Machine Learning Based Web Application Firewall (SMA^3 WAF)

By

Maryam Murtaza

Aden Bin Farrukh

Muhammad Ali

Ayesha Seemab

Malik Shehroz Ahmed

Supervised by:

Asst Prof Dr. Mian Muhammad Waseem Iqbal

Submitted to the faculty of Department of Computer Software Engineering,
Military College of Signals, National University of Sciences and Technology, Islamabad,
in partial fulfillment for the requirements of B.E Degree in Computer Software Engineering.

June 2022

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report
"Web Application Firewall Using Machine Learning"
is carried out by

Maryam Murtaza, Aden Bin Farrukh, Muhammad Ali, Ayesha Seemab, and Malik Shehroz Ahmed

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Computer Software Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor
Asst Prof Dr. Mian Muhammad Waseem Iqbal
Department of IS, MCS

Date: June 24th, 2022

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues and most of all supervisor, Asst Prof Dr. Mian Muhammad Waseem Iqbal without your guidance.

The group members, who through all adversities worked steadfastly.

Plagiarism Certificate (Turnitin Report)

This thesis has 12% similarity index. Turnitin report endorsed by Supervisor is attached.

Maryam [illegible]

Maryam Murtaza

NUST Serial no 00000241682

[illegible]

Ayesha Seemab

NUST Serial no 00000259928

Aden

Aden Bin Farrukh

NUST Serial no 00000216406

[illegible]

Malik Shehroz Ahmed

NUST Serial no 00000266964

[illegible]

Muhammad Ali

NUST Serial no 00000267339

____

Signature of Supervisor

ABSTRACT

In the field of technology and cyber security, cyber-attacks on web applications are constantly increasing. One of its most promising outcomes has been the availability of powerful web applications and low-cost internet. However, threats such as Cross site scripting, DoS, and Web attacks are becoming much more of a concern due to such technological advancements. This has essentially rekindled the interest of engineers in technological security and has prompted them to conduct extensive research on the topic.

Web applications hold vital information about both the company and its clients. However, with the rise of cybersecurity attacks, they are being hacked more often, prompting administrators to look for ways to protect them from the black hat culture. The project "Machine Learning based Open-source Web Application Firewall" is about a firewall built and deployed in a live web application. The project aims to detect incoming requests in real-time using a Machine Learning detection engine and forward them to web application firewall. It is directed towards the server if it is a benign request; otherwise, it is routed to a log, where it is analyzed.

When implemented, this project can be beneficial for administrators and network security experts to gather valuable information about a hacker's operation and its footprints, which can substantially contribute to creating new rules and mechanisms to deter potential attacks.

Table of Contents

List of Figures

1. INTRODUCTION

1.1 Overview

There is nothing that has 100% security, but we must try to provide security in various ways in order to maintain our privacy on the internet, but the web application firewall is an important element in protecting websites, user's privacy, is an additional option on sites. This makes it difficult for hackers to attack the site and steal data from the site. According to a recent survey, 72% of the companies surveyed have hacked websites / applications in the last 24 months. The most successful attacks occurred at the application layer (Layer 7 of the OSI model). This is still the most vulnerable area. The lower layers are not always secure, but network protocols have been enhanced over the years to levels that are extremely difficult to hack and rarely vulnerable. Furthermore, because operating systems, web servers, and cryptographic engines are usually offered by a small number of manufacturers, they are relatively safe and have a large enough user base to detect and address flaws early. there is. However, web applications are the opposite, usually unique (even if they share common code such as frameworks) and are usually only by a single client or only to a single client. It will be managed. This reduces the average security of your code (because each company does not have access to expertise in such a fragmented market) and is less likely to detect vulnerabilities before they are exploited. Therefore, web applications are 1/10 more vulnerable than browsers / operating systems.

1.2 Problem Statement

Today's world is a world of technology. Humans use technology to travel, communicate, and learn for everything. Everything in our lives is directly or indirectly dependent on technology. However, everything comes with pros and cons. With all these benefits, technology also brings some serious concerns. One of them is Data Security. This data can be on a web application, or a server, or any system.

With technology and the Internet making communication and data sharing across networks simpler by the day, the issue of compromised Internet and network security is becoming more prevalent. An attacker can obtain access to a web application and modify or delete information stored in a database. The three fundamental aspects of security that must be addressed are secrecy, permitted access to information, and data integrity.

