**A Project Report**

**On**

**Signature and Anomaly based Web Application Firewall**

submitted for partial fulfillment of the requirements

for the award of the degree of

Bachelor of Technology

in

Computer Science

**Submitted by**

Prachi Sharma 2000290120108

Manya Varshney 2000290120093

Priyansha Singhal 2000290120116

**Under supervision of**

Mr. Abhishek Goyal

Assistant Professor



**KIET Group of Institutions, Ghaziabad**

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow**

**May, 2024**

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** | **Page No.** |
| DECLARATION……………………………………………………………………. | ii |
| CERTIFICATE……………………………………………………………………… | iii |
| ACKNOWLEDGEMENTS…………………………………………………………. | iv |
| ABSTRACT………………………………………………………………………..... | v |
| LIST OF FIGURES…………………………………………………………………. | vi |
| LIST OF TABLES…………………………………………………………………… | vii |
| LIST OF ABBREVIATIONS………………………………………………………. | viii |
|  |  |
| CHAPTER 1 INTRODUCTION  1.1 INTRODUCTION TO PROJECT  1.2 PROJECT CATEGORY  1.3 OBJECTIVES  1.4 STRUCTURE OF THE REPORT  CHAPTER 2 LITERATURE REVIEW  2.1 LITERATURE REVIEW  2.2 RESEARCH GAP  2.3 PROBLEM FORMULATION  CHAPTER 3 PROPOSED SYSTEM  3.1 PROPOSED SYSTEM  3.2 UNIQUE FEATURES OF THE SYSTEM  CHAPTER 4. REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION  4.1 FEASIBILITY STUDY (TECHNICAL, ECONOMICAL, OPERATIONAL)  4.2 SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT WHICH MUST INCLUDE THE FOLLOWING:  4.2.1 DATA REQUIREMENT  4.2.2 FUNCTIONAL REQUIREMENT  4.2.3 PERFORMANCE REQUIREMENT  4.2.4 MAINTAINABILITY REQUIREMENT  4.2.5 SECURITY REQUIREMENT)  4.3 SDLC MODEL TO BE USED  4.4 SYSTEM DESIGN  4.4.1 DFD LEVEL 0 AND LEVEL 1  4.4.2 USE CASE DIAGRAM  4.5 DATABASE DESIGN  CHAPTER 5. IMPLEMENTATION, TESTING, AND MAINTENANCE  5.1 INTRODUCTION TO LANGUAGES,TOOLS AND TECHNOLOGIES USED FOR IMPLEMENTATION  CHAPTER 6 TESTING AND MAINTAINENCE  6.1 TESTING TECHNIQUES AND TEST CASES USED  CHAPTER 7. RESULTS AND DISCUSSIONS  7.1 DESCRIPTION OF MODULES WITH SNAPSHOTS  7.2 KEY FINDINGS OF THE PROJECT  7.3 BRIEF DESCRIPTION OF DATABASE WITH SNAPSHOTS    CHAPTER 8. CONCLUSION AND FUTURE SCOPE  REFERENCES( REFERENCES SHOULD BE IN IEEE STANDARD)  PROOF OF PATENT PUBLICATION  ACCEPTANCE PROOF | 1  3  4  5  5  7  7  8  14  17  19  19  20  20  21  22  22  23  23  24  24  25  28  42  43  60  62  65  67  68 |

**DECLARATION**

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature of Students

Priyansha Singhal Manya Varshney Prachi Sharma

(2000290120116) (2000290120093) (2000290120108)

**Date:- 14-05-2024**

## CERTIFICATE

This is to certify that Project Report entitled “ Signature and Anomaly based web application firewall ” which is submitted by Manya Varshney , Prachi Sharma and Priyansha Singhal in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

**Date: 14-05-2024 Supervisor**

Mr. Abhishek Goyal

**ACKNOWLEDGEMENT**

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Professor (Mr. Abhishek Goyal), Department of Computer Science, KIET, Ghaziabad, for his constant support and guidance throughout the course of our work. Her sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. Ajay Srivastava , Head of the Department of Computer Science, KIET, Ghaziabad, for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project.

Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

**ABSTRACT**

Attacks on web applications and web-based services were conducted using Hyper-Text Transfer Protocol (HTTP), which is also used as the communication protocol of web-based applications. Due to the dynamic structure of web applications and the fact that they have many variables, detection and prevention of web-based attacks are made more difficult. In this study, a hybrid learning-based web application firewall (WAF) model is proposed to prevent web-based attacks, by using signature-based detection (SBD) and anomaly-based detection (ABD). SBD is a rule-based model, defined as misuse. It is revealed by examining the characteristics, behaviors, and content of the attack. Signature-based methods generally work fast and are effective against attack types listed in the signature database. In general, intrusion detection systems work on the basis of a signature database . When a new attack technique is developed, the signature database needs to be updated in order to set the system to be efficient. Detection of known web-based attacks is done by using SBD, while detection of anomaly HTTP requests is done by using ABD.

Learning-based ABD is implemented by using Artificial Neural Networks (ANN). Thus, an adaptation of the model against zero-day attacks is ensured by learning-based ABD by using ANN. a. In this project, we used methods based on deep-neural-network and parallel- feature-fusion that features engineering as an integral part of them and plays the most important role in their performance. The proposed methods use stacked autoencoder and deep belief network as feature learning methods. . Results show that deep model and feature fusion model demonstrated hierarchical feature learning which had better performance in terms of accuracy and generalization in a reasonable time.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Description** | **Page No.** |
| 1 | Dataflow Diagram | 23 |
| 2 | Use Case Diagram | 24 |
| 3  4  5  6  7  8  9  10 | Testing for a SQL query  Result  Entry point of project  Port number for server request  Server starting on given port  Screenshot of a proxy server  Jupyter notebook machine learning model  Model gives results based on SQL query | 41  41  42  42  43  52  55  58 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table. No.** | **Description** | **Page No.** |
| 1.1 | Database CSIC 2010 | 60 |

**LIST OF ABBREVIATIONS**

WAF Web Application Firewall

HTTP Hyper Text Transfer Protocol

ANN Artificial Neural Network

SBDNR Signature based Detection of Normal Request

SBDAR Signature based Detection of Anomaly Request

SBDKIT Signature based Detection of Known Intrusion Types

TCP/IP Transmission Control Protocol/Internet Protocol

MSE Mean Squared Error

**CHAPTER 1**

**INTRODUCTION**

**1.1 INTRODUCTION TO PROJECT**

A web application firewall (WAF) is a security solution designed to protect web applications from a variety of online threats, such as cross-site scripting (XSS), SQL injection, and other types of attacks. It operates by monitoring and filtering HTTP traffic between a web application and the Internet, acting as a barrier between the two.

Web Application Firewalls (WAFs) play a crucial role in safeguarding web applications from malicious attacks and threats. They employ various detection techniques to filter and block harmful traffic, ensuring the security and integrity of web applications. One of the fundamental approaches in WAFs is signature-based detection, which relies on predefined patterns or signatures to identify and thwart known attack patterns. Additionally, anomaly-based detection provides an extra layer of security by scrutinizing URL structures and query parameters for irregularities and suspicious behavior.

In signature-based detection, security experts create signatures that represent known vulnerabilities and attack patterns. These signatures are then employed to scan incoming and outgoing HTTP traffic, enabling WAFs to take predefined actions when a match is found. Regular updates to the signature database are essential to stay current with the evolving threat landscape. Signature-Based Detection of Normal Requests (SBDNR) is an integral stage where incoming HTTP requests are matched against a list of normal signatures. Requests identified as normal in this stage are added to the normal signature list. When these requests reappear in the web application, they are directly passed through, enhancing overall performance.

In contrast, Signature-Based Detection of Anomaly Requests (SBDAR) focuses on HTTP requests previously recognized as anomalies. When these previously flagged requests resurface, SBDAR can block them without the need for additional anomaly-based detection. Anomaly-based detection takes a closer look at the structure of URLs, analyzing query parameters and absolute paths. This approach is vital for detecting subtle variations that could indicate a malicious intent. It involves training Recurrent Neural Networks (RNNs) on tokenized normal URLs, allowing them to learn patterns and recognize deviations from the norm. An MLP (Multilayer Perceptron) further refines the decision-making process, classifying URLs as normal or anomalous based on the output of the RNNs. The HTTP dataset CSIC 2010, comprising both regular and abnormal requests directed at an e-commerce website, serves as the foundation for our research.

This dataset, consisting of over 25,000 abnormal queries and 36,000 regular requests, is randomly split into training and test sets, providing the empirical basis for our experimentation and findings. In this research, we explore the effectiveness of signature-based and anomaly-based detection techniques in securing web applications and contribute to the evolving field of web application security.

A web application firewall (WAF) functions through a systematic process starting with traffic monitoring, where it inspects incoming and outgoing web traffic, analyzing various parameters like HTTP headers, cookies, query strings, and request payloads. Following this, the WAF applies rule-based filtering, making decisions on whether to allow, block, or log each request based on predefined rules and policies tailored to the specific security needs of the web application. It offers protection against a broad spectrum of threats, encompassing SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), among others, by identifying malicious patterns and taking appropriate action to mitigate risks. Regular updates to rule sets are crucial for WAFs to stay effective against evolving threats, ensuring they can recognize and block new attack patterns as they emerge.

Additionally, WAFs maintain detailed logs of web traffic and security events for forensic analysis, compliance auditing, and threat detection. While ensuring robust security measures, WAFs also focus on performance optimization to minimize latency and maintain high performance, employing techniques such as caching and content delivery networks (CDNs). Deployment options for WAFs vary, including hardware appliances, software solutions, virtual appliances, and cloud-based services, each with its own advantages and considerations in terms of scalability, management overhead, and cost. Overall, a web application firewall serves as a critical component of an organization's cybersecurity infrastructure, safeguarding sensitive data, ensuring regulatory compliance, and preserving the reputation of web applications and their users.

**1.2 PROJECT CATEGORY**

The project category of a Web Application Firewall (WAF) typically falls within the broader domain of cybersecurity or network security. WAF projects are primarily aimed at enhancing the security posture of web applications by protecting them against various web-based attacks, vulnerabilities, and threats. They contribute to network security efforts by inspecting and filtering web traffic at the application layer, thereby mitigating risks associated with unauthorized access, data breaches, and malicious activities. Moreover, WAF projects are closely related to application security initiatives, addressing vulnerabilities and security risks specific to web applications, such as injection attacks, cross-site scripting (XSS), and other common exploits.

They play a crucial role in compliance efforts, particularly for organizations subject to regulatory requirements such as PCI DSS, HIPAA, GDPR, or industry-specific regulations governing data security and privacy. In addition to risk management strategies, WAF projects contribute to incident response and threat management by detecting and blocking malicious activities in real-time. Integrating WAFs with incident response processes, threat intelligence feeds, and security information and event management (SIEM) systems enhances proactive threat detection and response capabilities. Furthermore, with the increasing adoption of cloud-based applications and services, WAF projects may focus on securing web applications hosted in cloud environments. Deploying WAFs in cloud infrastructures, such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP), helps protect cloud-hosted applications from cyber threats and ensures the confidentiality, integrity, and availability of application resources.

Overall, the project category of a Web Application Firewall encompasses various aspects of cybersecurity, network security, application security, compliance, risk management, incident response, and cloud security, depending on the specific objectives and scope of the project.

**1.3 OBJECTIVES**

The purpose of this project is to develop a web application firewall that uses Signature-Based Detection (SBD), Anomaly-Based Detection (ABD) and Artificial Neural Networks (ANN) as one of the artificial intelligence techniques. By using SBD, the detections were made against known web-based attack types such as SQL (Structured Query Language) Injection, Cross-Site Scripting (XSS), Command Injection, and Directory Traversal Attacks. HTTP requests that do not conform to the structure of the web application architecture were detected using ABD.

**CHAPTER 2**

**LITERATURE REVIEW**

**2.1 LITERTURE REVIEW**

Abdul Razzaq [1] examines Web Application Firewall (WAF) solutions, recommending F5, Barracuda, Web Sniper, i-Sentry, Secure IIS, Easy Guard, Web Defend, Secure Sphere, Anchiva, Citrix, WebApp secure, eServer Secure, Server Defender AI, and Mod Security. [2] A Debian implementation of the Snort intrusion detection system, which effectively identifies and analyzes real-time network traffic, alerting security personnel is used for necessary action.[5] This work developed a hybrid system using signature-based and anomalous request detection, reducing deficiencies in both approaches, proving faster but ineffective against zero-day attacks. [7] Therefore a web application firewall model using features engineering and machine learning to identify common web assaults by analyzing HTTP request components and categorizing them as normal or abnormal. [17] Researchers explore methods to detect and prevent security vulnerabilities in web applications, including anomaly-based and misuse-based detection mechanisms, to protect servers from malicious activities.

[8] The study proposes a feature learning method using deep neural networks and isolation forest classifier, demonstrating its superior accuracy over feature-free techniques on the CSIC 2010 dataset.[9] deep learning models with LSTM layers for analysing character sequences of malicious web application payloads, identifying optimal structures for detection and is determining benign or malicious input words.[10] The study introduces deep neural network and parallel feature fusion approaches, focusing on feature engineering, and uses one-class SVM, isolation forest, and elliptic envelope classifiers. [11] A hybrid learning-based WAF model, utilizing signature-based and anomaly-based detection, prevent web-based attacks, achieving a high mean percentage of 96.59%.[12]A layered architecture model for detecting DDoS, XSS, and SQL injection attacks with 97.57% accuracy, utilizing LSTM deep learning and a web application firewall.

[13] A novel deep learning approach using Recurrent Neural Networks and a classifier is proposed for detecting anomalous requests in web applications, offering a competitive and feature-free method.[14] or an unsupervised/semi-supervised web attack detection method using RSMT, demonstrating autoencoders' efficiency and accuracy, but further research is needed for zero-day attacks.[15] This study reviews and analyses network attack detection technology's evolution, focusing on deep learning techniques to address data category imbalance, high-dimensional processing, concept distribution drift, real-time interpretability, and security issues.[16] Machine learning techniques improve Web Application Firewalls' detection and accuracy, outperforming MODSECURITY when configured with OWASP Core Rule Set and enhancing resilience to zero-day attacks. [6] This paper examines the effectiveness of machine learning-based techniques in enhancing WAFs, highlighting their advantages over signature and rule-based methods, including zero-day attack defense.

