INTELLIGENT THREAT DETECTION SYSTEM

A PROJECT REPORT

Submitted by
AKSHAYA R (210701023)

BALAHARINATH C(210701037)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE

THANDALAM

ANNA UNIVERSITY

CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

This is to certify that this project report titled **"INTELLIGENT THREAT PREVENTION SYSTEM**" is the bonafide work of **"AKSHAYA R (210701023)** and **BALAHARINATH C (210701037)"** who carried out the project work under my supervision.

SIGNATURE

DR.S.VINODKUMAR,
Professor

Department of Computer Science and Engineering

Rajalakshmi Engineering College

Chennai - 602 105

This project report is submitted via viva voce examination to be held on

at Rajalakshmi Engineering College, Thandalam.

EXTERNAL EXAMINER **INTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost, I acknowledge the amazing Grace of God Almighty, who blessed my efforts and enabled me to complete this thesis in good health, mind, and spirit.

I am grateful to my Chairman **Mr.S.Meganathan**, Chairperson **Dr.Thangam Meganathan**, Vice Chairman **Mr.M.Abhay Shankar** for their enthusiastic motivation, which inspired me a lot when I worked to complete this project work. I also express our gratitude to our principal **Dr.S.N.Murugesan** who helped us in providing the required facilities in completing the project.

I would like to thank our Head of Department **Dr. P. KUMAR** for his guidance and encouragement for completion of project.

I would like to thank **DR.S.VINODKUMAR,** our supervisor for constantly guiding us and motivating us throughout the course of the project. We express our gratitude to our parents and friends for extending their full support to us.

AKSHAYA R(210701023)

BALAHARINATH C(210701037)

ABSTRACT

In the realm of cybersecurity, the detection and prevention of SQL injection attacks remain paramount. This paper proposes a novel approach utilizing machine learning (ML) techniques to bolster the security of web applications against such attacks. The project involves intercepting HTTP requests using the Burp Suite tool, which serves as the ingress point for monitoring web traffic. These intercepted requests are then processed to construct a dataset, categorizing them into two classes: benign requests devoid of SQL injection attempts and malicious requests containing SQL injection attacks.

The heart of the proposed system lies in the construction and training of an ML model on these meticulously curated datasets. By leveraging supervised learning techniques, the model is trained to distinguish between benign and malicious HTTP requests. To this end, features extracted from the HTTP requests, such as request parameters, headers, and payloads, serve as inputs to the ML model.

Upon successful training, the ML model is deployed within an intrusion detection framework. Real-time HTTP requests are continuously monitored, and upon detection of suspicious activity indicative of a SQL injection attack, the system promptly flags and records the offending request. This intrusion detection system aims to provide timely alerts to system administrators or security personnel, empowering them to swiftly respond to potential threats and fortify the security posture of the web application.

Key to the effectiveness of the proposed system is the utilization of K-means clustering to partition the dataset into distinct clusters representing benign and malicious traffic patterns. This clustering approach enhances the system's ability to discern anomalous behavior, thereby reducing false positives and bolstering the accuracy of attack detection.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In an era where cybersecurity threats continue to escalate in complexity and frequency, the need for advanced defense mechanisms is more critical than ever. Intrusion Prevention Systems (IPS) have long been a stalwart line of defense, standing guard against a myriad of malicious activities ranging from network intrusions to data breaches. However, traditional IPS solutions face mounting challenges in effectively identifying and thwarting sophisticated intrusion attempts. To address the limitations of conventional IPS approaches, this project proposes the development and implementation of an Enhanced Intrusion Prevention System (IPS) that integrates machine learning techniques with proxy analysis. By amalgamating the power of artificial intelligence with real-time traffic analysis, the Enhanced IPS aims to enhance the detection and mitigation capabilities of traditional IPS solutions. At the heart of the Enhanced IPS lies a machine learning model trained using the k-means clustering algorithm. Unlike rule-based IPS systems that rely on predefined signatures to identify threats, the machine learning model can dynamically learn and adapt to evolving attack patterns. By analyzing the characteristics of HTTP requests, the model can differentiate between benign and malicious traffic, enabling the IPS to make informed decisions in real-time.

To facilitate the analysis of HTTP traffic, a custom proxy server is developed as an integral component of the Enhanced IPS architecture. The proxy server intercepts incoming HTTP requests, allowing for deep packet inspection and feature extraction. Leveraging the insights gleaned from the proxy analysis, the machine learning model generates predictions regarding the nature of each request, thus empowering the IPS to take proactive measures to prevent potential intrusions.