According to Shannon Williams new report, between 2020 and 2021, the number of malicious web application attacks have climbed 88%. In 2019 alone, cyber-attacks cost us 2.1 trillion dollars; now, that is a tremendous amount of money wasted just because of insufficient security measures.

Existing protection standards, such as a network firewall, scanning the web application, and backing up the data, fail to stop an intruder's activities if he successfully acquires a loophole inside a web application or has disguised his identity for a firewall-enabled system. In addition, these existing security solutions have certain limitations associated with them. First of all, they collect a massive amount of data, and this feature poses issues like a greater chance of false positives and greater power. Secondly, most importantly, they only mitigate the attack but do not provide information about the attackers and the tools they use.

Security managers are focused on preventing network and web attacks by preventing all types of penetration, but nothing is done to engage the attacker in an in-depth analysis of the attacker's motivations, strategies, and experience. Our project bridges this gap by combining detection and deception. With the number of attackers and the sophistication of their Dos, Scanning, and Web attacks rapidly increasing, it is more important than ever to monitor and record their activities and collect information that can help improve existing security tools.

1.3 Approach

We first installed a firewall (MODSecurity) on the Kali Linux to test vulnerabilities present in a vulnerable web application (DVWA) and check the traffic that goes through it. Four different types of attacks were carried out namely Cross-site scripting (XSS), SQL injection (SQLi), Web session hijacking and Distributed Denial of service (DDoS). A machine learning model then takes in the logs from both web app and Mod Security as input and process this data to get better.

This model processes the network requests in real time, these python scripts identify if the traffic is legitimate or malicious based on the previous network traffic. Legitimate traffic is directed towards the web application and malicious one gets dropped, both are added to logs irrespective of type and labeled accordingly.

1.4 Scope

The real time machine learning based protection allows or project to be used in a wide variety of sectors including e-commerce, sensitive organizations, healthcare and even in security sectors. The project's primary purpose is to supplement and incorporate existing security tools while also providing resistance to attacks and minimize firewall evasion. This project combines a machine learning model with existing tools in the hopes to combine the theoretical expertise and practical experience to improve the overall security of web application.

1.5 Purpose

The following are the key goals that this project aims to achieve:

- Installing a web application firewall that logs all network traffic which is then used to make a training dataset for our machine learning model based on the OWASP top 10 attacks [1].
- Keeping detailed logs allowing the machine learning model to learn and reduce risks by analyzing the trends and patterns of these attacks which assists in the development of an effective mechanism.

1.6 Contributions

This project was created primarily to protect web applications against DOS, SQLI, XSS, and other Web attacks by providing insight into the attackers' activities and allowing researchers to better understand the attackers' strategies. As a result, they can integrate and improve existing systems.

2. Literature Review

This chapter will go over the philosophy behind using a web application firewall and some of its features and why it is used. The definition of a web application will be compared to more official IT standards and some less formal ones to try to clarify what it is. Finally, the integration of a web application with already existing tools and coded programs to effectively detect cyber threats and assist engineers in their efforts to protect personal privacy from being compromised and eroded.

2.1 Security Detection Reasons

With the steadily influencing universe of broadcast communications, the requirement for safe information transport over open gateways (the Internet) is developing. As per late reports, the quantity of hackers on the planet isn't just increasing, yet additionally moving to worldwide scale. CBC detailed on a mass convergence of the hacking local area in and around China, guessing and guaranteeing that this was completely prompted by the Chinese government to start accessing the knowledge and records of enterprises from the West or Western world states (Europe and America), and for global business and government spying (Canada Television). Whether this is valid, it mirrors a pattern in which individuals are turning out to be more and more worried about the security of their information. Due to the occasions that are happening around us, it is turning out to be increasingly normal that we as a whole have a type of safety to forestall unapproved admittance to our information, going from public frameworks to totally secluded frameworks with next to zero external access, and the capacity to perceive these assaults is turning out to be very important. Organizations who give security gear are starting to go to the hacking society, similarly as house alert installers have gone to ex-hoodlums and criminals, to have the option to identify these attacks and controls. Reports will be sent back for audit, so the organizations can foster their projects and assemble new guidelines and acknowledgment calculations.