[18] A systematic mapping study analyzed 41 studies from 1994-2014 on web application security vulnerability detection, identifying strengths, weaknesses, best practices, and directions for future research.[19] WAF-A-MoLE uses mutation operators to model adversary presence, but machine learning-based WAFs face bypass risk. Future directions include testing techniques, finding mutations, and improving detection.[20] This research paper analyzes the evolution of NGFWs and WAFWs, highlighting their characteristics and future security needs for enterprises reliant on web-based applications.[4]A method for deploying WAF on a web-based application, demonstrating its effectiveness in preventing cross-site scripting, SQL injection, and unauthorized vulnerability web scanning. [3] This study discusses the importance of a Web Application Firewall (WAF) and its advantages and disadvantages, emphasizing the need for security testing to address vulnerabilities.

**2.2 RESEARCH GAPS**

Research gaps in the realm of Web Application Firewall (WAF) implementation encompass a spectrum of critical areas. These include assessing the effectiveness of advanced threat detection techniques like machine learning and behavioral analysis, particularly in the context of emerging threats and zero-day attacks. Scalability and performance optimization strategies for WAF deployments, especially in high-traffic or cloud-based environments, demand further exploration to balance security controls with operational efficiency.

Integrating WAFs into DevOps and CI/CD pipelines remains an area ripe for investigation, emphasizing the need for seamless security automation within agile development workflows. Moreover, research into comprehensive threat modeling, attack simulation exercises, and user-centric security approaches can enhance the efficacy of WAFs by identifying realistic attack vectors and differentiating between legitimate user activities and malicious behavior. Understanding evasion tactics and bypass techniques employed by attackers to circumvent WAF protections is crucial for bolstering WAF resilience and efficacy.

Lastly, navigating the evolving landscape of regulatory compliance and industry standards, and examining how WAFs can aid organizations in meeting these requirements, presents a pertinent avenue for research in ensuring robust security postures for web applications.

**2.3 PROBLEM FORMULATION**

An unprotected website is a security risk to customers, other businesses, and public/government

sites. It allows for the spread and escalation of malware, attacks on other websites, and even attacks against national targets and infrastructure. In many of these attacks, hackers will try to harness the combined power of thousands of computers and sites to launch this attacks, and the attacks rarely lead directly back to the hackers.

**CHAPTER 3**

**PROPOSED SYSTEMS**

**3.1 PROPOSED SYSTEM**

**Signature-Based Detection of Normal Requests (SBDNR).**

SBDNR is a signature-based detection stage where HTTP requests directed at web applications are normally detected through ABD. Normally, the detected HTTP requests are added to the normal signature lists so when they appear in the web application again, they are directed to the web application without running the anomaly detection process. As such, an increased speed of its performance can be ensured.

**Signature-Based Detection of Known Intrusion Types (SBDKIT)**

Known attacks are the web-based attack types that target web applications revealing their weaknesses by using security vulnerabilities of web applications and containing intrusion qualities. In this study, the most common web-based attack types are used as an example. These are: SQL Infection, Cross-Site Scripting (XXS), Directory Traversal Attack, Command Infection and Signature-Based Detection of Anomaly Requests (SBDAR).

**SQL Injection:** SQL is a structural language specialized in querying within database management systems. Relational database applications in various sizes are reached through SQL queries. Many database management systems that support SQL apply special add-ons in the standard language. Web applications may be used to generate different SQL sentences for user originated inputs and requests.

The SQL injection method is a penetration test. It allows for misuse of SQL sentences using vulnerabilities that arise from user inputs not being verified or being verified inadequately. If the web application does not efficiently detect user requests, the structure of the SQL sentences inside the web application could be changed; a database management system penetrated and privileges of the web application’s administrator could be captured through the SQL Injection method. When performing a SQL injection, characters containing special meanings unique to SQL need to be used.

**Cross-Site Scripting (XSS)** : It is generally defined as running desired client originated scripts in users browser by means of adding client-originated scripts inside HTML codes. XSS are generally written in HTML/JavaScript languages; however, scripting can also be done in VBScript, ActiveX, Java, Flash or other languages that are supported by web applications. Cross-Site Scripting (XSS) is a prevalent cybersecurity vulnerability characterized by the injection of malicious client-originated scripts into web applications, typically within HTML code.

This exploitation enables attackers to execute arbitrary scripts in users' browsers, leading to various malicious activities such as data theft, session hijacking, and defacement of web pages. While XSS attacks are commonly crafted using HTML and JavaScript languages due to their ubiquity in web development, attackers can leverage alternative scripting languages supported by web applications, including VBScript, ActiveX, Java, and Flash. By exploiting XSS vulnerabilities, attackers can manipulate the functionality of web applications to execute unauthorized actions, compromise user privacy, and undermine the integrity of web content, emphasizing the critical importance of implementing robust security measures to mitigate the risk of XSS attacks.

**Command Injection:** A command injection attack is an intrusion type aimed at running scripts on the operating system containing the application through an unprotected application. These attacks are only possible when an application runs on system shell user-originated data that are unsafe (forms, cookies, HTTP Headers etc).

A command injection attack represents a type of cyber intrusion designed to execute unauthorized scripts on the underlying operating system by exploiting vulnerabilities in an unprotected application. This attack vector typically targets applications that utilize user-originated data without adequate validation or sanitization measures, particularly within interactive elements such as forms, cookies, and HTTP headers.

By manipulating input fields or parameters, attackers can inject malicious commands directly into the system shell, bypassing security controls and executing arbitrary code with the privileges of the application or user. Command injection attacks pose a significant threat to the security and integrity of web applications, as they can lead to a range of detrimental consequences including data exfiltration, system compromise, and unauthorized access to sensitive resources. Mitigating the risk of command injection requires implementing robust input validation and sanitization mechanisms, ensuring that user-supplied data is properly sanitized and validated before being processed by the application, thereby thwarting attempts to execute malicious commands and fortifying the overall security posture of the system.

**Signature-Based Detection of Anomaly Requests (SBDAR)**

SBDAR, or Signature-Based Detection and Response, serves as a crucial mechanism within web application security frameworks, particularly in the context of anomaly detection. It functions as a secondary layer of defense, designed to complement the Anomaly-Based Detection (ABD) process by swiftly identifying and blocking HTTP requests flagged as anomalies. When HTTP requests undergo anomaly detection and are subsequently identified as anomalous, they are rerouted back into the web application environment for further evaluation. However, if these requests are detected as anomalies once again upon re-entry, they are subject to immediate blocking through the SBDAR protocol, bypassing additional ABD processes.

The primary objective of SBDAR is to provide a rapid and decisive response to potentially malicious HTTP requests that have already been identified as anomalous. By leveraging signature-based detection techniques, SBDAR aims to quickly match incoming HTTP requests against a database of known attack signatures or patterns associated with malicious activity. These signatures are meticulously curated by security researchers and vendors to encompass a wide range of common attack vectors, including SQL injection, cross-site scripting (XSS), and directory traversal.

Upon receiving an HTTP request flagged as an anomaly, the SBDAR system initiates a real-time inspection of the request payload, headers, parameters, and other relevant attributes. It employs signature-based pattern matching algorithms to compare the characteristics of the request against the established database of attack signatures. If the incoming request exhibits patterns or attributes that align with known attack signatures, it is swiftly intercepted and blocked by the SBDAR module, preventing it from reaching the web application backend.

One of the key advantages of SBDAR is its ability to provide immediate protection against known attack patterns, even in situations where anomaly detection mechanisms may have initially failed to flag the request as anomalous. This proactive approach helps mitigate the risk of potential security breaches and ensures a rapid response to emerging threats. Additionally, SBDAR operates independently of the ABD process, allowing it to function as a standalone defense mechanism capable of intercepting and neutralizing threats in real-time.

By integrating SBDAR into web application security architectures, organizations can enhance their ability to detect and respond to malicious HTTP requests effectively. The combination of anomaly-based detection and signature-based response mechanisms provides comprehensive coverage against a wide spectrum of cyber threats, bolstering the overall resilience of web applications against attacks. Ultimately, SBDAR plays a pivotal role in safeguarding web application environments by thwarting malicious activity at the earliest stages of the attack lifecycle, thereby minimizing the potential impact on organizational assets and data.

**Anomaly-Based Detection (ABD)**

Anomaly HTTP requests behave differently than normal HTTP data. In order to carry out detection of HTTP requests which do not conform to the normal HTTP request structure, three features have been selected. According to the results of the experiment conducted with WAF 2015, CSIC 2010 and ECML-PKDD 2007 data sets, Letter Frequency Analysis, Request Length Analysis and Alphanumerical Character Analysis features are selected.

**Alphanumerical Character Analysis:**

The term alphanumeric is used to define a character sequence of letters and numerals (A- Z, a-z, 0-9) in the Latin alphabet. Similarly, each member of this sequence is defined as Alphanumeric. It is a cluster of definitions generated to allow for optimum memory use during data storage of computers. Generally, in one byte, the ASCII correspondence of the alphanumerical value is kept. HTTP Requests coming to the web application have a specific character sequence. This specific character sequence can be analyzed by using the Eq. 1.

According to the HTTP request structure, we can compute the specific character sequence which is found in defined global cluster (e) and generated by using equation (1). The equation (1) which is used for alphanumeric character analysis is presented. In equation (1), “|” means conditional probability, which suggests that the total alphanumerical character value is determined, based on the specific character sequence which is also a condition in the global cluster (e).

**a = Xn i ri ∈ e |(a + 1),**  (1)

where a = Total of Alphanumerical Character Values, r = HTTP Request, n = Total Number of Characters Forming Request, e = Global Cluster. Request Length Analysis: Requests coming to web applications have a specific request structure depending on the developing structure of the web application. One of the request structure feature is the request length. Request length values of memory overflow and cross-site scripting attacks are different to normal requests. Following [9], average length and variance values of the requests are used.

For Request Length Analysis,

Eq. 2 is used. The Eq. 2 is given for computing the probability of an HTTP request whether it is an anomaly or not. However, we could not decide whether a HTTP request is an anomaly or not by only using the Eq. 2. Eq. 2 produces one of the parameters that is used to digitize a HTTP request for ANN input data.

**P = σ 2 (l − µ) 2**  (2)

where P = Probability of Length, µ = Average (Average Value of Requests), σ 2 = Variance (Average Variance Value of Requests), l = Length

**Artificial Neural Networks(ANN)**

To implement learning-based ABD, the best ANN learning model is produced by using HTTP datasets. Firstly, ANN input and output data were normalized by using min-max method linearly between 0-1. The purpose of ANN training is to reach minimum error rate, by using the lowest number of hidden layers and neurons. In the training of ANN weights, a feedforward backpropagation algorithm is selected.

The most important process at this stage is the process of determining the number of hidden layers, the number of neurons in the hidden layers, and the activation function. Hyperbolic tangent is used as the activation function. The hidden layer and neuron numbers are determined experimentally to obtain the best ANN result.

The number of layers can be increased according to the difficulty of the problem, but if the number of layers is much more than required, it causes an increase of process time and memorization of the network. To generate the best ANN model, Mean Squared Error (MSE) value is selected as the lowest possible (close to 0) and the value of regression is selected as the highest (close to 1). In order to train the network, 80 % of all datasets were determined as training data, and 20 % of the dataset as test data. The network has one hidden layer and ten neurons in that hidden layer.

**3.2 UNIQUE FEATURES OF THE SYSTEM**

**Signature-Based Detection (SBD):**

The system incorporates two pivotal components, namely Signature-Based Detection of Normal Requests (SBDNR) and Signature-Based Detection of Anomaly Requests (SBDAR), to bolster its security infrastructure. Through these processes, incoming HTTP requests are meticulously scrutinized and categorized based on their characteristics, thereby facilitating optimized performance and enhanced threat mitigation capabilities. SBDNR and SBDAR operate in tandem to discern between normal and anomalous requests, leveraging signature-based detection techniques to identify patterns indicative of potential security threats. Normal requests, deemed benign and conforming to established usage patterns, are cataloged within the normal signature list for reference and subsequent processing. Conversely, anomalous requests, displaying irregular or suspicious attributes, are promptly flagged and included in the anomaly signature list, warranting further investigation and response measures.

A primary focus of the system lies in combating known intrusion types, notably SQL Injection, Cross-Site Scripting (XSS), and Command Injection, through the implementation of Signature-Based Detection of Known Intrusion Types (SBDKIT). This approach prioritizes the detection and prevention of well-documented attack vectors that pose significant risks to web application security. By employing signature-based detection mechanisms tailored to these specific intrusion types, the system can swiftly identify and neutralize attempted exploits, mitigating the associated security risks and safeguarding the integrity of web applications. SBDKIT functions by matching incoming HTTP requests against a comprehensive database of known attack signatures or patterns associated with SQL Injection, XSS, and Command Injection, enabling proactive identification and blocking of malicious activity. Through continuous refinement and updating of the signature database, the system remains adept at detecting emerging threats and adapting to evolving attack techniques, thereby ensuring robust protection against known intrusion types.

In essence, the integration of SBDNR, SBDAR, and SBDKIT within the system's architecture exemplifies a proactive and multifaceted approach to web application security. By leveraging signature-based detection techniques tailored to both normal and anomalous request patterns, the system can effectively differentiate between benign and potentially malicious activity, enabling prompt response and mitigation measures. Furthermore, the emphasis on detecting and preventing known intrusion types underscores the system's commitment to thwarting well-established attack vectors that pose inherent risks to web applications. Through continuous monitoring, analysis, and refinement of signature databases, the system remains vigilant against evolving threats, thereby fortifying the security posture of web applications and preserving the confidentiality, integrity, and availability of sensitive data and resources.

**Anomaly-Based Detection (ABD):**

The system employs Anomaly-Based Detection (ABD) as a cornerstone of its security framework to identify HTTP requests exhibiting aberrant behavior. ABD operates by scrutinizing various features inherent in HTTP requests, such as letter frequency, request length, and alphanumerical character analysis, to discern anomalies indicative of potential security threats. Among these features, alphanumerical character analysis plays a pivotal role in detecting deviations from expected patterns within HTTP requests. This analysis entails examining the specific character sequences present in HTTP requests to identify anomalies that may signify malicious intent or abnormal behavior.

**Alphanumerical Character Analysis:**

Alphanumerical character analysis leverages conditional probability to assess the total alphanumerical character value based on a predefined global cluster. By calculating the conditional probability of encountering specific character sequences within HTTP requests, the system can discern whether the observed sequences align with established norms or deviate significantly from expected patterns. Anomalies in alphanumerical character sequences may manifest as irregularities in the distribution or frequency of characters, unexpected combinations of alphanumeric characters, or patterns indicative of known attack vectors such as SQL injection or cross-site scripting (XSS).