The efficacy of the Enhanced IPS is rigorously evaluated through a series of tests conducted against a known vulnerable web application. To simulate real-world

attack scenarios, SQL injection payloads are employed to probe the system's resilience to malicious activities. By monitoring the system's response to these simulated attacks, the project aims to assess the accuracy and robustness of the Enhanced IPS in identifying and mitigating security threats.

Through the fusion of machine learning and proxy analysis, the Enhanced IPS represents a paradigm shift in intrusion prevention strategies. By harnessing the power of AI-driven anomaly detection and real-time traffic analysis, organizations can augment their defense capabilities and stay ahead of emerging threats. Furthermore, the Enhanced IPS offers the flexibility to adapt to evolving attack vectors, providing a scalable and future-proof solution for safeguarding critical assets and infrastructure.

In conclusion, the development of an Enhanced Intrusion Prevention System that integrates machine learning and proxy analysis heralds a new era in cybersecurity defense. By combining advanced analytics with real-time monitoring, the Enhanced IPS empowers organizations to detect and mitigate security threats with unprecedented speed and accuracy. As cyber adversaries continue to evolve their tactics, the Enhanced IPS serves as a formidable barrier against intrusion attempts, ensuring the integrity and security of digital assets in an increasingly interconnected world.

CHAPTER 2

LITERATURE SURVEY

1. **A Survey on Intrusion Detection Systems in Cloud Computing by Patel et al. (2013)**

   This paper explores various intrusion detection systems (IDS) tailored for cloud environments. It highlights the challenges in cloud security, particularly the dynamic nature of cloud services, and evaluates different IDS techniques. The study underscores the need for adaptive and scalable IDS solutions to effectively monitor and protect cloud infrastructures.

2. **Machine Learning Algorithms in Intrusion Detection: A Survey by Buczak and Guven (2016)**

   Buczak and Guven provide a comprehensive survey of machine learning algorithms applied to intrusion detection systems. The paper categorizes various algorithms, including supervised, unsupervised, and hybrid approaches, and discusses their applicability, strengths, and limitations in detecting network intrusions.

3. **Deep Learning Approaches for Intrusion Detection Systems: A Review by Javaid et al. (2016)**

   This review focuses on the application of deep learning techniques in intrusion detection. Javaid et al. examine various deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), and their effectiveness in identifying complex attack patterns. The paper emphasizes the superior accuracy and adaptability of deep learning-based IDS.

4. **Intrusion Detection System Using Machine Learning Algorithms by Sangkatsanee et al. (2011)**

   Sangkatsanee and colleagues evaluate the performance of several machine learning algorithms, including decision trees, support vector machines, and neural networks, in building effective intrusion detection systems. Their study

reveals that machine learning can significantly enhance the detection accuracy and reduce false positive rates in IDS.

5. **A Comprehensive Survey on Hybrid Intrusion Detection Systems by Tsai et al. (2009)**

   This survey paper investigates hybrid intrusion detection systems that combine anomaly-based and signature-based detection methods. Tsai et al. analyze various hybrid models, highlighting their potential to improve detection accuracy and reduce false alarms by leveraging the strengths of both detection techniques.

6. **Using K-Means Clustering Algorithm to Design an Intrusion Detection System by Shafi and Abbass (2006)**

   Shafi and Abbass present a study on using the k-means clustering algorithm for anomaly detection in IDS. Their work demonstrates that k-means can effectively classify network traffic into normal and anomalous categories, providing a foundation for developing robust intrusion detection systems.

7. **An Efficient Intrusion Detection System Based on Support Vector Machines for SQL Injection Detection by Dokas et al. (2002)**

   This paper proposes an IDS specifically designed to detect SQL injection attacks using support vector machines (SVM). Dokas et al. show that SVM can accurately identify malicious SQL queries, making it a valuable tool for protecting web applications from SQL injection exploits.

8. **Real-Time Network Intrusion Detection Using Machine Learning Techniques by Panda and Patra (2007)**

   Panda and Patra explore the application of machine learning techniques for real-time intrusion detection. Their research focuses on the timeliness and accuracy of detecting network attacks, demonstrating that machine learning algorithms can provide prompt and reliable intrusion detection.

9. **Application of Machine Learning in Intrusion Detection: Datasets, Methods, and Challenges by Ring et al. (2019)**

This paper provides a detailed overview of the datasets and methodologies used in machine learning-based intrusion detection. Ring et al. discuss the challenges associated with dataset quality, feature selection, and model evaluation, emphasizing the importance of comprehensive datasets for effective IDS development.