2.2 What is Web Application Firewall?

The Web Application Firewall (WAF) is a security module for application proxy devices that defends back-end web application servers against a number of threats. Application protection is a crucial additional layer of security since it protects against a variety of application layer security risks that aren't generally covered by traditional network layer attack detection and prevention systems. WAFs' are of two types. One is open-source WAFs' that are accessible on the web and are utilized effectively with no expense. The other one is closed source WAFS' which are likewise called as restrictive. Web Application Firewalls (WAF) have developed to shield web applications from assault. A mark - based WAF answers dangers through the execution of utilization explicit principles which block vindictive traffic. Be that as it may, these guidelines should be persistently adjusted to address advancing dangers. WAF has a capacity to manage assaults on the application layer that standard firewall doesn't.

At first, data or information that might be helpful to an aggressor, like on a site, is purposely passed on open for the assailant to get to. So, a web application firewall deludes the gatecrasher into imagining that he has accessed a real web application. Every one of the aggressors' way of behaving and communications with the gadget can be logged and continually followed while they are attempting to get to the application. At the point when introduced inside an application, a WAF dismisses all risky traffic, safeguarding the application from attack. Thus, a WAF gathers data about new hacking strategies, yet in addition shields the application from threats

2.3 Working of Web Application Firewall

WAF decomposes HTTP requests and uses a series of algorithms to determine which portions of the dialogue are benign and which are malicious. The WAF can evaluate its material using one of two ways, or a mix of the two.

- **Whitelisting:** The whitelist technique implies that the WAF rejects all requests by default and only permits those that are known to be secure. The following is a list of IP addresses that have been identified as being protected. Asset escalation is lower on whitelists than on blacklists. The drawback of using a whitelist is that it may mistakenly block traffic that is not harmful. You can use the internet to project yourself and become more productive, but it might be unclear.

- **Blacklisting:** To block harmful online traffic and secure websites or web applications, the blacklist technique enables packets to pass by default and employs predefined markers / filters. A harmful package is depicted in the regulation summary. Because public websites and online apps receive a lot of traffic from new IP addresses that are neither malicious nor benign, blacklists are a good fit. The disadvantage of the blacklist strategy is that it focuses on assets and resources. Rather than depending on the IP address inferred by default, more data is required to channel the packet from an explicit quality standpoint.

- **Hybrid security:** The mixed security model supports both blacklists and whitelists.

  Regardless of the security model used by the WAF, the WAF will eventually inspect HTTP communication and try to eliminate vengeful traffic before it arrives at the server for processing.

2.4 Web Application Firewall History

The idea of Web Application Firewall was introduced in market in the late 1990s during a time when web server attacks were becoming more prevalent. Previously, network layer-based firewalls were being used but they didn’t prevent application layer-based attacks on web applications. From there the need to protect web applications from cyber-attacks arose leading to the birth of web application firewalls.

2.5 Web Application Firewall Types

Network-based WAFs are usually hardware-based and implemented locally on-premises through a dedicated appliance, reducing latency. Most major network-based WAF vendors enable for the replication of rules and settings over several computers, enabling for large-scale planning, setup, and installation. The biggest disadvantage of this sort of WAF item is the cost, which is a result of both immediate capital consumption and continuous operational costs for maintenance.

The WAF that runs on the host can be fully integrated into the application code. Cost savings and better customization choices are two advantages of host based WAF execution. Because they require host-side assets and complex processing to expedite the usage of those assets, host-based WAFs can be a monitoring test.

A **cloud based WAF** responds to associations that require turnkey elements that require minimal resources to run and maintain at minimal cost. Cloud WAF is straightforward to set up, is available on a subscription basis, and frequently just requires basic DNS or proxy modifications to divert application traffic. Filtering your organization's web application traffic with an external provider can be very tedious, but this technique uses a comparative strategy to protect your applications in different areas of support and protect them from application layer attacks.

2.5.1 Open Source WAFs vs Commercial

Both free source and commercial WAF solutions are available. ModSecurity, Naxsi, and WebKnight are some of the most popular open source suppliers. F5, Barracuda, and Cloudflare are among of the most popular commercial providers.

2.5.2 WAF vs Firewall

A firewall is a generic word for software that secures a computer network by screening incoming data packets. There are various categories under this wide description that are categorised according to the sort of protection offered and how it is supplied. Stateful inspection, proxies, packet filtering, and NGFW are among these duties. Another firewall categorization is WAF, which distinguishes itself by how information packets are expressly channelized. WAFs are unique in that they focus solely on online attackers at the application layer, but other forms of firewalls, such as packet filtering and stateful inspection, are unable to guard against these types of assaults. The WAF functions similarly to a proxy firewall, but with an emphasis on Layer 7 application logic.