Through rigorous alphanumerical character analysis, the system can effectively identify suspicious HTTP requests that exhibit aberrant character sequences, enabling proactive detection and mitigation of potential security threats. By leveraging conditional probability and global cluster definitions, the system can discern anomalies in alphanumerical character sequences with a high degree of accuracy, facilitating timely response and remediation measures. Moreover, the integration of alphanumerical character analysis into the broader ABD framework enhances the system's ability to detect and mitigate emerging threats that may evade traditional signature-based detection mechanisms.

In essence, alphanumerical character analysis serves as a crucial component of the system's ABD capabilities, enabling the detection of anomalies within HTTP requests that may evade detection through traditional means. By leveraging conditional probability and global cluster definitions, the system can accurately identify deviations from expected patterns in alphanumerical character sequences, thereby enhancing its ability to detect and mitigate potential security threats. Through continuous refinement and optimization of alphanumerical character analysis techniques, the system remains adept at identifying and mitigating emerging threats, thereby bolstering the overall security posture of web applications and safeguarding against a wide range of cyber threats.

**CHAPTER 4**

**REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION**

**4.1 FEASIBILITY STUDY (TECHNICAL, ECONOMICAL, OPERATIONAL)**

**4.1.1 TECHNICAL REQUIREMENTS**

It mentions the minimum software requirements that will be used to test the Application. In preparation for testing the application, it's imperative to establish the minimum software requirements necessary for smooth and efficient execution. The software stack comprises Python 3.10.7 and Visual Studio Code version 1.7 for the development and deployment of the proxy server and Signature-Based Detection module. Leveraging the capabilities of Python 3.10.7 ensures compatibility and access to the latest features, while Visual Studio Code version 1.7 offers a robust integrated development environment (IDE) for streamlined coding and debugging processes. For Anomaly-Based Detection, Jupiter Notebook version 3.10 is employed, providing an interactive computational environment ideal for data analysis and machine learning tasks. Additionally, a preference is set for Windows 8 and above as the operating system, ensuring compatibility and optimal performance across the testing environment.

The flow of HTTP requests within the application follows a structured path: originating from the client, they are first directed to the proxy server for routing and processing. Subsequently, the requests pass through various stages, including interaction with the test database, signature-based detection, and anomaly-based detection, before finally being accepted for further processing. This meticulous flow ensures comprehensive evaluation and validation of the application's functionality and security measures, culminating in a robust and resilient system ready to withstand the rigors of real-world deployment. Through adherence to these software requirements and meticulous testing procedures, the application can be confidently validated and deployed, offering enhanced security and reliability for web-based operations.

Client --> Proxy Server --> Database (Test) --> Signature (Test) --> Anomaly (Test) --> Accept

**4.1.2 OPERATIONAL REQUIREMENTS**

Regulatory Compliance: Regulatory compliance is a cornerstone of the Web Application Firewall's (WAF) operational framework, ensuring adherence to Indian regulatory requirements such as the Information Technology Act and the Reserve Bank of India's (RBI) cybersecurity guidelines. The Information Technology Act of 2000 establishes legal frameworks for electronic governance and cybersecurity in India, outlining provisions for the protection of electronic records and penalties for cybercrimes. Compliance with this act ensures that the WAF aligns with India's legal requirements for data privacy, security, and electronic transactions. Additionally, adherence to the RBI's cybersecurity guidelines is paramount for financial institutions and entities handling sensitive financial data. These guidelines provide comprehensive directives for cybersecurity risk management, incident reporting, and resilience measures to safeguard against cyber threats. By complying with these regulatory frameworks, the WAF ensures the protection of sensitive data, mitigates legal risks, and fosters trust among stakeholders, reinforcing its role as a robust cybersecurity solution tailored to Indian regulatory standards.

Risk Management: Risk management is another critical aspect of the WAF's operational strategy, aimed at identifying and mitigating potential operational risks that could compromise the security and reliability of web applications. Operational risks encompass a wide range of factors, including downtime, misconfigurations, software vulnerabilities, and human errors, which could lead to service disruptions, data breaches, or financial losses. To mitigate these risks, the WAF adopts a proactive approach to monitoring and risk mitigation, leveraging industry best practices and risk management frameworks such as the ISO 31000 standard. Proactive monitoring involves continuous surveillance of web traffic, system logs, and security events to detect anomalies, intrusions, or unauthorized access attempts in real-time. Additionally, regular vulnerability assessments and penetration testing are conducted to identify and address software vulnerabilities and configuration weaknesses that could be exploited by malicious actors. By implementing robust risk management practices, the WAF enhances the resilience of web applications against potential threats, minimizes the likelihood of security incidents, and safeguards the reputation and trust of the organization and its stakeholders.

**4.2 SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT WHICH MUST INCLUDE THE FOLLOWING:**

**4.2.1 DATA REQUIREMENT**

A Web Application Firewall (WAF) relies on a diverse array of data inputs to effectively monitor, analyze, and secure web traffic, encompassing various facets of HTTP request and response information. This includes comprehensive data such as headers, URLs, parameters, cookies, and payloads, providing granular insights into the nature and characteristics of web traffic. Moreover, user-specific authentication details and session identifiers are vital components, enabling the WAF to differentiate between legitimate users and potential attackers, thereby enhancing security measures. In addition to generic HTTP data, application-specific information such as logic workflows and business processes are essential for contextual understanding and accurate threat assessment. By incorporating knowledge of application logic and behavior, the WAF can identify abnormal patterns or deviations that may indicate malicious activity. Furthermore, security-related intelligence, including known vulnerabilities, attack signatures, and security policies, serves as a foundational pillar for threat detection and prevention. By leveraging this intelligence, the WAF can proactively identify and block known attack vectors, reducing the likelihood of successful exploitation and enhancing overall security posture. Together, these diverse data inputs enable the WAF to create a comprehensive and adaptive defense mechanism, capable of effectively mitigating a wide range of threats and ensuring the integrity and confidentiality of web applications and their users' data.

**4.2.2 FUNCTIONAL REQUIREMENT**

To fulfill its tasks effectively, a Web Application Firewall (WAF) requires several key components and capabilities. Firstly, it must accurately identify web applications using deep packet inspection techniques, agnostic to port numbers or encryption methods. Once identified, the WAF should employ allowlisting mechanisms to permit legitimate traffic while blocking unauthorized or malicious requests based on criteria such as IP addresses or user agents. Thorough threat scanning is essential, involving signature-based detection and heuristic analysis to detect exploits, malware, and spyware. Exploit detection capabilities should be continuously updated to address emerging vulnerabilities. Additionally, the WAF should include features for data loss prevention, employing pattern matching and encryption to safeguard sensitive information. Regulatory compliance enforcement is also crucial, ensuring adherence to standards such as PCI DSS and GDPR through the implementation of security policies and audit log generation. By incorporating these functional requirements, the WAF can effectively secure web applications while ensuring compliance with regulatory standards and protecting sensitive data from threats and exploitation.

**4.2.3 PERFORMANCE REQUIREMENT**

Performance requirements for Web Application Firewall (WAF) implementation are crucial to ensure seamless integration without degrading the performance or user experience of protected web applications. Firstly, minimal latency is imperative to maintain a seamless browsing experience, particularly for real-time interactions such as online gaming or financial transactions. Scalability is essential, with the WAF needing to scale horizontally and vertically to handle increasing traffic levels efficiently. High throughput capacity is necessary to manage peak traffic volumes without bottlenecks or degradation. Efficient resource utilization minimizes CPU, memory, and network bandwidth consumption, ensuring the WAF can handle traffic spikes effectively. Content caching, compression, and SSL/TLS offloading techniques further optimize performance by reducing the workload on backend servers and conserving network resources. Load balancing capabilities distribute traffic across multiple servers efficiently, enhancing overall performance and availability. Comprehensive monitoring and optimization features enable administrators to identify and address performance bottlenecks proactively. By meeting these requirements, organizations can seamlessly integrate WAF solutions, enhancing security without sacrificing performance or user experience.

**4.2.4 MAINTAINABILITY REQUIREMENT**

Maintainability requirements for a Web Application Firewall (WAF) are essential to ensure the long-term effectiveness, manageability, and adaptability of the solution. Firstly, a modular architecture with extensible APIs and plugin frameworks enables seamless integration of new features and updates to accommodate evolving security needs. A centralized management interface provides administrators with comprehensive visibility and control over security policies, traffic logs, and system health. Automated updates and patch management mechanisms ensure that the WAF remains current with the latest security patches and threat intelligence updates, reducing the burden on administrators and mitigating security risks. Efficient configuration management, logging, and auditing capabilities facilitate consistent policy enforcement, troubleshooting, and compliance reporting. Clear documentation and training resources empower administrators to effectively utilize the WAF's features and capabilities. Health monitoring and alerting mechanisms proactively detect and notify administrators of potential issues or system failures, while vendor support and maintenance contracts ensure timely assistance with troubleshooting and bug resolution. Interoperability and integration capabilities enable seamless collaboration with existing security infrastructure and enhance overall security effectiveness and visibility. By addressing these maintainability requirements, organizations can ensure that their WAF remains resilient, adaptable, and well-supported throughout its lifecycle, maximizing the effectiveness of their web application security defenses.

**4.2.5 SECURITY REQUIREMENT**

Security requirements for a Web Application Firewall (WAF) are paramount to safeguard web applications against an array of threats and vulnerabilities. Firstly, robust threat detection and prevention mechanisms are essential, employing techniques such as signature-based detection, behavioral analysis, and machine learning to identify and mitigate attacks like SQL injection, cross-site scripting (XSS), and directory traversal. Granular access control mechanisms enforce fine-grained security policies based on factors like IP address and user agent, while SSL/TLS inspection capabilities decrypt and analyze encrypted HTTPS traffic for hidden threats. Input validation, output encoding, and session management practices mitigate injection attacks, XSS vulnerabilities, and session-related threats. Additionally, content filtering and data loss prevention features prevent the unauthorized transmission of sensitive information, while integration with security intelligence sources enhances the WAF's ability to proactively defend against known threats. Comprehensive auditing and logging capabilities ensure the recording of security events and policy violations for forensic analysis and compliance reporting. By fulfilling these security requirements, a WAF can effectively fortify web applications, preserving the confidentiality, integrity, and availability of critical resources and information.

**4.3 SDLC MODEL TO BE USED**

The decision to adopt a Waterfall methodology for a project is typically based on specific project requirements, constraints, and organizational factors. Here are some common reasons for choosing the Waterfall methodology:

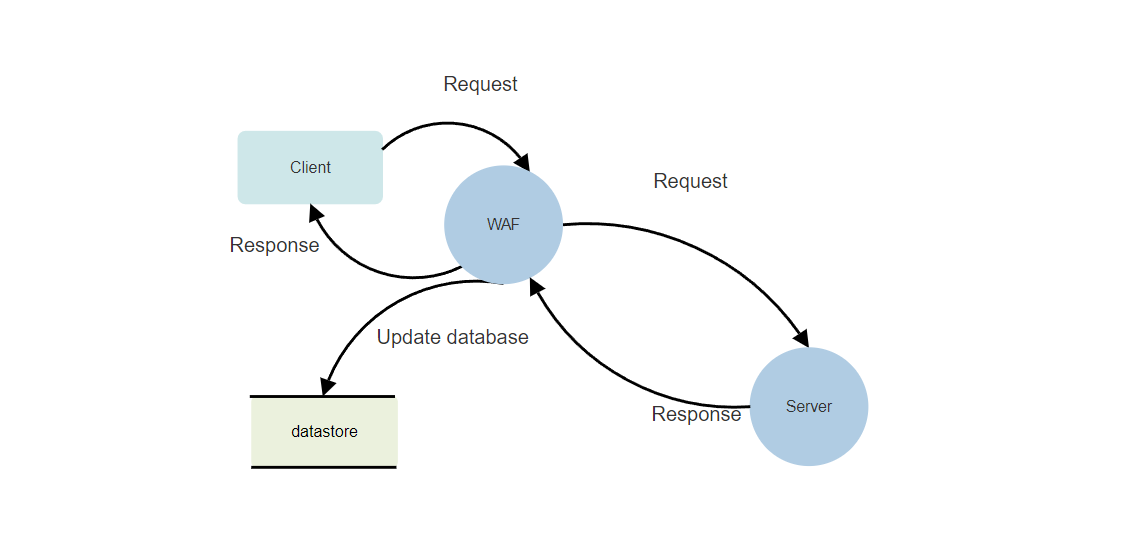
**Well-Defined Requirements**: When the project has clearly defined and stable requirements that are unlikely to change significantly throughout the project's lifecycle. Waterfall is suitable when you can gather and document all the requirements up front.

**Low Uncertainty:** If there is a high level of confidence in the project scope and objectives, and the technology and processes to be used are well-understood, Waterfall can be a good choice. It is less adaptable to uncertainty and change.

**Large-Scale and Complex Projects:** Waterfall can be beneficial for large-scale, complex projects where a comprehensive and detailed project plan is essential for successful execution.