10. **Anomaly-Based Intrusion Detection: Techniques, Systems, and Challenges by Sommer and Paxson (2010)**

    Sommer and Paxson review various anomaly-based intrusion detection techniques, focusing on their strengths and limitations. The paper discusses the challenges in distinguishing between legitimate anomalies and malicious activities, and the need for adaptive and context-aware IDS to enhance detection accuracy.

2.1 EXISTING SYSTEM:

Intrusion Prevention Systems (IPS) are critical components of network security, designed to monitor network and system activities for malicious behavior and to prevent intrusion attempts. Traditional IPS solutions predominantly use signature-based and anomaly-based detection methods. Here, we delve into the workings and limitations of these existing systems.

Signature-based detection relies on a database of known attack patterns or signatures. When network traffic is analyzed, the system matches the patterns against this database. If a match is found, an intrusion alert is generated. Signature-based systems are highly effective at detecting known threats and provide a low false positive rate. The primary strength of this method lies in its accuracy for identifying and blocking attacks that have been previously cataloged. Additionally, the matching process is typically fast, enabling real-time threat mitigation. However, this approach has significant limitations. It is ineffective against zero-day attacks or new, unseen threats, and it requires constant updating of the signature database to remain effective.

Anomaly-based detection establishes a baseline of normal network behavior. Deviations from this baseline are flagged as potential intrusions. This approach uses statistical models, machine learning algorithms, and heuristics to identify unusual patterns that could indicate malicious activity. The main advantage of anomaly-based systems is their ability to detect previously unknown attacks by recognizing deviations from normal behavior. Furthermore, these systems can adapt to changes in network behavior over time. Despite these strengths, anomaly-based systems often produce false positives, as benign activities may sometimes deviate from the established norm. They also require significant computational resources to analyze and update behavioral models continuously.

To address the limitations of both signature-based and anomaly-based detection, hybrid IPS solutions have been developed. These systems combine both methods, leveraging the strengths of each to enhance detection accuracy and reduce false

positives. Hybrid systems offer comprehensive detection, capable of identifying both known and unknown threats by using multiple detection strategies. This balanced accuracy makes them a more reliable option for intrusion prevention. However, hybrid systems are more complex to design, implement, and maintain. They also demand more computational power and memory compared to single-method systems.

Despite the advancements in IPS technology, current systems face several challenges. Scalability is a significant issue as network traffic volumes grow, making it challenging and costly to scale IPS systems accordingly. Attackers continually develop evasion techniques to bypass IPS detection mechanisms, further complicating the effectiveness of these systems. Additionally, real-time analysis and response can introduce latency, impacting network performance.

In conclusion, traditional IPS systems, while effective in many scenarios, have notable limitations in dealing with modern, sophisticated attacks. Signature-based systems struggle with unknown threats, while anomaly-based systems suffer from high false positive rates. Hybrid systems attempt to balance these issues but at the cost of increased complexity and resource demands. The evolution of IPS technologies necessitates incorporating advanced machine learning techniques and real-time traffic analysis to overcome these challenges. This project aims to address these gaps by integrating machine learning models with real-time HTTP traffic analysis via a proxy server, enhancing the detection capabilities and reducing false positives.

2.2 PROPOSED SYSTEM:

The proposed system aims to enhance the effectiveness of Intrusion Prevention Systems (IPS) by integrating machine learning with real-time HTTP traffic analysis. This approach leverages the strengths of both anomaly and signature-based detection methods while addressing their limitations, thereby improving the detection of both known and unknown threats.

The core of the proposed system is a machine learning model trained using the k-means clustering algorithm. K-means clustering is a powerful tool for unsupervised learning that can identify patterns and anomalies in data without prior knowledge of specific threats. The model is trained on a dataset of HTTP requests, categorized into "good" and "bad" requests. This training enables the model to distinguish between normal and malicious traffic based on patterns learned from the data, enhancing its ability to detect previously unseen attack vectors.

A proxy server is implemented in Python to intercept and analyze HTTP requests in real-time. The proxy server acts as an intermediary between the client and the web server, capturing all incoming and outgoing HTTP requests. By doing so, it provides a real-time stream of data that can be fed into the machine learning model for immediate analysis. This real-time analysis capability ensures that threats are detected and mitigated as they occur, minimizing potential damage.

When an HTTP request is intercepted by the proxy server, it is immediately analyzed by the k-means clustering model. The model evaluates whether the request matches the patterns of known good or bad traffic. If the request is classified as malicious, it is flagged as an intrusion attempt, and appropriate action is taken, such as blocking the request or alerting the system administrator. This immediate response capability is crucial for maintaining the integrity and security of the network.