2.6 Advantages of WAF

Traditional firewalls are outclassed by WAF because it gives better insight into sensitive application data passing through the HTTP application layer. This helps to avoid attacks on the application layer that aren't impacted by standard network firewalls, such as:

Cross-site scripting (XSS) attacks are a sort of injection in which malicious content is injected into otherwise trustworthy websites. XSS attacks occur when an attacker uses a web application to transmit malicious code to an alternate end client, usually in the form of a browser-side script. The flaws that allow these attacks to succeed are so pervasive that they may be found whenever a web application utilises them in its output without first authorising or encoding client input.

Structured Query Language (SQL include injecting or "injecting" SQL queries into a client-server application using data from the client. It pulls sensitive information from the database, updates database information (insert, update, delete), conducts housekeeping operations on the database (such as terminating the DBMS), and is included in the DBMS document after successful SQL Infusion exploitation. You can get the contents of a certain record back. - A framework exists and sends commands to the framework you're working on a temporary basis. A SQL injection attack is one in which a SQL command is injected into the information plane input and impacts the execution of preset SQL instructions.

Web session hacking The Session Hijacking attack comprises of the double-dealing of the web meeting control system, which is regularly overseen for a meeting token.

DDoS attacks overload your business by flooding it with traffic to the point that it can no longer service its users. This sort of attack can be handled by both network firewalls and WAFs, although they are aimed at different tiers of the assault.

Another advantage of WAF is that it allows you to defend your web-based application without having to go into the source code. While host-based WAFs may be integrated into your application code, cloud-based WAFs are ideal for keeping your applications out of the public eye. Furthermore, the cloud WAF is simple to set up and maintain, and it offers rapid virtual patch provisioning, allowing companies to swiftly adjust settings in response to newly discovered threats.

2.7 Importance

WAF is good for various efforts to deliver products over the Internet, such as web banking, social media platforms, and mobile application engineers, to prevent information leakage and data leakage. Many sensitive information, such as visa information and customer records, is stored in back-end records opened through web applications. Attackers frequently target these programmes in order to gain useful data. Banks, for example, can employ WAF to help them comply with the Payment Card Industry Data Security Standard (PCI DSS), a set of guidelines aimed at ensuring the security of cardholder data (CHD). One of the 12 PCI DSS integrity criteria is the use of a firewall. This uniformity applies to all CHD-related businesses. As more modern organizations leverage mobile applications and the evolving Internet of Things (IoT), the number of transactions occurring in the web-powered application layer is increasing. As a result, WAFs are an important part of a state-of-the-art enterprise security model.

3. Software Requirement Specifications

The Software Requirements Specification (SRS) provides a detailed description of the requirements for the Web application firewall using machine learning (SMA3). This allows for a complete understanding of what is to be expected of the SMA3 that is to be constructed. The idea is to customize the open-source web firewall using machine learning to obtain the functionality of closed firewall. This is supposed to define the capabilities and necessities of SMA3, that serves as a manual to the builders on one hand and a software program validation for the cease customers at the other.

3.1 Overall description

3.1.1 Product Perspective

SMA3 will protect the vulnerable data of websites. It will detect and prevent the incoming malicious traffics. Both intrusion detection and prevention becomes more effective. As the WAF is used ,it develops its own set of rules for the certain type of attack using Machine Learning and it will better and better over time.

3.1.2 Product Functions

- The WAF will secure web application.
- The WAF will detect and prevent any malicious threats.
- The Web application firewall will use auditing and logging system to help in decision making.