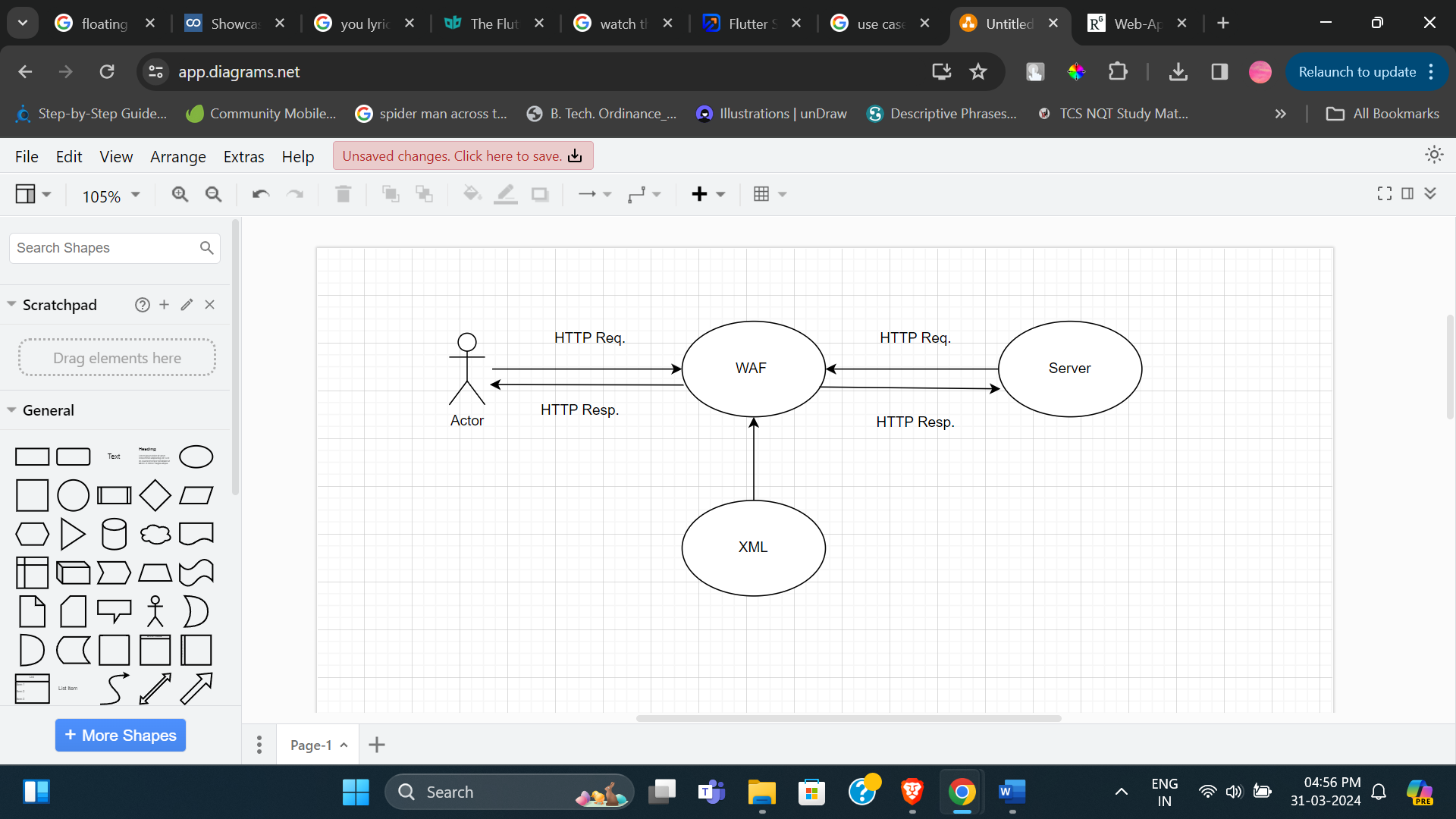
**4.4 SYSTEM DESIGN**

**4.4.1 DATA FLOW DIAGRAMS**

****

**Fig. 1**

**4.4.2 USE CASE DIAGRAM**

****

**Fig . 2**

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 INTRODUCTION TO TOOLS AND TECHNOLOGIES USED FOR IMPLEMENT TECHNOLOGY USED**

**Anaconda Distribution :** Anaconda is a comprehensive distribution of Python and R programming languages primarily designed for scientific computing and data science tasks. It provides a convenient package management system through Conda, which simplifies the installation and management of various libraries and dependencies. Anaconda comes pre-packaged with a plethora of commonly used data science tools and libraries, making it an attractive option for researchers, data scientists, and developers. With Anaconda, users can easily set up environments tailored to their specific projects, ensuring reproducibility and compatibility across different computing environments. Its ease of use and robustness have made it a popular choice for individuals and organizations working in fields such as machine learning, artificial intelligence, and computational biology.

**TensorFlow, Keras :**TensorFlow is an open-source machine learning framework developed by Google that provides a flexible and scalable platform for building and training deep learning models. It offers extensive support for neural networks, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more. TensorFlow's computational graph abstraction allows for efficient execution on both CPUs and GPUs, making it suitable for a wide range of applications. Keras, on the other hand, is a high-level neural network library written in Python that serves as an interface for building and training neural networks. It provides a user-friendly API that abstracts away the complexities of TensorFlow, enabling rapid prototyping and experimentation. Together, TensorFlow and Keras form a powerful ecosystem for developing and deploying machine learning models for tasks such as image recognition, natural language processing, and reinforcement learning.

**Oracle:**Oracle Corporation is a leading provider of database management systems (DBMS) and enterprise software solutions. Its flagship product, Oracle Database, is one of the most widely used relational database management systems in the world. Known for its reliability, scalability, and performance, Oracle Database powers mission-critical applications across industries such as finance, telecommunications, and healthcare. It supports various data models, including relational, object-relational, and XML, and offers a wide range of features such as data security, backup and recovery, and advanced analytics. Oracle's commitment to innovation has led to the development of cloud-based database services, enabling organizations to leverage the power of Oracle Database in a scalable and cost-effective manner.

**Database Management System (DBMS):** A Database Management System (DBMS) is a software application that facilitates the creation, management, and manipulation of databases. It provides users with an interface to interact with the database, perform queries, and retrieve information efficiently. DBMSs can be categorized into different types based on their data model (e.g., relational, NoSQL), architecture (e.g., centralized, distributed), and functionality (e.g., SQL-based, object-oriented). Relational database management systems (RDBMS) such as Oracle, MySQL, and PostgreSQL are the most commonly used type of DBMS, employing a tabular data model based on relations or tables. NoSQL databases, on the other hand, eschew the rigid structure of relational databases in favor of a more flexible data model, making them well-suited for applications with rapidly changing data requirements. Regardless of the type, a DBMS plays a critical role in modern computing, serving as the backbone of data-driven applications and systems.

**Python:** Python is a high-level, interpreted programming language known for its simplicity and readability. It has a large and active community that contributes to its extensive standard library and a wide range of third-party packages for various purposes, including web development, data analysis, machine learning, and more. Python's syntax emphasizes readability and simplicity, making it an ideal language for beginners and experienced programmers alike. Its versatility and ease of use have led to its widespread adoption in various domains, including scientific computing, web development, and automation. Python's popularity continues to grow, driven by its simplicity, flexibility, and the vast ecosystem of libraries and frameworks available to developers.

**CHAPTER 6**

**TESTING AND MAINTAINENCE**

**6.1 TESTING TECHNIQUES AND TEST CASES USED**

The decision to adopt a Waterfall methodology for a project is a strategic choice influenced by a variety of factors, ranging from project requirements and constraints to organizational culture and industry standards. One of the primary considerations in opting for Waterfall is the presence of well-defined and stable requirements. When a project's objectives, functionalities, and deliverables are clearly outlined from the outset, Waterfall's linear and sequential approach can be advantageous. This methodology is particularly suitable for scenarios where stakeholders have a clear vision of the final product and minimal expectation for alterations or additions throughout the project's lifecycle. By gathering and documenting all requirements upfront, Waterfall fosters a structured and systematic development process, reducing the likelihood of scope creep and ensuring alignment between project outcomes and stakeholders' expectations.

Moreover, the decision to choose Waterfall may also be influenced by the level of uncertainty associated with the project. If there is a high degree of confidence in the project scope, objectives, and the technologies and processes to be employed, Waterfall can provide a solid framework for project management. Unlike Agile methodologies, which thrive in environments characterized by ambiguity and frequent changes, Waterfall is less adaptable to uncertainty. Instead, it relies on a predefined sequence of phases, including requirements analysis, design, implementation, testing, deployment, and maintenance, each building upon the completion of the previous stage. This rigidity can be beneficial in contexts where predictability and control are paramount, such as in industries with stringent regulatory requirements or mission-critical projects where deviations from the original plan could have significant consequences.

In addition to stability and predictability, regulatory compliance often plays a crucial role in the selection of the Waterfall methodology. Certain industries, such as healthcare, finance, and government, are subject to strict regulatory frameworks and compliance standards. These regulations often mandate rigorous documentation, traceability, and validation processes throughout the project lifecycle. Waterfall's emphasis on comprehensive documentation and formalized reviews aligns well with these requirements, providing auditable evidence of adherence to regulatory mandates. By following a structured and well-documented approach, organizations can mitigate compliance risks and ensure that their projects meet the necessary regulatory standards, reducing the likelihood of legal and financial penalties.

Furthermore, the suitability of Waterfall extends to large-scale and complex projects where meticulous planning and coordination are essential for success. Such projects often involve multiple stakeholders, interdependent tasks, and intricate dependencies that require careful management and oversight. Waterfall's phased approach facilitates the development of a detailed project plan, outlining the sequence of activities, resource allocation, and timelines for each stage of the project. This comprehensive planning not only enables better risk management and resource allocation but also enhances communication and collaboration among project team members and stakeholders. By breaking down the project into manageable phases and milestones, Waterfall provides a roadmap for navigating the complexities of large-scale endeavors, ensuring alignment with organizational goals and objectives.

In conclusion, the decision to adopt a Waterfall methodology for a project is driven by a combination of factors, including the clarity and stability of project requirements, the level of uncertainty, regulatory compliance requirements, and the scale and complexity of the project. By providing a structured and systematic approach to project management, Waterfall offers a viable solution for organizations seeking predictability, control, and compliance in their development initiatives. However, it's essential to evaluate the specific needs and characteristics of each project to determine whether Waterfall is the most suitable methodology or if an alternative approach, such as Agile, may be better suited to the project's unique requirements and constraints.

**Test Levels**

**Unit Testing :** Testing a Web Application Firewall (WAF) is a crucial aspect of ensuring the security and effectiveness of web applications against potential threats and vulnerabilities. This process involves multiple test levels, each designed to comprehensively assess different aspects of the WAF's functionality, security, and performance. Unit testing serves as the foundation of WAF testing, focusing on verifying the functionality and effectiveness of individual components, rules, or policies within the WAF deployment. Rule testing is a key component of unit testing, where tests are created to validate the behavior of individual rules within the WAF's rule set. These tests typically involve verifying that rules such as SQL injection rules correctly identify and block malicious attempts while minimizing false positives and false negatives. Similarly, policy testing is conducted to verify the enforcement of security policies defined within the WAF, ensuring that configured policies such as access control rules, input validation rules, or content filtering rules are correctly applied and enforced.

Furthermore, protocol and parameter handling testing are essential to assess the WAF's ability to handle various HTTP protocols, headers, parameters, and payloads. This testing phase ensures compliance with HTTP standards, proper parsing and validation of HTTP requests and responses, and handling of edge cases or malformed input. By generating test cases to cover common HTTP methods, URL structures, query strings, request headers, and request/response payloads, organizations can verify that the WAF behaves predictably and securely in different scenarios. Integration testing is another critical test level that focuses on verifying the interoperability of the WAF with other components of the web application environment. This includes testing the WAF's ability to communicate with upstream and downstream systems, exchange data and configuration settings, and integrate with existing security infrastructure such as SIEM platforms or threat intelligence feeds.

Moreover, performance testing is integral to evaluating the impact of the WAF on web application performance and scalability. This test level includes assessing factors such as throughput, response times, and resource utilization under different load conditions and traffic patterns. By conducting performance tests as part of the testing regimen, organizations can identify any performance bottlenecks or resource constraints that may affect the WAF's effectiveness in production environments. Additionally, logging and monitoring testing ensure the proper generation and handling of security logs, alerts, and audit trails by the WAF. This involves verifying that security events are accurately logged, categorized, and forwarded to designated log management systems or SIEM platforms for analysis and reporting purposes. It also ensures that monitoring and alerting mechanisms are functioning correctly, triggering alerts for security incidents, policy violations, or operational issues according to predefined thresholds and conditions.

Incorporating these test levels into the development and deployment process of a WAF is essential for validating its functionality, security, and performance. By conducting rigorous testing, organizations can identify and mitigate potential vulnerabilities and weaknesses in the WAF deployment, thereby enhancing the overall security posture of their web applications. Additionally, comprehensive testing helps build confidence in the effectiveness of the WAF, ensuring that it can effectively protect web applications against various threats and attacks. Ultimately, investing in thorough testing practices enables organizations to proactively safeguard their web applications and data from potential security breaches and cyber threats, thereby safeguarding their reputation and preserving customer trust.

**Integration Testing:** Integration testing for a Web Application Firewall (WAF) is critical for validating its seamless integration with the broader web application environment, encompassing web servers, application frameworks, network infrastructure, and security systems. This multifaceted testing approach involves several key steps to ensure the WAF's compatibility, effectiveness, and reliability within the overall infrastructure. Deployment configuration testing is the initial phase, focusing on verifying the correct deployment and configuration of the WAF within the network architecture. This includes ensuring proper network routing, DNS resolution, and load balancing configurations to route traffic through the WAF for inspection and protection. Traffic routing and load balancing testing then assess the routing of web traffic through the WAF, validating that all incoming and outgoing HTTP requests and responses are intercepted and inspected as intended. Additionally, this phase verifies that load balancers and reverse proxies are configured to effectively direct traffic to the WAF instances, distributing the load evenly across multiple WAF nodes if deployed in a clustered or distributed architecture.

Protocol compatibility testing is another crucial aspect, ensuring that the WAF can seamlessly handle various web protocols, including HTTP, HTTPS, WebSocket, and SPDY. This involves validating the WAF's ability to decrypt, inspect, and re-encrypt HTTPS traffic without issues, as well as its compatibility with encrypted traffic (SSL/TLS).

Furthermore, web application framework support testing evaluates the WAF's compatibility with different frameworks such as Ruby on Rails, Django, Laravel, or .NET, ensuring it can effectively protect applications built on various platforms without introducing compatibility issues. API and web service testing validate the WAF's compatibility with APIs and web services used by web applications, ensuring it can inspect and protect RESTful APIs, SOAP services, GraphQL endpoints, and other web service protocols, while handling common data formats like JSON and XML.

Error handling and failover testing assess the WAF's ability to handle and recover from failure scenarios such as node failures, network interruptions, or backend server outages. This involves simulating failure scenarios and verifying that the WAF can gracefully handle and recover from incidents, including automatic failover to standby instances or routing traffic to alternative servers. Additionally, logging and monitoring integration testing validates the integration of the WAF with logging and monitoring systems such as SIEM platforms and log management solutions, ensuring that security events and audit logs are properly forwarded for analysis and reporting.

Finally, threat intelligence feeds and external integrations testing assesses the WAF's integration with external threat intelligence feeds, vulnerability databases, and security orchestration platforms, ensuring it can receive threat intelligence updates and dynamically adjust its security policies and rulesets in response to emerging threats.

By conducting thorough integration testing, organizations can ensure that their WAF seamlessly integrates with the web application environment, effectively protects against web-based threats, and maintains the reliability and performance of web applications and services.Verify that the WAF is correctly deployed and configured within the network architecture. Ensure that network routing, DNS resolution, and load balancing configurations are properly configured to route traffic through the WAF for inspection and protection.

Test the routing of web traffic through the WAF to ensure that all incoming and outgoing HTTP requests and responses are properly intercepted and inspected.Verify that load balancers and reverse proxies are configured to direct traffic to the WAF instances effectively, distributing the load evenly across multiple WAF nodes if deployed in a clustered or distributed architecture. Validate the compatibility of the WAF with different web protocols, such as HTTP, HTTPS, WebSocket, and SPDY.