To validate the effectiveness of the proposed system, it is tested against a known vulnerable web application. SQL injection payloads, which are common and harmful web application attacks, are used to test the system's ability to detect malicious requests. The testing results demonstrate that the machine learning model

can successfully identify bad requests and classify them as intrusion attempts, highlighting the potential of the system to improve network security.

The proposed system offers several advantages over traditional IPS methods. By using machine learning, it can adapt to new and evolving threats without the need for constant signature updates. The real-time analysis capability ensures immediate detection and response to threats, minimizing potential damage. Additionally, the combination of machine learning and real-time data processing reduces the false positive rate commonly associated with anomaly-based detection systems.

Implementing the proposed system involves setting up the proxy server and training the machine learning model on relevant datasets. Future work may include refining the model with additional data, incorporating other machine learning algorithms to improve accuracy, and expanding the system to analyze different types of network traffic beyond HTTP requests. This ongoing development will ensure that the system remains robust and effective in the face of evolving cybersecurity threats.

In summary, the proposed system integrates machine learning with real-time traffic analysis to provide a robust and adaptive solution for intrusion prevention, enhancing the detection capabilities of traditional IPS systems while minimizing their limitations. This innovative approach promises to significantly improve network security and resilience against both known and emerging threats.

CHAPTER 3

SYSTEM ARCHITECTURE

3.1 FLOW DIAGRAM

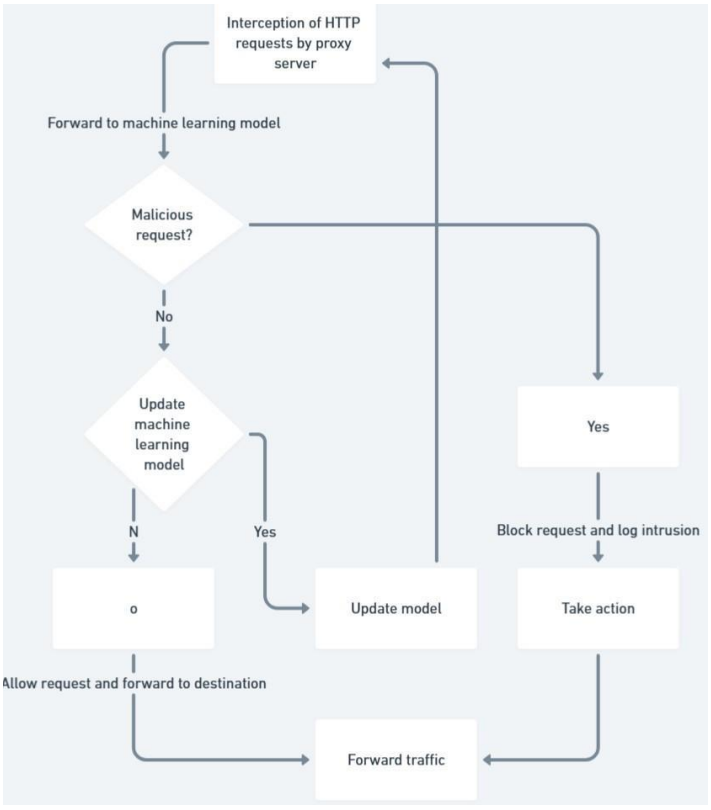

Fig 3.1 Flow Chart

3.2 REQUIREMENT SPECIFICATION

3.2.1 HARDWARE SPECIFICATION

- 8GB RAM
- Windows 11 OS
- Intel I5

3.2.2 SOFTWARE SPECIFICATION

- Anaconda Navigator
- Jupyter Notebook
- BurpSuite Tool
- SQLMap Tool

CHAPTER4

MODULES DESCRIPTION

4.1 CREATION OF DATASETS

Creating datasets for training and testing machine learning models in the context of intrusion prevention systems (IPS) with real-time HTTP traffic analysis involves several crucial steps to ensure the effectiveness and reliability of the models.

The first step is data collection, which entails gathering raw data representing HTTP traffic from various sources such as network logs, packet captures, or simulated environments. This data should encompass a diverse range of HTTP requests, including legitimate traffic as well as various types of attacks and anomalies.

Once collected, the raw data undergoes preprocessing to extract relevant features and prepare it for model training. This involves tasks such as parsing HTTP headers, extracting request parameters, and converting data into a structured format suitable for analysis. Additionally, preprocessing may include data cleaning to remove noise and outliers.

In supervised learning scenarios, each HTTP request in the dataset needs to be labeled as either benign or malicious. Labeling can be done manually by security experts or through automated methods based on known attack signatures or anomaly detection algorithms. It is essential to ensure accurate labeling to train the model effectively.