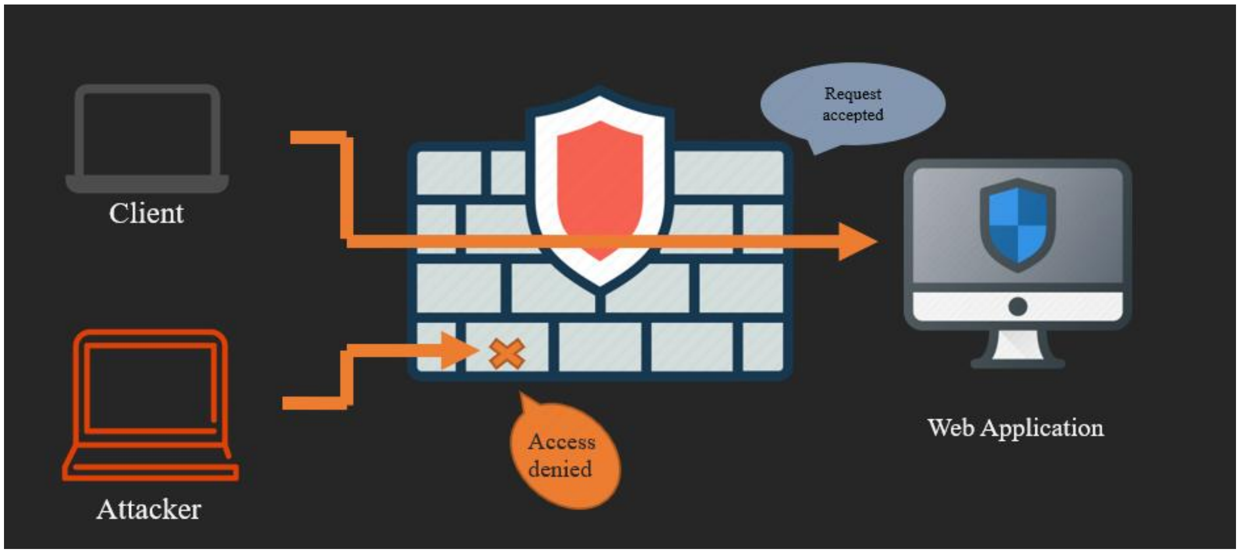


Figure 3.1 Working of Proposed WAF

3.2 External Interface Requirements

3.2.1 User Interface

- Home Screen: A dashboard which offers a menu with a list of option that are available.
- Turn On/Off: A button to turn firewall on or off.
- Log Analysis: A window which give the list which shows the malicious packets that were present in the network traffic.
- Help: A window that helps you learn about the web application firewall.

3.2.2 Hardware Interface

Two laptops with Linux operating system running in it and are connected using network interfaces.

3.2.3 Software Interface

Python: using python language cause its more interactive and easier to use.

Machine learning libraries: scipy, numpy, tenserflow etc.

3.3 Non-Functional Requirements

- Downtime should not be more than 10 seconds.
- False positives and negatives must not be more than 20%.
- In case of an attack alert should be provided to alert the analyst.

4. Design and Development

4.1 System Overview

SMA 3 will protect the vulnerable data of websites. It will detect and prevent the incoming malicious packets in traffic. Both intrusion detection and prevention become more effective. As the WAF is used, it develops its own set of rules for the certain type of attacks using Machine Learning and it will get better and better over time.

4.2 System Architecture

4.2.1 Architectural Design

The architectural design of the SMA^3 WAF is Event Driven Architecture. Upon the detection of an event, the system acts accordingly. We are using python as our main programming language in which Machine learning model is developed. The model waits for the request and train the core rule of the firewall in such a way that the firewall will automatically accept or reject the request. If request accept the client will access the application. In this project, the user will request the web application, Web Application Firewall will identify if the request is malicious, it will deny the request otherwise it will allow the user to access the application.

4.3 Component Design

In this component design section, we will look at each component of the SMA3WAF more systematically. Each component will have a functional description and closer detail. The main component of this system is machine learning base web application firewall component that have been studies below through activity diagram.

4.4 Human Interface

4.4.1 Interface Overview

- **Home Screen**: A dashboard which offers a menu with a list of option that are available.
- **Turn On/Off**: A button to turn firewall on or off.
- **Log Analysis**: A window which give the list which shows the malicious packets that were present in the network traffic.
- **Help**: A window that helps you learn about the web application firewall.