Test the handling of encrypted traffic (SSL/TLS) and ensure that the WAF can decrypt, inspect, and re-encrypt HTTPS traffic without issues.Test the compatibility of the WAF with various web application frameworks, such as Ruby on Rails, Django, Laravel, or .NET. Ensure that the WAF can effectively protect applications built on different frameworks without introducing compatibility issues or breaking functionality.Verify the WAF's compatibility with APIs and web services used by the web applications.

Test the WAF's ability to inspect and protect RESTful APIs, SOAP services, GraphQL endpoints, and other web service protocols. Ensure that the WAF can handle JSON, XML, and other data formats commonly used in web service communication, validating the parsing, validation, and filtering of API requests and responses. Conduct tests to evaluate the WAF's error handling mechanisms and failover capabilities. Simulate failure scenarios, such as WAF node failures, network interruptions, or backend server outages, and verify that the WAF can gracefully handle and recover from such incidents. Test failover mechanisms, such as automatic failover to standby WAF instances or routing traffic to alternative backend servers, to ensure uninterrupted service availability in case of failures. Validate the integration of the WAF with logging and monitoring systems, such as SIEM platforms, log management solutions, and security information dashboards.

Verify that security events, policy violations, and audit logs generated by the WAF are properly forwarded to designated logging systems for analysis, correlation, and reporting. Test the integration of the WAF with external threat intelligence feeds, vulnerability databases, and security orchestration platforms.

Ensure that the WAF can receive threat intelligence updates, such as IP reputation lists, malware signatures, and exploit patterns, and dynamically adjust its security policies and rulesets to respond to emerging threats. By conducting thorough integration testing, organizations can ensure that their Web Application Firewall seamlessly integrates with the web application environment, effectively protects against web-based threats, and maintains the reliability and performance of web applications and services.

**System Testing**: System testing for a Web Application Firewall (WAF) entails validating its functionality, performance, security, and reliability as an integrated system within the context of the web application environment. This comprehensive testing approach aims to ensure that the WAF effectively detects and mitigates various types of web-based attacks, enforces security policies, and integrates seamlessly with other components of the web application infrastructure.

Functional testing forms a foundational part of system testing, involving the verification of the WAF's ability to accurately identify and counter web-based threats such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and directory traversal. Additionally, functional testing assesses the enforcement of security policies, access controls, input validation rules, and content filtering mechanisms to ensure they function as intended, providing effective protection for web applications.

Performance testing is another critical aspect of system testing, focusing on evaluating the WAF's throughput, latency, and resource utilization under varying traffic conditions and load levels. This includes stress testing, load testing, and scalability testing to identify performance bottlenecks, resource constraints, or scalability limitations that may impact the WAF's effectiveness in real-world scenarios. Security testing is also paramount, involving the validation of the WAF's security posture through assessments, vulnerability scans, and penetration tests.

This phase aims to identify and exploit potential security vulnerabilities or misconfigurations in the WAF deployment while testing its resistance to evasion techniques such as HTTP request smuggling or protocol-level attacks.

Furthermore, protocol and content handling testing evaluates the WAF's ability to handle various HTTP protocols, headers, parameters, and payloads. This involves verifying its capacity to parse, validate, and sanitize input data, including URL parameters, form data, and HTTP headers. Additionally, integration testing ensures the seamless interoperability of the WAF with other components of the web application environment, including web servers, application frameworks, load balancers, and logging systems. It verifies that traffic routing, load balancing, and failover mechanisms function correctly and that the WAF integrates seamlessly without introducing compatibility issues or disruptions.

Logging and reporting testing validates the generation, retention, and retrieval of security logs, audit trails, and reporting data produced by the WAF. This ensures that security events, policy violations, and operational metrics are accurately logged and available for analysis and reporting purposes, supporting incident response, forensic analysis, and compliance auditing requirements. Overall, system testing for a WAF encompasses a comprehensive evaluation of its functionality, performance, security, and integration capabilities within the broader web application environment, aiming to ensure robust protection against web-based threats and vulnerabilities.

System testing for a Web Application Firewall (WAF) involves verifying its functionality, performance, security, and reliability as a whole system within the context of the web application environment. Here's how system testing can be conducted for a WAF .Verify that the WAF accurately detects and mitigates various types of web-based attacks, including SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and directory traversal.

Test the enforcement of security policies, access controls, input validation rules, and content filtering mechanisms to ensure that they function as intended and effectively protect web applications. Measure the throughput, latency, and resource utilization of the WAF under different traffic conditions and load levels. Assess its ability to handle peak traffic volumes without degrading performance or introducing latency.

Conduct stress testing, load testing, and scalability testing to identify performance bottlenecks, resource constraints, or scalability limitations that may affect the WAF's effectiveness in production environments. Validate the security posture of the WAF by conducting security assessments, vulnerability scans, and penetration tests. Identify and exploit potential security vulnerabilities or misconfigurations in the WAF deployment.

Test the WAF's resistance to evasion techniques, such as HTTP request smuggling, request fragmentation, or protocol-level attacks, to ensure that it can effectively detect and block sophisticated attack vectors. Test the WAF's handling of various HTTP protocols, headers, parameters, and payloads. Verify its ability to parse, validate, and sanitize input data, including URL parameters, form data, and HTTP headers. Evaluate the WAF's handling of different content types, encoding methods, and data formats, such as JSON, XML, binary data, and multi-part form submissions, to ensure comprehensive coverage of web application traffic.

Validate the integration of the WAF with other components of the web application environment, including web servers, application frameworks, load balancers, and logging systems. Verify that traffic routing, load balancing, and failover mechanisms are functioning correctly, and that the WAF seamlessly interoperates with upstream and downstream systems without introducing compatibility issues or disruptions.

Test the generation, retention, and retrieval of security logs, audit trails, and reporting data produced by the WAF. Ensure that security events, policy violations, and operational metrics are properly logged and available for analysis and reporting. Validate the accuracy, completeness, and timeliness of logging and reporting features to support incident response, forensic analysis, and compliance auditing requirements. Verify the high availability and redundancy of the WAF deployment by testing failover mechanisms, load balancing configurations, and disaster recovery procedures. Conduct failover testing and failback testing to ensure seamless transition between active and standby WAF instances in case of failures or maintenance events. Ensure that the WAF deployment meets regulatory compliance requirements relevant to the organization's industry and geographic location. Validate compliance with standards such as PCI DSS, HIPAA, GDPR, and others. Test the WAF's ability to enforce data protection, access control, and privacy requirements mandated by regulatory frameworks through policy enforcement, data encryption, and access logging mechanisms. By conducting comprehensive system testing, organizations can ensure that their Web Application Firewall effectively protects web applications against various threats, maintains high performance and reliability, and complies with regulatory standards and industry best practices.

**Test Completeness**

Test completeness is a critical aspect of the software testing process, ensuring that all relevant test scenarios have been executed, and the software meets the desired quality standards before release. Several criteria can be employed to assess test completeness, with two primary considerations being 100% test coverage and the resolution of open bugs.

Firstly, achieving 100% test coverage is a fundamental criterion for assessing test completeness. Test coverage refers to the extent to which the source code or functional requirements of the software have been exercised by test cases. It encompasses various dimensions, including statement coverage, branch coverage, and path coverage, each aiming to ensure that all parts of the software are thoroughly tested. Statement coverage involves verifying that each line of code in the software has been executed at least once during testing. Branch coverage extends this concept by ensuring that all possible branches or decision points within the code have been traversed. Path coverage takes this a step further by examining all possible execution paths through the code, including loops and conditional statements. By striving for 100% test coverage across these dimensions, testers can have confidence that the software has been thoroughly exercised and potential defects have been identified.

Secondly, ensuring that all open bugs are fixed or scheduled for resolution in the next release is another important criterion for assessing test completeness. During the testing process, defects or issues may be identified in the software, ranging from minor glitches to critical vulnerabilities. These bugs are typically logged in a defect tracking system and assigned a priority based on their severity and impact on the software's functionality. As part of test completeness, it is essential to review the status of open bugs and ensure that they are either fixed before release or prioritized for resolution in subsequent releases. This involves collaborating with the development team to prioritize and address identified issues, conducting regression testing to verify bug fixes, and updating the defect tracking system accordingly. By addressing open bugs in a timely manner, testers can help improve the overall quality and reliability of the software.

In summary, test completeness is a multifaceted concept that encompasses various criteria for evaluating the thoroughness and effectiveness of the testing process. Achieving 100% test coverage across different dimensions of the software, including statement, branch, and path coverage, is essential for ensuring that all parts of the code have been thoroughly exercised. Additionally, ensuring that all open bugs are addressed before release or scheduled for resolution in subsequent releases is crucial for maintaining the integrity and reliability of the software. By adhering to these criteria, testers can help mitigate risks, improve software quality, and ensure that the software meets the desired quality standards before deployment.

**Test Cases Used**

Testing a Web Application Firewall (WAF) involves creating various test cases to ensure it effectively detects and mitigates web application threats. Here are some example test cases for a WAF:

Testing for various security vulnerabilities, such as SQL injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Remote File Inclusion (RFI), Command Injection, and Path Traversal, is crucial to ensure the effectiveness of a Web Application Firewall (WAF) in protecting web applications against common attack vectors. These vulnerabilities pose significant risks to the security and integrity of web applications, making it imperative to develop comprehensive test cases to assess the WAF's detection and prevention capabilities. Below are detailed test cases for each vulnerability category:

**SQL Injection Test Cases :**

1. Send a request with a SQL injection payload in a URL parameter to evaluate the WAF's ability to detect and block SQL injection attacks targeting the application's database.

2. Inject SQL code into form fields and verify that the WAF accurately identifies and blocks the malicious request, preventing unauthorized access to sensitive data or database manipulation.

3. Test various SQL injection techniques, including UNION-based and Blind SQL injection, to assess the WAF's detection capabilities under different attack scenarios.

**Cross-Site Scripting (XSS) Test Cases**:

1. Inject XSS payloads into input fields and assess whether the WAF detects and blocks them, preventing attackers from executing malicious scripts in users' browsers.

2. Test for various types of XSS attacks, such as stored, reflected, and DOM-based XSS, to evaluate the WAF's ability to mitigate different XSS attack vectors effectively.

**Cross-Site Request Forgery (CSRF) Test Cases:**

1. Craft a CSRF attack and verify that the WAF prevents the forged request from being executed, protecting against unauthorized actions initiated by malicious third-party sites.

2. Test with different HTTP methods (e.g., POST, GET) to ensure the WAF provides comprehensive protection against CSRF attacks across different scenarios.

**Remote File Inclusion (RFI) Test Cases:**

1. Send requests with RFI payloads to evaluate the WAF's ability to detect and block attempts to include remote files, preventing attackers from executing malicious code on the server.

2. Attempt to include remote files via input fields and assess the WAF's response to ensure that it effectively mitigates RFI attacks.

**Command Injection Test Cases:**

1. Inject OS command payloads in request parameters and assess whether the WAF identifies and blocks the malicious input, preventing attackers from executing arbitrary commands on the server.

**Path Traversal Test Cases:**

1. Send requests with path traversal payloads to test the WAF's ability to detect and prevent directory traversal attacks, ensuring that attackers cannot access unauthorized files or directories on the server.

By executing these comprehensive test cases, organizations can evaluate the effectiveness of their WAF in mitigating common security vulnerabilities and protecting web applications against potential threats. Additionally, regular testing and validation of the WAF's security features help ensure that it remains up-to-date and capable of defending against emerging attack techniques and tactics.