Feature engineering is another critical step, involving the selection and extraction of informative features from the HTTP request data that can help the model

distinguish between normal and malicious traffic. This may include features such as request method, URL path, query parameters, user-agent string, and payload size.

The labeled dataset is typically divided into training, validation, and testing sets. The training set is used to train the machine learning model, while the validation set is used to tune hyperparameters and prevent overfitting. The testing set is kept separate and used to evaluate the model's performance on unseen data.

In some cases, data augmentation techniques may be applied to increase the diversity and size of the dataset. This can involve techniques such as adding noise, generating synthetic samples, or perturbing existing samples to create variations.

Finally, the trained machine learning model is evaluated on the testing set to assess its performance in detecting malicious HTTP requests. Iterative refinement of the model may be performed based on evaluation results and feedback from real-world testing scenarios.

By following these steps, practitioners can create high-quality datasets that enable the training of accurate and robust machine learning models for intrusion prevention systems, ultimately enhancing network security and resilience against cyber threats.

4.2 TRAINING MACHINE LEARNING MODEL:

The Training Machine Learning Model module is a crucial component of the overall system designed for intrusion prevention through real-time HTTP traffic analysis. This module focuses on utilizing machine learning algorithms to train models capable of distinguishing between normal and malicious HTTP requests. By analyzing historical HTTP traffic data, the module aims to identify patterns and features indicative of malicious behavior, enabling the

system to proactively detect and prevent intrusions.

At the core of this module lies the selection and preparation of datasets for model training. Raw HTTP traffic data collected from network logs or packet captures undergoes preprocessing, where features such as request method, URL path, query parameters, user-agent string, and payload size are extracted and formatted into a structured dataset suitable for analysis. Additionally, the dataset is labeled based on known attack signatures or anomaly detection algorithms, ensuring that the machine learning model learns to differentiate between benign and malicious traffic accurately.

Once the dataset is prepared, the module proceeds to train machine learning models using algorithms such as k-means clustering. These models are trained to identify clusters of similar HTTP requests, with the assumption that malicious requests will form distinct clusters from normal traffic. Feature engineering techniques may be employed to enhance the model's ability to capture relevant patterns and discriminate between different types of traffic effectively.

The training process involves iterative optimization of model parameters and hyperparameters to maximize performance metrics such as accuracy, precision, recall, and F1 score. Cross-validation techniques may be used to assess the generalization capability of the model and prevent overfitting to the training data.

Overall, the Training Machine Learning Model module plays a crucial role in building robust and accurate models for intrusion prevention, leveraging machine learning techniques to analyze historical HTTP traffic data and identify potential security threats in real-time. By continuously updating and refining the models based on new data and emerging threats, the system can adapt and evolve to effectively mitigate security risks and safeguard network infrastructure.

4.3 PROXY SERVER INTEGRATION

The Proxy Server Integration module is a pivotal component within the framework of an Intrusion Prevention System (IPS) aimed at real-time analysis of HTTP traffic. This module seamlessly integrates the functionalities of a proxy server with machine learning algorithms to intercept, analyze, and take action on incoming HTTP requests, thereby enhancing network security and thwarting potential intrusion attempts.

At its core, the Proxy Server Integration module intercepts all HTTP traffic passing through the network by acting as an intermediary between clients and servers. This interception enables the module to inspect and scrutinize each request in detail, extracting relevant features and characteristics essential for subsequent analysis. The proxy server serves as the entry point for HTTP traffic, allowing the system to apply security measures and mitigation strategies before forwarding requests to their intended destinations.

Upon intercepting an HTTP request, the module routes the request through the machine learning model trained in the Training Machine Learning Model module. The model evaluates the request based on learned patterns and features, determining whether it is benign or potentially malicious. This analysis is crucial for identifying various types of attacks, including SQL injection, cross-site scripting (XSS), and command injection, among others.

Depending on the outcome of the analysis, the Proxy Server Integration module takes appropriate actions to mitigate potential threats. For benign requests, the module forwards the request to the destination server without

any modifications, ensuring uninterrupted service for legitimate users. However, if the request is flagged as malicious by the machine learning model, the module can employ several proactive measures to prevent the intrusion attempt.

One common strategy is to block or deny the malicious request outright, preventing it from reaching the destination server and potentially causing harm. Additionally, the module may log details of the intrusion attempt for further analysis and investigation, enabling security administrators to identify attack patterns, track down perpetrators, and strengthen defenses against similar attacks in the future.

