16. Pornprasit C, Tantithamthavorn C. Fine-tuning and prompt engineering for large language models-based code review automation. Infor Softw Tech. 2024 Nov;79(11):1–10. doi:10.1016/j.infsof.2024.107523.
17. Upadhyay D, Ware NR. Evolving trends in web application vulnerabilities: a comparative study of OWASP Top 10 2017 and OWASP Top 10 2021. Int J Eng Tech Manag Sci. 2023 Nov–Dec;7(6):262. doi:10.46647/ijetms.2023.v07i06.038.

18. Chen Z, Xu L, Zheng H, Chen L, Tolba A, Zhao L, et al. Evolution and prospects of foundation models: from large language models to large multimodal models. Comput Mater Contin. 2024;74(1):1–20. doi:10.32604/cmc.2024.052618.
19. Vidivelli S, Ramachandran M, Dharunbalaji A. Efficiency-driven custom chatbot development: unleashing langchain, RAG, and performance-optimized LLM fusion. Comput Mater Contin. 2024;74(2):1245–61. doi:10.32604/cmc.2024.054360.
20. Li Y, Liu P, Wang H, Chu J, Wong EW. Evaluating large language models for software testing. Comp Stand Interfaces. 2024;2024(2):103942. doi:10.1016/j.csi.2024.103942.
21. Alahmari SS, Hall LO, Mouton PR, Goldgof DB. Repeatability of fine-tuning large language models illustrated using QLoRA. IEEE Access. 2024;12(140):153221–31. doi:10.1109/ACCESS.2024.3470850.
22. Chen J, Hu S, Zheng H, Xing C, Zhang G. GAIL-PT: an Intelligent penetration testing framework with generative adversarial imitation learning. Comput Secur. 2023;126(4):103055. doi:10.1016/j.cose.2022.103055.
23. Sohail SS, Farhat F, Himeur Y, Nadeem M, Madsen D, Singh Y, et al. Decoding ChatGPT: a taxonomy of existing research, current challenges, and possible future directions. J King Saud Univ–Comput Inf Sci. 2023;35(8):101675. doi:10.1016/j.jksuci.2023.101675.
24. Sohail SS, Farhat F, Himeur Y, Nadeem M, Madsen DØ, Singh Y, et al. The future of GPT: a taxonomy of existing ChatGPT research, current challenges, and possible future directions. SSRN Electron J. 2023. doi:10.2139/ssrn.4413921.
25. Guo Q, Cao J, Xie X, Liu S, Li X, Chen B, et al. Exploring the potential of ChatGPT in automated code refinement: an empirical study. 2023. doi:10.48550/arXiv.2309.08221.
26. Trinity A. Dataset. Kaggle [Online]. [cited 2024 Jul 10]. Available from: https://www.kaggle.com/code/alextrinity/sqli-xss-detection/.
27. Zada I, Alatawi MN, Saqlain SM, Alshahrani A, Alshamran A, Imran K, et al. Fine-tuning cyber security defenses: evaluating supervised machine learning classifiers for windows malware detection. Comput Mater Contin. 2024 Aug;80(2):2917–39. doi:10.32604/cmc.2024.052835.
28. Watson D. Damn vulnerable web application (DVWA) [Online]. [cited 2024 Nov 19]. Available from: https://github.com/digininja/DVWA.
29. Union E. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). Offic J Euro Union. 2016 May;L119:1–88.
30. Organisation for Economic Co-operation. OECD principles on artificial intelligence; 2019 May [Cited 2025 Jan 02]. Available from: https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0449.
31. The IEEE Global Initiative on Ethics of Autonomous, Systems I. Ethically aligned design: a vision for prioritizing human well-being with autonomous and intelligent systems, first edition. IEEE; 2019 [cited 2025 Jan 02]. Available from: https://standards.ieee.org/industry-connections/ec/autonomous-systems.html.
32. NIST. Framework for improving critical infrastructure cybersecurity, Version 1.1 [Online]. U.S. Department of Commerce; 2018. [Cited 2025 Jan 02]. Available from: https://www.nist.gov/cyberframework.