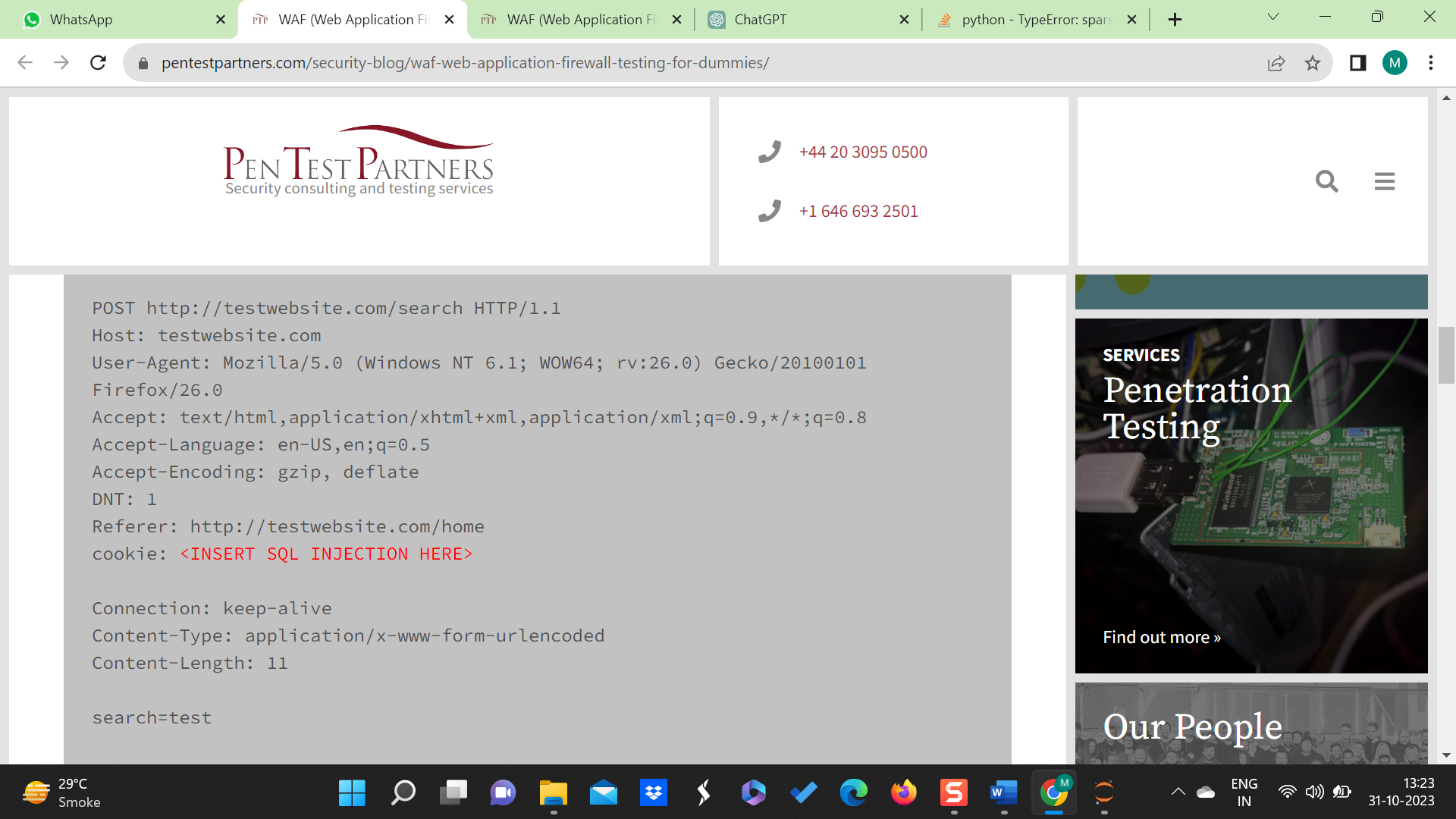


Fig. 3 - Testing for a SQL Query

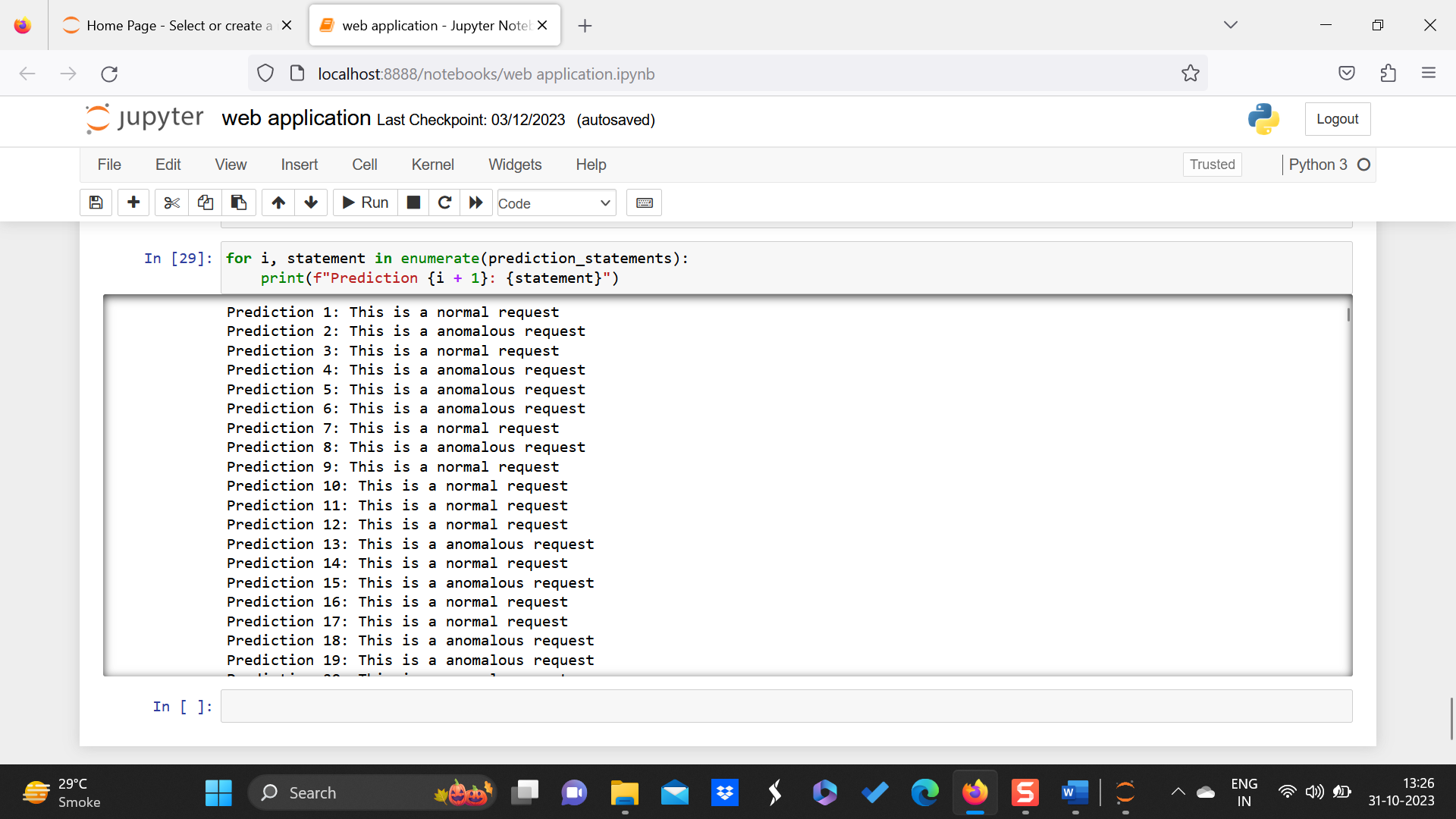
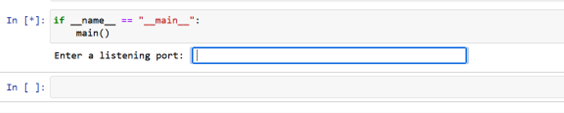


Fig. 4 – Result

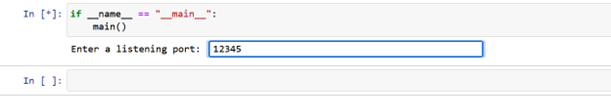
**CHAPTER 7**

**RESULTS AND DISCUSSIONS**

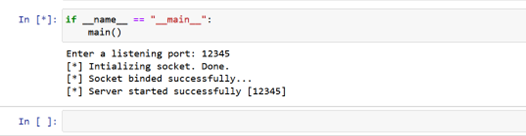
**7.1 PROJECT SNAPSHOTS**

****

**Fig. 5 – Entry point of project**

****

**Fig. 6 – Port number for server request**

****

**Fig. 7 – Server starting on given port**

**7.2 KEY FINDINGS OF THE PROJECT**

**MODULE 1: SIGNATURE BASED DETECTION**

Signature-based detection is a fundamental technique used by Web Application Firewalls (WAFs) to identify and block known web-based attacks. Here's how signature-based detection works within a WAF:

1. Signature Creation: Security researchers and WAF vendors create signatures that represent specific patterns or characteristics of known attacks. These signatures are typically based on the unique syntax, semantics, or behaviors associated with common web-based attacks, such as SQL injection, cross-site scripting (XSS), or command injection.

Signatures may encompass various elements of HTTP requests and responses, including URLs, parameters, headers, cookies, payloads, and response codes.

2. Signature Database: The WAF maintains a database or repository of signatures representing known web-based attacks. This signature database is continuously updated with new signatures to address emerging threats and vulnerabilities.

Signature updates may be provided by the WAF vendor through subscription services, threat intelligence feeds, or manual updates initiated by administrators.

3. Traffic Inspection: As web traffic passes through the WAF, it is inspected in real-time against the signature database. The WAF examines HTTP requests and responses, parsing and analyzing their content to identify patterns that match known attack signatures.

Each incoming request or response is compared against the signatures in the database to determine if it exhibits any malicious characteristics indicative of a known attack.

4. Pattern Matching: Signature-based detection relies on pattern matching algorithms to identify matches between observed traffic patterns and known attack signatures. These algorithms may employ techniques such as regular expressions, string matching, or content parsing to identify malicious patterns within HTTP traffic.

Matching signatures trigger the activation of corresponding security rules or policies within the WAF to block or mitigate the detected attacks.

5. Action Enforcement: When a match is detected between observed traffic and a signature in the database, the WAF takes predefined actions to enforce security policies and mitigate the detected threat.

Depending on the severity and nature of the attack, actions may include blocking the malicious request, redirecting the user to a predefined error page, logging the security event for further analysis, or alerting administrators for manual review and intervention.

6. False Positive Management: Signature-based detection may occasionally produce false positives, where legitimate traffic is incorrectly flagged as malicious due to similarities with known attack patterns.

WAF administrators can fine-tune signature-based detection rules and policies to minimize false positives, adjusting sensitivity levels, whitelisting trusted sources, or refining signature criteria based on observed traffic patterns.

7. Continuous Updates: To effectively defend against evolving threats, the WAF must continuously update its signature database with new signatures, attack patterns, and threat intelligence feeds.

Regular updates ensure that the WAF remains current with the latest attack trends and vulnerabilities, enabling it to detect and block emerging threats in real-time.

By leveraging signature-based detection, Web Application Firewalls can proactively identify and mitigate known web-based attacks, providing a critical layer of defense to protect web applications against a wide range of threats and vulnerabilities.

Signature-based detection is a foundational technique utilized by Web Application Firewalls (WAFs) to identify and thwart known web-based attacks effectively. In today's cyber landscape, where web applications are increasingly targeted by malicious actors seeking to exploit vulnerabilities for various purposes such as data theft, unauthorized access, or disruption of services, signature-based detection plays a crucial role in fortifying the security posture of organizations' digital assets. This comprehensive approach to detecting and mitigating threats involves several key components and processes, each contributing to the overall efficacy of the WAF in safeguarding web applications against a wide range of cybersecurity risks.

The process of signature-based detection begins with the creation of signatures, which are meticulously crafted representations of specific patterns or characteristics associated with known web-based attacks. These signatures are developed by security researchers and WAF vendors who meticulously analyze the syntax, semantics, and behaviors underlying common attack vectors such as SQL injection, cross-site scripting (XSS), or command injection. Signatures may encompass various elements of HTTP requests and responses, including URLs, parameters, headers, cookies, payloads, and response codes, enabling the WAF to scrutinize web traffic with precision and granularity.

Once created, these signatures are compiled into a comprehensive signature database or repository, which serves as the backbone of the WAF's detection capabilities. This database is continuously updated to reflect the evolving threat landscape, with new signatures added to address emerging threats and vulnerabilities. These updates may be delivered to the WAF through subscription services, threat intelligence feeds, or manual updates initiated by administrators, ensuring that the WAF remains current and adept at detecting the latest attack trends and techniques.

As web traffic traverses the WAF, it undergoes real-time inspection against the signature database. The WAF meticulously parses and analyzes HTTP requests and responses, leveraging sophisticated pattern matching algorithms to identify matches between observed traffic patterns and known attack signatures. These algorithms may employ techniques such as regular expressions, string matching, or content parsing to identify malicious patterns indicative of a potential attack. When a match is detected, corresponding security rules or policies within the WAF are triggered, enabling proactive mitigation of the identified threats.

However, the effectiveness of signature-based detection relies not only on its ability to accurately identify malicious traffic but also on its capacity to minimize false positives. False positives occur when legitimate traffic is incorrectly flagged as malicious due to similarities with known attack patterns, potentially leading to disruptions in service or user experience. To mitigate this risk, WAF administrators can fine-tune detection rules and policies, adjusting sensitivity levels, whitelisting trusted sources, or refining signature criteria based on observed traffic patterns. By striking the right balance between detection accuracy and false positive mitigation, organizations can maximize the effectiveness of their WAF while minimizing the risk of inadvertently blocking legitimate traffic.

Furthermore, signature-based detection is not a static process but rather a dynamic and iterative one that requires continuous updates and refinements to remain effective in the face of evolving threats. To this end, the WAF must undergo regular updates to its signature database, incorporating new signatures, attack patterns, and threat intelligence feeds to keep pace with the rapidly changing threat landscape. These updates enable the WAF to detect and block emerging threats in real-time, providing organizations with a proactive defense mechanism to safeguard their web applications against the latest cybersecurity risks.

In summary, signature-based detection is a fundamental technique used by Web Application Firewalls to identify and mitigate known web-based attacks effectively. By leveraging meticulously crafted signatures, sophisticated pattern matching algorithms, and continuous updates to the signature database, organizations can bolster the security of their web applications and protect against a wide range of cybersecurity threats. However, it is essential to strike a balance between detection accuracy and false positive mitigation to ensure the optimal performance of the WAF while minimizing the risk of disrupting legitimate traffic. By embracing signature-based detection as part of a comprehensive cybersecurity strategy, organizations can fortify their defenses and mitigate the risks posed by malicious actors in today's digital landscape.

MODULE 2: ANOMALY BASED DETECTION

Anomaly-based detection is another key technique used by Web Application Firewalls (WAFs) to identify and mitigate web-based attacks. Unlike signature-based detection, which relies on predefined patterns of known attacks, anomaly-based detection focuses on identifying deviations from normal or expected behavior within web traffic. Here's how anomaly-based detection works within a WAF:

1. Baseline Establishment: Anomaly-based detection begins with the establishment of a baseline or profile of normal behavior for the web application. The WAF analyzes historical traffic patterns and usage statistics to build a model of typical user behavior, application workflows, and network traffic characteristics.

The baseline captures metrics such as HTTP request rates, traffic volume, user session durations, HTTP response codes, and other relevant parameters that define normal operation.

2. Behavioral Analysis: As web traffic passes through the WAF, it is subjected to behavioral analysis techniques that compare observed patterns against the established baseline. The WAF monitors various aspects of HTTP traffic, including request parameters, headers, cookies, user agents, and session attributes.

Deviations from the baseline are flagged as anomalies that may indicate potentially malicious or suspicious behavior. Anomalies can manifest as unusual request rates, unexpected parameter values, abnormal user agent strings, or atypical session behaviors.

3. Statistical Analysis: Anomaly detection algorithms use statistical analysis methods to identify outliers and deviations from expected distributions of traffic attributes. These methods may include mean and standard deviation calculations, z-score analysis, or machine learning algorithms trained on historical data.

By analyzing traffic patterns statistically, the WAF can identify abnormalities that may signify malicious activity, such as unusually high request rates indicative of a denial-of-service (DoS) attack or unexpected parameter values indicative of parameter tampering.

4. Thresholds and Alarms: Anomaly-based detection relies on predefined thresholds and alarms to trigger alerts or actions when significant deviations from normal behavior are detected. Thresholds may be dynamically adjusted based on observed traffic patterns and environmental conditions to adapt to changing usage patterns.

When an anomaly surpasses a predefined threshold, the WAF may generate alerts, log the security event, or take automated actions to mitigate the detected threat, such as blocking the suspicious request or temporarily restricting access to the affected resource.

5. Adaptive Learning: To maintain accuracy and effectiveness, anomaly-based detection mechanisms incorporate adaptive learning capabilities that continuously update the baseline model in response to evolving traffic patterns and usage trends.

Adaptive learning algorithms dynamically adjust the baseline parameters and anomaly thresholds based on real-time observations, allowing the WAF to adapt to changes in application behavior and mitigate false positives caused by legitimate changes or fluctuations in traffic.

6. False Positive Management: Anomaly-based detection may generate false positives when legitimate but uncommon behaviors are mistaken for anomalies. To minimize false positives, WAF administrators can fine-tune anomaly detection parameters, adjust sensitivity levels, or define whitelists for known benign behaviors.