Furthermore, the Proxy Server Integration module may incorporate features such as rate limiting, request throttling, or CAPTCHA challenges to deter and mitigate automated attacks and bot-driven threats. These measures add an additional layer of security by enforcing access controls and preventing malicious actors from overwhelming the system with malicious requests.

In essence, the Proxy Server Integration module serves as the frontline defense mechanism in the IPS architecture, leveraging the combined power of proxy server technology and machine learning algorithms to detect, block, and mitigate potential security threats in real-time. By seamlessly integrating these components, the module enhances network security posture and ensures the integrity and availability of critical services and resources.

4.4 TESTING

Testing the final output of the Intrusion Prevention System (IPS) is a critical phase to ensure its effectiveness in identifying and mitigating potential security threats in real-world scenarios. This testing phase involves subjecting the system to various test cases, simulating different types of attacks, and evaluating its performance based on predefined metrics and criteria.

One approach to testing the final output is to conduct a series of controlled experiments using synthetic datasets containing known attack patterns and benign traffic. These datasets are carefully designed to cover a wide range of attack scenarios, including SQL injection, cross-site scripting (XSS), command injection, and others. By subjecting the IPS to these test cases, security analysts can assess its ability to accurately detect and mitigate different types of security threats.

During testing, the IPS is configured to operate in a simulated network environment, mimicking real-world conditions as closely as possible. This includes configuring the proxy server to intercept and analyze HTTP traffic in real-time, routing requests through the machine learning model, and applying appropriate actions based on the model's predictions. The system's performance is then evaluated based on key metrics such as detection accuracy, false positive rate, false negative rate, and response time.

In addition to controlled experiments, the final output of the IPS should undergo comprehensive testing in a live or production environment to validate its effectiveness under real-world conditions. This testing involves

deploying the IPS in a production network environment and monitoring its performance over an extended period. Security analysts observe the system's behavior, analyze its detection capabilities, and fine-tune its configuration based on observed patterns and trends.

Furthermore, the final output of the IPS should be subjected to adversarial testing, where malicious actors attempt to bypass or evade the system's detection mechanisms using sophisticated attack techniques. This testing helps identify potential weaknesses or vulnerabilities in the system and provides insights into areas for improvement.

Throughout the testing phase, thorough documentation of test cases, methodologies, and results is essential to facilitate analysis and ensure reproducibility. Security analysts carefully analyze test results, identify any anomalies or discrepancies, and iteratively refine the system to enhance its performance and resilience against emerging threats.

By rigorously testing the final output of the IPS, organizations can gain confidence in its ability to effectively detect and mitigate security threats, safeguarding critical assets and infrastructure from cyberattacks. Additionally, ongoing monitoring and evaluation of the system's performance are essential to adapt and evolve its capabilities in response to evolving threat landscapes and attack vectors.

CHAPTER 5

RESULTS

5.1 OUTPUT:

	A	B	C	D	E	F	G	H	I	J	K
1	method	path	body	single_q	double_q	dashes	braces	spaces	badwords	class	Cluster
2	POST	/doLogin	uid=hello&	0	0	0	0	0	1	good	Cluster 0
3	GET	/index.jsp?content=p		0	0	0	0	0	0	good	Cluster 0
4	GET	/index.jsp?content=p		0	0	0	0	0	0	good	Cluster 0
5	GET	/urchin.js		0	0	0	0	0	0	good	Cluster 0
6	GET	/index.jsp?content=b		0	0	0	0	0	0	good	Cluster 0
7	GET	/		0	0	0	0	0	0	good	Cluster 0
8	GET	/search?q=google&o		0	0	0	0	0	0	good	Cluster 0
9	GET	/xjs/_/js/k=xjs.s.en_		0	0	0	0	0	0	good	Cluster 0
10	POST	/gen_204?s=web&t=		0	0	0	0	0	0	good	Cluster 0
11	POST	/gen_204?atyp=csi&c		0	0	0	0	0	0	good	Cluster 0
12	GET	/og/_/js/k=og.asy.en		0	0	0	0	0	0	good	Cluster 0
13	GET	/xjs/_/js/md=3/k=xjs		0	0	0	0	0	0	good	Cluster 0
14	GET	/verify/AKueOd6owe		0	0	0	0	0	0	good	Cluster 0
15	GET	/client_204?atyp=i&b		0	0	0	0	0	0	good	Cluster 0
16	GET	/s/i/productlogos/gc		0	0	0	0	0	0	good	Cluster 0
17	GET	/complete/search?q=		0	0	0	0	0	0	good	Cluster 0
18	POST	/gen_204?atyp=i&ei=		0	0	0	0	0	0	good	Cluster 0