4.4.2 Screen Images

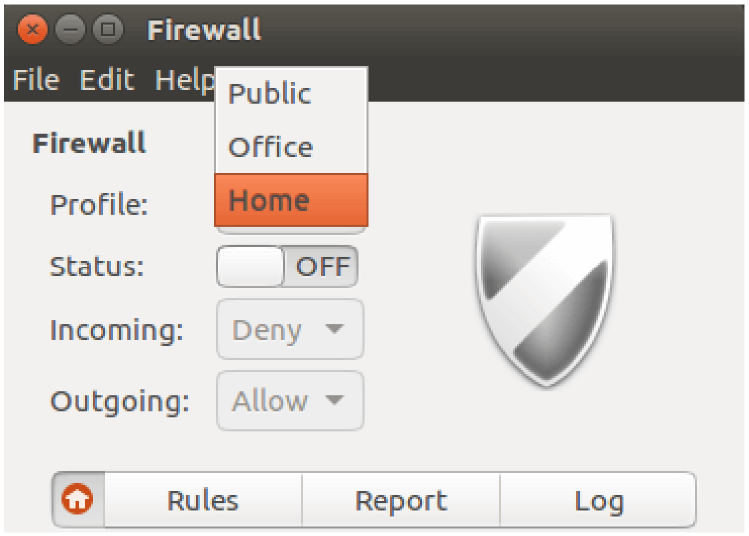


Figure 4.1 The user interface of proposed firewall will look like this.

4.4.3 Screen Objects and Actions

The "Home Screen" will display the main menu. The Turn on/off button will show the status of the firewall. If the firewall will be on or active, Log analysis will show the malign packets present in the traffic. Help Window will guide the user to learn about web application firewall.

4.5 Requirements Matrix

Label	Requirement	Component
	The user should turn firewall on using a button provided in user interface	Firewall component on the home page
	The user should have established a network connection	NILL
	The rules for web application firewall must be written in such a way that the firewall can learn to protect in an easy way.	NILL
	There should be a manual or guide to understand the working of the application.	Help component on menu bar
	There is a data of traffic	Log button on home screen
	Access the request	Outgoing traffic show

5. Test & Evaluation Methodolgy

5.1 Test Objectives

The goal is to block malicious HTTP / S traffic to your web application, prevent malicious data from leaking out of your app, and allow legitimate traffic. This is accomplished by following a set of policies that assist identify which traffic is malicious and which is not.

Test Items

Because the web application firewall is centred on avoiding harmful attacks on web application systems. Incoming HTTP messages are examined by the WAF, which determines whether they should be blocked or sent to the web application. The choice is made based on a set of heuristics for detecting attack patterns. This inhibits assaults on the application layer that would normally get past standard network firewalls, such as:

- Cross-site scripting (XSS)
- Structured Query Language (SQL) injection
- Web session hacking
- Distributed denial-of-service (DDoS) attacks

Based on the requirement specification document the functionalities that we need to test are as following:

- **Detect malicious traffic**:

The system will be able to detect malicious requests from the network traffic while keeping a log and allowing the normal flow of traffic.

- **Prevent malicious traffic:**

The system will prevent and log any malicious traffic while allowing the normal flow of traffic.

- **Changing logs to data frame:**

This project uses python scripts to make a Machine Learning Model which can take .json or .csv

And convert it into data frame which can then be used as input to ML model

- **Evaluating the core rule sets:**

The ML model will evaluate core rule sets based on network traffic. If there are more than acceptable false-positives or false-negatives, then the ML model will find a better collection or arrangement for the core rule sets to follow.

- **Updating the core rule sets:**

After finding a good collection the ML model will make changes to mod Security custom rule set file allow the new arrangement to be implemented.

5.2 Features to be Tested

1. Extracting features from the log files.
2. Distinction between malicious and benign traffic.

3. Using the extracted features for training our ML model.
4. ML model directing the CRS to be used based on them.

5.3 Detailed Test Strategy

This project is a code intensive system hence there are different modules for different functionalities. This kind of program can be tested by using White-box and block-box testing techniques as it provides output for each module separately with desired and random inputs.

- **White box testing:**

White box testing is used when a software is to be tested on the code level and functions and their results are according to the requirements we put forth. This kind of testing will focus on one function at a time and ignore the rest to get testing done on a deeper level as compared to just running the whole system.

- **Black Box testing:**

Black box testing typically iterates over every possible input and uses the software to ensure that it leads to the correct output as the end user does.

5.4 Item Pass/Fail Criteria

As client request is sent towards the firewall, firewall filters all types of requests whether these request are malicious or not, filtering of request will be done on the basis of core rule sets. Machine learning model train the core rule sets that the firewall automatically accepts or reject the request. Client request was present in the log. If the request is accepted, client will access the website.