Continuous monitoring and analysis of false positives enable administrators to refine anomaly detection policies and improve the accuracy of the detection mechanism over time.

By employing anomaly-based detection, Web Application Firewalls can effectively identify and mitigate previously unknown or zero-day attacks, providing a proactive defense against emerging threats and unauthorized activities targeting web applications.

Anomaly-based detection stands as a pivotal technique utilized by Web Application Firewalls (WAFs) to discern and mitigate web-based attacks, operating in stark contrast to the signature-based approach that hinges on preconceived patterns of known attacks. Rather than relying on predefined signatures, anomaly-based detection scrutinizes web traffic for deviations from expected or normal behavior, allowing WAFs to identify and respond to emerging threats and zero-day attacks. The mechanism of anomaly-based detection within a WAF encompasses several key processes, each contributing to its efficacy in fortifying web applications against a myriad of cybersecurity risks.

The journey of anomaly-based detection commences with the establishment of a baseline or profile of normal behavior for the web application. This baseline serves as a reference point, encapsulating historical traffic patterns, usage statistics, and network characteristics to delineate typical user behavior and application workflows. Parameters such as HTTP request rates, traffic volume, user session durations, and response codes are meticulously analyzed to construct a comprehensive model of normal operation, laying the foundation for subsequent anomaly detection.

As web traffic traverses the WAF, it undergoes rigorous behavioral analysis aimed at comparing observed patterns against the established baseline. The WAF monitors various facets of HTTP traffic, including request parameters, headers, cookies, user agents, and session attributes, seeking out deviations or anomalies that may signify potentially malicious or suspicious behavior. Anomalies can manifest in diverse forms, such as unusual request rates, unexpected parameter values, abnormal user agent strings, or anomalous session behaviors, warranting further scrutiny and investigation.

To discern anomalies amidst the vast sea of web traffic, anomaly detection algorithms leverage statistical analysis methods that identify outliers and deviations from expected distributions of traffic attributes. These methods encompass mean and standard deviation calculations, z-score analysis, or machine learning algorithms trained on historical data. By statistically analyzing traffic patterns, the WAF can pinpoint abnormalities indicative of malicious activity, such as unusually high request rates suggestive of a denial-of-service (DoS) attack or unexpected parameter values indicative of parameter tampering.

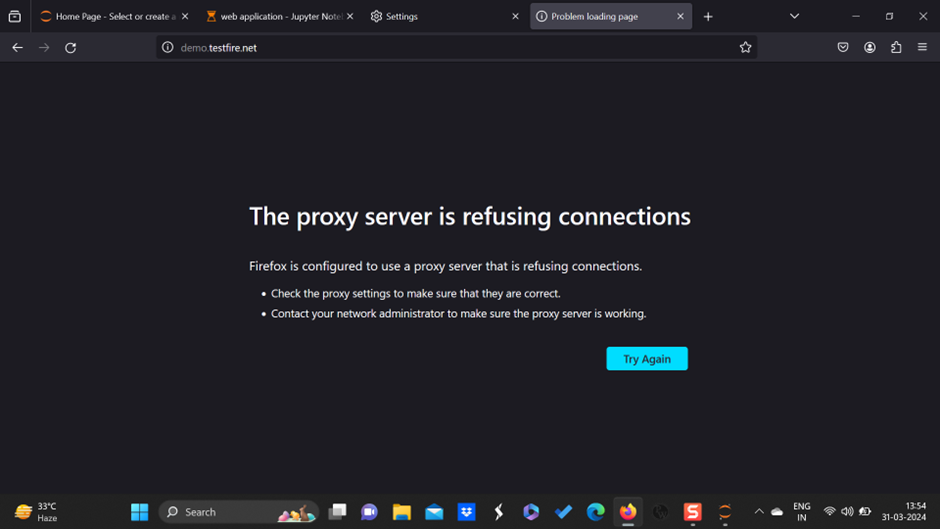
An essential aspect of anomaly-based detection lies in the establishment of thresholds and alarms, which serve as triggers for alerts or actions when significant deviations from normal behavior are detected. These thresholds may be dynamically adjusted based on observed traffic patterns and environmental conditions, allowing the WAF to adapt to evolving usage patterns and mitigate false positives. Upon surpassing a predefined threshold, the WAF may generate alerts, log the security event, or take automated actions to thwart the detected threat, such as blocking the suspicious request or temporarily restricting access to the affected resource.

To ensure the continued accuracy and efficacy of anomaly-based detection, adaptive learning mechanisms are incorporated into the detection framework, enabling the WAF to dynamically update the baseline model in response to evolving traffic patterns and usage trends. These adaptive learning algorithms adjust baseline parameters and anomaly thresholds based on real-time observations, facilitating adaptation to changes in application behavior and mitigating false positives arising from legitimate changes or fluctuations in traffic.

Nevertheless, the effectiveness of anomaly-based detection hinges on its ability to strike a balance between detection accuracy and false positive mitigation. False positives may occur when legitimate but uncommon behaviors are mistaken for anomalies, potentially leading to disruptions in service or user experience. To address this challenge, WAF administrators can fine-tune anomaly detection parameters, adjust sensitivity levels, or define whitelists for known benign behaviors. Continuous monitoring and analysis of false positives enable administrators to refine anomaly detection policies and enhance the accuracy of the detection mechanism over time.

In summary, anomaly-based detection serves as a proactive defense mechanism deployed by Web Application Firewalls to identify and mitigate web-based attacks. By scrutinizing web traffic for deviations from expected behavior and leveraging statistical analysis and adaptive learning mechanisms, anomaly-based detection enables WAFs to detect and respond to emerging threats and zero-day attacks, bolstering the security posture of web applications and safeguarding against a myriad of cybersecurity risks.

**7.3 SNAPSHOTS OF SYSTEM WITH BRIEF DETAIL OF EACH**

****

**Fig. 8 – Screenshot of a proxy server**

A proxy server acts as an intermediary between clients (such as web browsers or applications) and servers, facilitating communication between them. Here's an overview of how a proxy server works and its various functions:

1. Request Forwarding: When a client sends a request to access a resource (e.g., a web page, file, or service), it first connects to the proxy server instead of directly contacting the destination server.

The proxy server receives the client's request and forwards it to the destination server on behalf of the client.

2. Response Relay: Similarly, when the destination server responds to the request, it sends the response back to the proxy server.

The proxy server then relays the response to the client that initiated the request.

3. Anonymity and Privacy: Proxy servers can be used to enhance anonymity and privacy for clients by hiding their IP addresses from the destination server.

When clients connect to a destination server through a proxy, the server only sees the IP address of the proxy server, shielding the client's identity.

4. Content Filtering and Caching: Proxy servers can filter and cache web content to improve performance, security, and control over the content accessed by clients.

Content filtering capabilities allow proxy servers to block access to specific websites, domains, or content categories based on predefined rules or policies.

Content caching enables proxy servers to store copies of frequently accessed web pages, files, or resources locally. When clients request the same content again, the proxy server can serve it directly from its cache, reducing bandwidth usage and latency.

5. Security and Access Control: Proxy servers can enforce security policies and access controls to regulate traffic flow and protect internal networks from external threats.

By inspecting and filtering incoming and outgoing traffic, proxy servers can block malicious or unauthorized content, URLs, and requests.

Access control features enable administrators to restrict access to specific websites, services, or protocols based on user roles, permissions, or authentication credentials.

6. Load Balancing: In enterprise environments or large-scale deployments, proxy servers can be configured as load balancers to distribute incoming traffic across multiple backend servers.

Load balancing helps optimize resource utilization, improve scalability, and enhance fault tolerance by evenly distributing the workload among backend servers.

7. Protocol Support: Proxy servers support various network protocols, including HTTP, HTTPS, FTP, SOCKS, and DNS. Different types of proxy servers, such as HTTP proxies, SOCKS proxies, and transparent proxies, specialize in specific protocols and use cases.

Overall, proxy servers play a versatile role in network infrastructure, providing functionality such as request forwarding, response relay, anonymity, content filtering, caching, security, access control, load balancing, and protocol support. They are commonly deployed in enterprise networks, cybersecurity solutions, content delivery networks (CDNs), and internet service providers (ISPs) to improve performance, security, and manageability of network traffic.

A proxy server serves as an intermediary between clients, such as web browsers or applications, and servers, facilitating communication between them. When a client initiates a request to access a resource, it connects to the proxy server instead of directly contacting the destination server. The proxy server then forwards the request to the destination server on behalf of the client. Similarly, when the destination server responds to the request, the proxy server relays the response back to the client.

Proxy servers can enhance anonymity and privacy for clients by hiding their IP addresses from the destination server, thus shielding their identity. Additionally, proxy servers offer functionalities like content filtering and caching, enabling them to improve performance, security, and control over the content accessed by clients. They can filter web content based on predefined rules, block access to specific websites or domains, and cache frequently accessed content to reduce bandwidth usage and latency.

Proxy servers also enforce security policies and access controls to regulate traffic flow, protect internal networks from external threats, and restrict access to specific websites or services based on user roles or permissions. In enterprise environments, proxy servers can be configured as load balancers to distribute incoming traffic across multiple backend servers, optimizing resource utilization, scalability, and fault tolerance.

Supporting various network protocols such as HTTP, HTTPS, FTP, SOCKS, and DNS, proxy servers play a versatile role in network infrastructure, commonly deployed in enterprise networks, cybersecurity solutions, content delivery networks (CDNs), and internet service providers (ISPs) to improve performance, security, and manageability of network traffic.

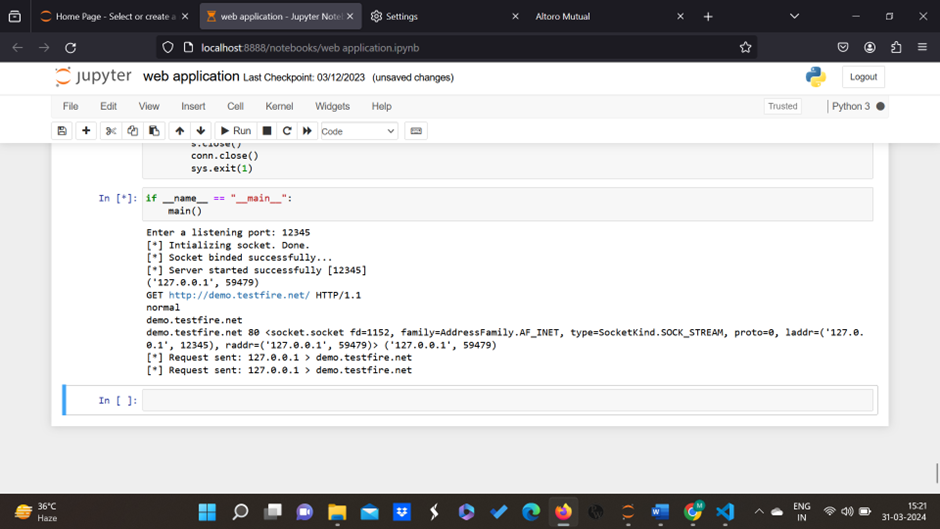


Fig. 8 –Jupyter Notebook Machine Learning model

Anomaly-based detection utilizing machine learning (ML) models represents a powerful and sophisticated approach to identifying and mitigating web-based attacks by detecting deviations from normal behavior in network traffic patterns. This methodology stands in contrast to signature-based detection, which relies on predefined patterns of known attacks. Instead, anomaly-based detection leverages advanced ML algorithms to analyze network traffic data and identify anomalous patterns indicative of potential threats. This comprehensive approach to cybersecurity encompasses various processes and components, each contributing to the overall efficacy of anomaly-based detection using ML models.

The process of anomaly-based detection with ML models begins with the collection and preprocessing of network traffic data. This data encompasses a wide range of attributes, including HTTP requests, response codes, headers, payloads, timestamps, and other relevant information. Data preprocessing involves cleaning, transforming, and formatting the raw data to make it suitable for training ML models. This may involve tasks such as data normalization, feature scaling, and handling missing values to ensure the quality and consistency of the dataset.

Once the data is preprocessed, the next step is feature extraction, where relevant features or attributes are extracted from the network traffic data. Feature extraction plays a crucial role in anomaly detection, as it determines the input variables used to train the ML models. Various techniques can be employed for feature extraction, including statistical analysis, frequency analysis, time-series analysis, or domain-specific feature engineering tailored to the characteristics of web traffic data. The goal is to identify informative features that capture the underlying patterns and dynamics of normal and anomalous network behavior.

With the extracted features in hand, ML models are then trained using labeled datasets consisting of both normal and anomalous traffic samples. Supervised learning algorithms are commonly used for anomaly detection, where the models learn to distinguish between normal and anomalous traffic patterns based on the extracted features. Popular supervised learning algorithms for anomaly detection include support vector machines (SVM), decision trees, random forests, or deep learning architectures such as convolutional neural networks (CNN) or recurrent neural networks (RNN).

During the training process, the ML models are exposed to labeled examples of normal and anomalous network traffic, allowing them to learn the underlying patterns and characteristics of each class. The models aim to generalize from the training data and develop a robust understanding of what constitutes normal behavior in the network. By learning to recognize deviations from normal behavior, the models can effectively identify potential threats and anomalies in real-time.

One of the key advantages of anomaly-based detection with ML models is its ability to adapt and evolve over time. As network traffic patterns change and new threats emerge, ML models can be updated and retrained using fresh data to maintain their effectiveness. This adaptability is crucial for staying ahead of evolving cyber threats and ensuring robust protection against emerging attack vectors.

In addition to adaptability, anomaly-based detection with ML models offers scalability and flexibility, making it suitable for a wide range of applications and environments. ML models can be deployed on-premises, in the cloud, or as part of managed security services, providing organizations with the flexibility to choose the deployment model that best fits their needs. Furthermore, ML models can be tailored to specific use cases and environments, allowing organizations to customize their anomaly detection systems to suit their unique requirements.

Another benefit of anomaly-based detection with ML models is its ability to provide insights into network behavior and security posture. By analyzing network traffic data, ML models can uncover hidden patterns, trends, and correlations that may not be apparent through traditional methods. This insight can help organizations gain a deeper understanding of their network infrastructure, identify potential vulnerabilities, and make informed decisions about security policies and controls.

Despite its many advantages, anomaly-based detection with ML models also poses some challenges and limitations. One of the main challenges is the need for labeled training data, which can be time-consuming and resource-intensive to collect and annotate. Additionally, ML models may struggle to generalize from limited or biased training data, leading to overfitting or poor performance on unseen data.