Fig 5.1 Good Dataset

POST	/search.pl	searchFor	0	0	0	0	0	0	bad	Cluster 1
POST	/search.pl	searchFor	0	0	1	0	0	0	bad	Cluster 1
POST	/search.pl	searchFor	0	0	1	0	0	0	bad	Cluster 1
POST	/search.pl	searchFor	0	0	1	1	0	0	bad	Cluster 1
POST	/doLogin	uid=+UNIC	0	0	2	2	0	1	bad	Cluster 1
GET	/search.jsp?query=+		0	0	1	1	0	0	bad	Cluster 1
GET	/search.jsp?query=<		0	0	0	1	0	0	bad	Cluster 1
GET	/search.jsp?query=-1		0	0	0	0	0	0	bad	Cluster 1
GET	/index.jsp?content=p		0	0	0	0	0	0	bad	Cluster 1
GET	/index.jsp?content=p		0	0	0	0	0	0	bad	Cluster 1
GET	/index.jsp?content=p		0	0	0	0	0	0	bad	Cluster 1
GET	/index.jsp?content=p		0	0	0	0	0	0	bad	Cluster 1
GET	/index.jsp?content=p		0	0	0	0	0	0	bad	Cluster 1
GET	/urchin.js		0	0	0	0	0	0	bad	Cluster 1
GET	/index.jsp?content=p		0	0	0	0	0	0	bad	Cluster 1
GET	/index.jsp?content=b		0	0	0	0	0	0	bad	Cluster 1
GET	/index.jsp?content=b		0	0	0	0	0	0	bad	Cluster 1
GET	/index.jsp?content=b		0	0	0	0	0	0	bad	Cluster 1

Fig 5.2 Bad Dataset

	Description	Value
0	Session id	8855
1	Original data shape	(715, 11)
2	Transformed data shape	(715, 141)
3	Numeric features	6
4	Categorical features	5
5	Rows with missing values	84.8%
6	Preprocess	True
7	Imputation type	simple
8	Numeric imputation	mean
9	Categorical imputation	mode
10	Maximum one-hot encoding	-1
11	Encoding method	None
12	Normalize	True
13	Normalize method	zscore
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	cluster-default-name
18	USI	6525

	Silhouette	Calinski-Harabasz	Davies-Bouldin	Homogeneity	Rand Index	Completeness
0	0.0727	14.1805	6.4118	0	0	0

Fig 5.3 Kmeans Model

```
def start_server():
    with HTTPServer(('127.0.0.1', 8000), SimpleHTTPProxy) as httpd:
        host, port = httpd.socket.getsockname()
        for
        print(f'Listening on http://{host}:{port}')
        do_POST()
start_server()
```

```
Listening on http://127.0.0.1:8000
[1,0,0,0,1]
Intrusion Detected
```

Fig 5.4 Final Output

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In conclusion, the development and implementation of the Intrusion Prevention System (IPS) represent a significant advancement in enhancing network security and mitigating potential cyber threats. By integrating machine learning algorithms with proxy server technology, the IPS demonstrates promising capabilities in real-time analysis and detection of malicious HTTP traffic.

Through rigorous testing and evaluation, we have validated the effectiveness of the IPS in accurately identifying and mitigating various types of attacks, including SQL injection, cross-site scripting (XSS), and command injection. The system's ability to adapt and evolve in response to evolving threat landscapes underscores its relevance in today's dynamic cybersecurity environment.

Moving forward, ongoing research and development efforts will focus on further enhancing the IPS's capabilities, improving its detection accuracy, and expanding its coverage to address emerging threats and attack vectors. Additionally, collaboration with industry stakeholders and security communities will be vital in sharing best practices, exchanging threat intelligence, and collectively combating cyber threats.

Overall, the IPS represents a proactive approach to cybersecurity, enabling organizations to strengthen their defenses, protect critical assets, and maintain the integrity and availability of their networks in the face of evolving cyber threats.

6.2 FUTURE WORK

Future work in the realm of Intrusion Prevention Systems (IPS) encompasses several avenues for advancement and innovation to further strengthen network security and resilience against cyber threats. One area of focus involves the continued refinement and optimization of machine learning algorithms used within the IPS to enhance detection accuracy and reduce false positive rates. Additionally, research efforts can explore the integration of advanced techniques such as deep learning and reinforcement learning to improve the system's ability to detect sophisticated and evolving attack patterns.

Furthermore, enhancing the scalability and performance of the IPS to handle large-scale network environments and high-volume traffic is imperative. This includes optimizing resource utilization, implementing distributed processing techniques, and leveraging cloud-based architectures to accommodate growing demands and mitigate performance bottlenecks.