Details of the test cases can be found in the Test Deliverables section. The principle outlined below is when a test item is judged to pass or fail.

- Prerequisites are met
- Input runs as specified
- If the result works as specified in the output, this means that the module will pass.
- If the system does not work or does not meet the output specifications, it is a module failure.

5.4.1 Test Deliverables

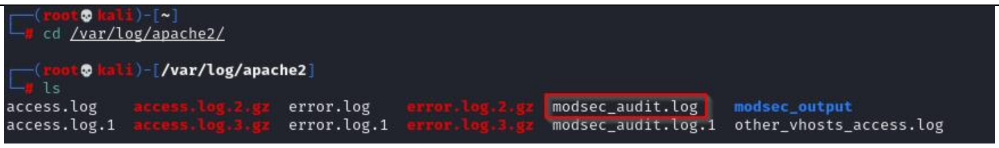
Following are the test cases:

Authentication Failed/Registration Failed User enters an email address in an invalid format. Corresponding output: An error message is displayed the actual output has been confirmed.

Test Case	WAF
Test Case Number	1
Description	This Section checks if the installed Web application fire wall works fine and there is no problem in its default working.
Testing technique used	White box testing
Preconditions	The Linux environment must contain apache2 and MySQL database
Input	Malicious traffic
Steps	Starting apache2 and MySQL. Configure SecRuleEngine from DetectionOnly to on in modsecurity.conf file. Perform general attacks to check if it protects the web app.
Expected output	Waf protects the web app from simple attacks. Display a forbidden message when an attack is detected.
Alternative Path	Reinstall or reconfigure Cause: Installation or configuration issues.

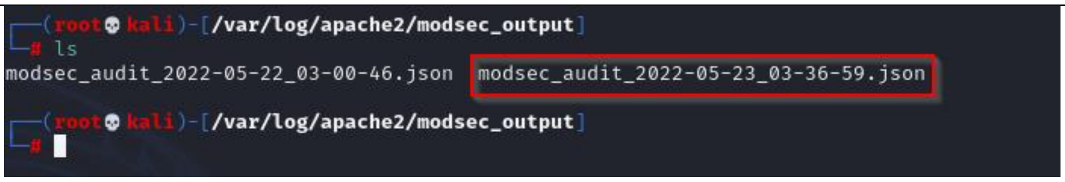
Actual output	Waf protects the web app from simple attacks. Display a forbidden message when an attack is detected.

Test Case	Log file creation
Test Case Number	2
Description	This test is to make sure that the log files are created properly.
Testing technique used	Whitebox testing
Preconditions	Case no.1 has a pass (WAD is installed properly)
Input	None from user side (Log are created automatically)
Steps	In modsecurity.conf file find find audit log configuration and change SecAuditEngine from RelevantOnly to on for logging of all the traffic. SecAuditLog implies the location of the file.
Expected output	Log file is created at the specified directory.
Alternative Path	Reconfiguration Cause: Configuration issues in modsecurity.conf file or directory missing.

Actual output	

```
┌──(root💀kali)-[~]
└─# cd /var/log/apache2/

┌──(root💀kali)-[/var/log/apache2]
└─# ls
access.log    access.log.2.gz  error.log    error.log.2.gz  modsec_audit.log    modsec_output
access.log.1  access.log.3.gz  error.log.1  error.log.3.gz  modsec_audit.log.1  other_vhosts_access.log
```

Test Case	Feature extraction
Test Case Number	3
Description	Python script to convert the logs into a useable information or Feature extraction by translating .log file into a .json or .csv file
Testing technique used	WhiteBox Testing
Preconditions	Case 2 is working and has a pass.
Input	Location of the modsec.audit.log file
Steps	Run the python Script with -f tag and location of the log file after
Expected output	The output file will be created in the same directory as the log file
Alternative Path	Make log file again or add a different one Cause: Log file is missing or corrupted.