Furthermore, anomaly-based detection with ML models may struggle to cope with the dynamic and evolving nature of network traffic patterns. As network environments change over time, ML models must be continuously updated and retrained to remain effective. This requires ongoing monitoring, maintenance, and tuning of the anomaly detection system to ensure optimal performance.

Despite these challenges, anomaly-based detection with ML models remains a valuable and effective approach to cybersecurity. By leveraging advanced ML algorithms, organizations can detect and mitigate web-based attacks more effectively, providing robust protection against a wide range of threats.



Fig. 9 – Model give results based on SQL Query

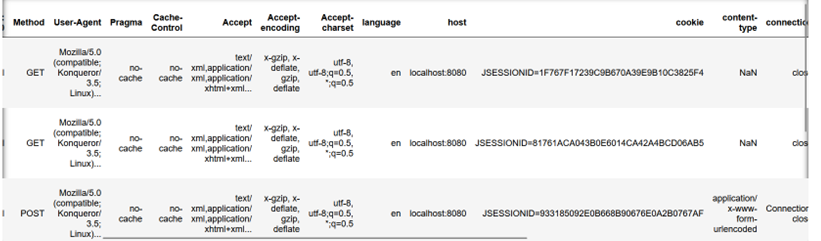
The model operates by analyzing the structure and content of SQL queries submitted by users to the web application. It evaluates various characteristics of the queries, such as syntax, semantics, and parameters used, to assess their safety. When a query is submitted, the model scrutinizes it for any signs of potential SQL injection vulnerabilities or suspicious patterns that deviate from normal behavior. If the model determines that the query is safe and does not pose a security risk, it indicates that the request is legitimate and can be executed without concern. On the other hand, if the model detects anomalies or irregularities in the query that could indicate a SQL injection attempt, it raises an alert or flag to notify administrators or security personnel about the potential threat.

The model utilizes a combination of techniques and methodologies to analyze SQL queries and make predictions about their safety. This may include rule-based heuristics, pattern matching algorithms, or more advanced machine learning approaches. Rule-based heuristics involve predefined rules or patterns that characterize safe or malicious SQL queries based on known attack vectors and common exploitation techniques. Pattern matching algorithms search for specific sequences or structures within the SQL query that resemble patterns commonly associated with SQL injection attacks. Machine learning approaches leverage historical data and labeled examples of safe and malicious SQL queries to train predictive models that can generalize and make accurate predictions about the safety of new queries.

In addition to detecting SQL injection attacks, the model may also provide insights into the nature of detected anomalies. It can analyze the characteristics of anomalous queries, such as the type of injection technique used (e.g., union-based, boolean-based, time-based), the target database or table, and the potential impact of the attack. This information can help security teams understand the nature of the threat and take appropriate measures to mitigate the risk, such as patching vulnerabilities, implementing input validation mechanisms, or updating security policies.

Overall, a model that evaluates SQL queries for safety plays a critical role in protecting web applications against SQL injection attacks and maintaining the integrity and confidentiality of sensitive data stored in backend databases. By continuously monitoring and analyzing SQL queries in real-time, the model enables organizations to detect and respond to potential threats promptly, reducing the risk of data breaches and security incidents. Additionally, the insights provided by the model can inform proactive security measures and help organizations improve their overall cybersecurity posture.

**7.3 BRIEF DESCRIPTION OF DATABASE WITH SNAPSHOTS**

****

**Fig. 10 – Database CSIC 2010**

The dataset consists of 60,000 entries representing simulated HTTP requests designed to assess the efficacy of a Web Application Firewall (WAF) in defending against cyber threats, notably SQL injection attacks. Each row encapsulates diverse features pertinent to HTTP requests, cataloged across several columns.

The "Method" column delineates the employed HTTP method—GET, POST, PUT, DELETE, or HEAD—signifying the action intended for the requested resource. Information pertaining to the client's browser or application is encapsulated in the "User-Agent" column, disclosing details such as the browser type, operating system, and device. Headers pertinent to caching and controlling proxy behavior, namely "Pragma" and "Cache-Control," a

recorded in their respective columns, elucidating directives for caching. The "Language" column denotes the preferred language settings articulated in the Accept-Language header, facilitating content negotiation and localization. The target server's hostname or IP address is disclosed in the "Host" column, elucidating the destination for the HTTP request.

Moreover, the dataset includes a "Cookie" column housing cookie values transmitted by the server, pivotal for session management and user authentication. Lastly, the "Content-Type" column delineates the MIME type of the content within the HTTP request body, facilitating data format and encoding comprehension.

This dataset furnishes a comprehensive array of features, enabling rigorous evaluation of the WAF's proficiency in discerning and mitigating potential security vulnerabilities, particularly SQL injection attempts and other cyber threats. Through meticulous analysis of request patterns and attributes, the WAF can detect anomalous behavior indicative of malicious activity, fortifying web applications against diverse cyber risks.

**CHAPTER 8**

**CONCLUSION AND FUTURE SCOPE**

Web Application Firewalls (WAFs) constitute an indispensable component in contemporary cybersecurity infrastructure. Throughout this project, we have delved into the significance of WAFs in fortifying web applications against a plethora of threats, ranging from SQL injection to cross-site scripting (XSS), thereby bolstering the overall security posture of organizations.

Our exploration underscores the pivotal role WAFs play as a frontline defense mechanism, intercepting and scrutinizing HTTP requests and responses to weed out malicious traffic. Through the deployment of signature-based detection, anomaly detection, and behavioral analysis techniques, WAFs emerge as vigilant gatekeepers, capable of discerning and mitigating both known and emerging threats in real-time.

Furthermore, our analysis reveals that the adoption of WAFs not only fosters a more resilient security environment but also facilitates compliance with regulatory standards such as the Payment Card Industry Data Security Standard (PCI DSS), Health Insurance Portability and Accountability Act (HIPAA), and General Data Protection Regulation (GDPR), thus ensuring the integrity and confidentiality of sensitive data.

**FUTURE SCOPE**

Looking ahead, the trajectory of WAFs points towards a horizon characterized by continued innovation and evolution. The following avenues represent promising areas for future exploration and development:

1. Integration of Machine Learning and AI: The integration of machine learning and artificial intelligence holds immense potential in augmenting WAF capabilities, enabling dynamic threat detection and response mechanisms that adapt to evolving cyber threats.

2. Advancements in Behavioral Analysis: Future iterations of WAFs will likely refine their behavioral analysis capabilities, leveraging insights gleaned from user and application behavior to discern anomalies indicative of security breaches.

3. Focus on API Security: With the proliferation of web APIs, WAFs will expand their purview to encompass API security, safeguarding against a diverse array of threats targeting API endpoints and data exchanges.

4. Adaptation to Serverless Architectures: As organizations gravitate towards serverless architectures, WAFs will need to adapt accordingly, providing robust security measures tailored to serverless environments and functions.

5. Seamless Integration with DevOps Practices: WAF solutions of the future will seamlessly integrate with DevOps workflows, streamlining the deployment and management of security policies within agile development environments.

6. Embrace of Zero Trust Principles: WAFs will play a pivotal role in implementing zero-trust principles, enforcing granular access controls and scrutinizing all traffic traversing the network perimeter to mitigate the risk of lateral movement and unauthorized access.

In essence, the future of Web Application Firewalls is characterized by a commitment to innovation, adaptability, and alignment with evolving technology.

**REFERENCES**

1. Razzaq, A., Hur, A., Shahbaz, S., Masood, M., & Ahmad, H. F. (2013, March). Critical analysis on web application firewall solutions. In 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS) (pp. 1-6). IEEE.

2. Victor Clincy, Hossain Shahriar, (2018) 42nd IEEE International Conference on Computer Software & Applications: Web Application Firewall: Network Security Models and Configuration.

3. Rizki Agung Muzaki; Obrina Candra Briliyant; Maulana Andika Hasditama; Hamzah Ritchi (October 2020).

4. Improving Security of Web-Based Application Using Mod Security and Reverse Proxy in Web Application Firewall.

5. Adem Tekerek; Cemal Gemci; Omer Faruk Bay(2014). Development of a Hybrid Web Application Firewall to Prevent Web Based Attacks.

6. Simon Applebauma, Tarek Gaberb , Ali Ahmedc, (2021) Signature-based and Machine-Learning-based Web Application Firewalls.

7. Aref Shaheed ,M. H. D. Bassam Kurdy(June 2022). Web Application Firewall Using Machine Learning and Features Engineering.

8. Ali Moradi Vartouni; Saeed Sedighian Kashi; Mohammad Teshnehlab (2018). An Anomaly Detection Method to Detect Web Attacks Using Stacked Auto-Encoder.

9. Sezer TOPRAK,(2022). Web Application Firewall Based on Anomaly Detection using Deep Learning.

10. Ali Moradi Vartouni , Mohammad Teshnehlab , Saeed Sedighian Kashi(February 2019).Leveraging deep neural networks for anomaly-based web application firewall.

11. A. Tekerek , O.F. Bay(2019).Design and implementation of an artificial intelligence-based web application firewall model.

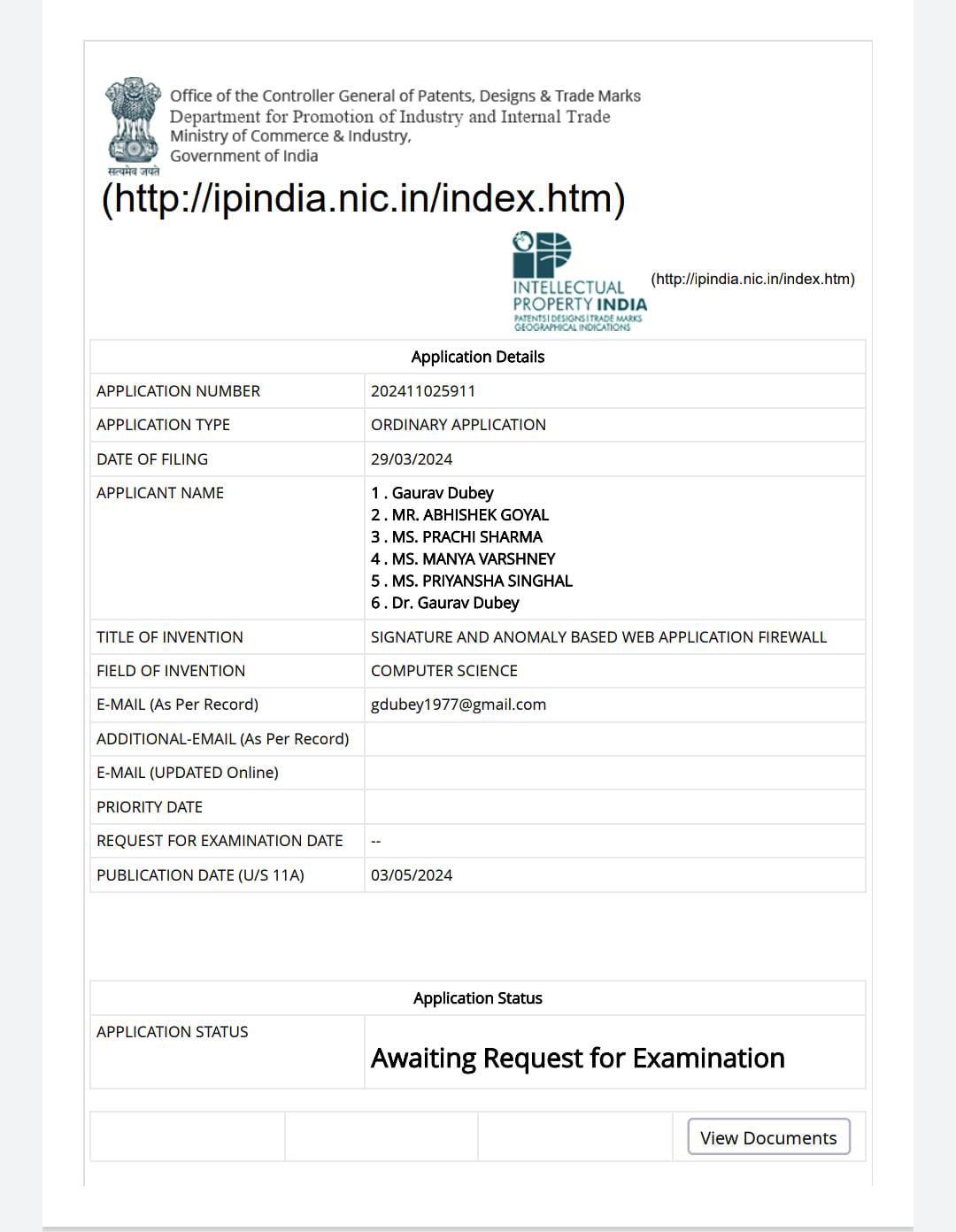
12. Babu R. Dawadi , Bibek Adhikari, Devesh Kumar Srivastava (February 2023) . Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks.

13. Jingxi Liang, Wen Zhao, Wei Ye (2017).Anomaly-Based Web Attack Detection.

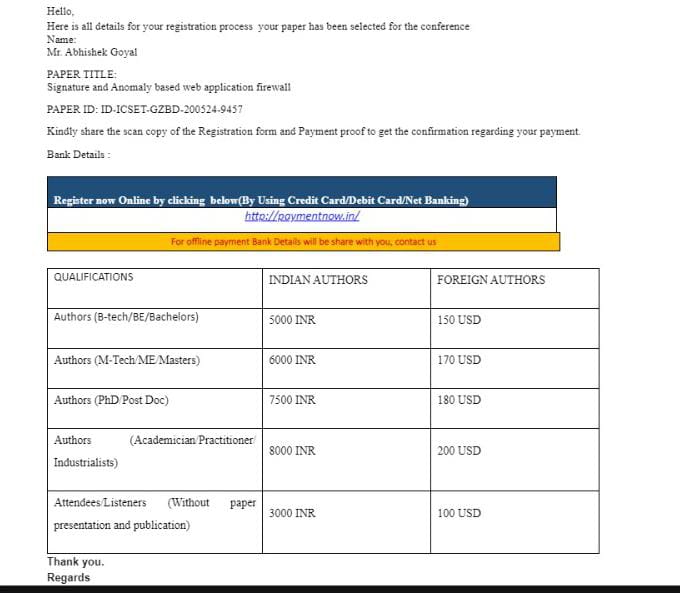
14. Yao Pan, Fangzhou Sun ,(2019) . Detecting web attacks with end-to-end deep learning.

15. Tao Yi , Xingshu Chen (2023).Review on the application of deep learning in network attack detection

**PATENT PUBLICATION PROOF :**

****

**ACCEPTANCE PROOF:**

****