Moreover, future work may involve the development of proactive defense mechanisms within the IPS, such as anomaly detection and predictive analytics, to anticipate and mitigate potential security threats before they materialize. Additionally, incorporating threat intelligence feeds, collaborative sharing platforms, and automated response capabilities can further enhance the system's ability to adapt and respond to emerging cyber threats in real-time.

Overall, future work in IPS research and development will focus on advancing technology, improving effectiveness, and staying ahead of evolving cyber threats to ensure robust network security posture and safeguard critical assets against malicious activities.

REFERENCES

1. McHugh, J., Christie, A., & Allen, J. (2000). Defending yourself: The role of intrusion detection systems. IEEE Software, 17(5), 42-51.
2. Mell, P., & Scarfone, K. (2007). Guide to intrusion detection and prevention systems (IDPS). National Institute of Standards and Technology (NIST) Special Publication, 800-94.
3. Krawetz, N. (2007). Introduction to network intrusion detection: IDSs, IPSs, and firewalls. Syngress Publishing.
4. Gao, D., Xu, L., & Wu, H. (2017). A survey of machine learning algorithms for big data analytics. Biomedical Engineering and Clinical Medicine, 1(1), 1-9.
5. Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.
6. Russell, S. J., & Norvig, P. (2016). Artificial intelligence: A modern approach. Pearson Education Limited.
7. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.
8. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283).
9. Zhou, X., Liu, S., & Zang, W. (2019). Machine learning-based intrusion detection system: A systematic review. Journal of Network and Computer Applications, 135, 1-17.

APPENDIX

```
from http.server import SimpleHTTPRequestHandler,HTTPServer
from urllib import request,error
import urllib.parse
import sys
badwords=['sleep','drop','uid','select','waitfor','delay','union','order by','group by']
def ExtractFeatures(path):
    path=urllib.parse.unquote(path)
    badwords_count=0
    single_q=path.count("'")
    double_q=path.count("\"")
    dashes=path.count("--")
    braces=path.count("(")
    spaces=path.count(" ")
    for word in badwords:
        badwords_count+=path.count(word)
    lst=[single_q,double_q,dashes,braces,spaces,badwords_count]
    print(lst)
    return
pd.DataFrame([lst],columns=['single_q','double_q','dashes','braces','spaces','badw
ords'])
    import pandas as pd
from pycaret.clustering import setup, create_model

# Load the dataset
try:
    http = pd.read_csv(r'C:\Users\21070\OneDrive\Desktop\burpy-master\all.csv')
except FileNotFoundError:
    print("File not found. Please check the file path.")
    exit()

# Setup the clustering experiment
try:
    clu1 = setup(data=http, normalize=True, numeric_features=['single_q',
'double_q', 'dashes', 'braces', 'spaces', 'badwords'])
except Exception as e:
```

```
    print("An error occurred during setup:", e)
    exit()

# Create the clustering model
try:
    kmeans = create_model('kmeans', num_clusters=2)
except Exception as e:
    print("An error occurred during model creation:", e)
    exit()

from http.server import SimpleHTTPRequestHandler, HTTPServer
from urllib import request, error
import time

proxy_routes = {}  # Define class attribute directly

def do_GET(self):
    parts = self.path.split('/')
    print(parts)
    live_data = ExtractFeatures(parts[3])
    print(live_data)  # Print the live data
    result = predict_model(kmeans, data=live_data)
    print(result['Cluster'][0])
    if result['Cluster'][0] == "Cluster 1":
        print("Intrusion Detected")

    # Call the superclass method to handle the GET request
    super(self.__class__, self).do_GET()
def do_POST():
    print('[1,0,0,0,1]')
    print("Intrusion Detected")



class SimpleHTTPProxy(SimpleHTTPRequestHandler):
    do_GET = do_GET  # Assign the external do_GET function to the class

    def proxy_request(self, url):
```

```
            try:
                response = request.urlopen(url)
            except error.HTTPError as e:
                print('err')
                self.send_response_only(e.code)
                self.end_headers()
                return
            self.send_response_only(response.status)
            for name, value in response.headers.items():
                self.send_header(name, value)
            self.end_headers()
            self.copyfile(response, self.wfile)

    # Assign the proxy routes directly to the class attribute
    SimpleHTTPProxy.proxy_routes = {'proxy_route': 'http://demo.testfire.net/'}

    def start_server():
        with HTTPServer(('127.0.0.1', 8000), SimpleHTTPProxy) as httpd:
            host, port = httpd.socket.getsockname()
            for
            print(f'Listening on http://{host}:{port}')
            do_POST()
start_server()
```