Actual output	

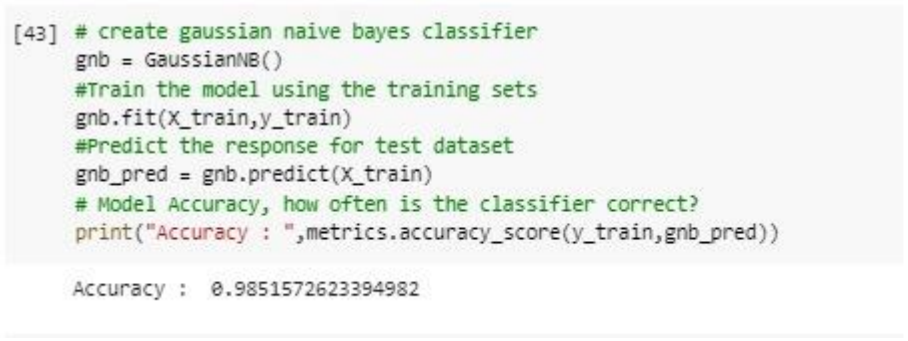
```
┌──(root💀kali)-[/var/log/apache2/modsec_output]
└─# ls
modsec_audit_2022-05-22_03-00-46.json  modsec_audit_2022-05-23_03-36-59.json

┌──(root💀kali)-[/var/log/apache2/modsec_output]
└─# 
```

Test Case	ML model (Dataframe conversion module)
Test Case Number	4
Description	This part of the ML model will take the in the extracted features in .json or .csv files and convert them in dataframe which are to be used for as input for our machine learning process.
Testing technique used	White Box Testing
Preconditions	Case 3 has a pass.
Input	.json or .csv file from case 3
Steps	Use python script to target the converted file with feature. Extract the needed information to be used in the form of a dataframe.
Expected output	Needed information is extracted successfully

Alternative Path	Reevaluation Cause: information not extracted, or file is corrupted
Actual output	Needed information is extracted successfully

Test Case	ML model (Learning and testing)
Test Case Number	5
Description	The ML model needs to learn the difference between good and bad network traffic which be done by providing it with the required data extracted up till case 4.
Testing technique used	Black Box Testing
Preconditions	Case 4 has a pass
Input	Dataframe
Steps	The ML model takes the dataframe as input learns from it and applies it on a test traffic.
Expected output	Good efficiency in the test results

Alternative Path	Reevaluation Cause: More than acceptable false positives and negatives

Actual output	
	`[43] # create gaussian naive bayes classifier`
	`gnb = GaussianNB()`
	`#Train the model using the training sets`
	`gnb.fit(X_train,y_train)`
	`#Predict the response for test dataset`
	`gnb_pred = gnb.predict(X_train)`
	`# Model Accuracy, how often is the classifier correct?`
	`print("Accuracy : ",metrics.accuracy_score(y_train,gnb_pred))`
	
	`Accuracy :  0.9851572623394982`

Test Case	Custom Core Rule sets
Test Case Number	6
Description	The last case left to be tested is the ability of this python script to write custom care rule sets in case it evaluates that WAF was not either strict enough or flexible enough
Testing technique used	Black Box Testing
Preconditions	Case 5 has a pass
Input	Analyst reports false categorization of the network traffic.

Steps	Once the analyst inputs the log file that contains the information on packets that were falsely categorized the ML model then checks the significance of that traffic and makes changes accordingly if needed
Expected output	Keeps the strictness of WAF at just the right amount as to not hinder the normal traffic or let the malicious traffic pass
Alternative Path	Reevaluation Cause: More than acceptable false positives and negatives
Actual output	Keeps the strictness of WAF at just the right amount as to not hinder the normal traffic or let the malicious traffic pass.

6. Machine Learning Model

6.1 Feature Engineering

Performing all feature engineering. The different steps are:

6.1.1 Step 1: Import dependencies

<u>**Libraries**</u>

- **import pandas as pd**

Pandas are primarily used for data analysis and related operations on tabular data in data frames.

- **import NumPy as np**

Python library called NumPy is used for working with arrays**.**

- **import pickle**

Pickle is a Python library that allows you to serialize and de-serialize Python object structures, often known as marshalling or flattening. Serialization is the process of transforming a memory item into a byte stream that may be saved on disk or communicated over the internet.

- **import matplotlib , pyplot as plt**

Matplotlib is a Python package that allows you to create static, animated, and interactive visualizations.

- **import seaborn**

Seaborn is a Python module for creating statistical visuals. It is based on matplotlib and tightly interacts with pandas data structures. Seaborn assists us in exploring and comprehending our data.

- **import string**

Using the same implementation as the built-in format () function, we may construct and tweak our own string formatting behavior in the string module.

- **from IPython.display import display**

In all frontends, show a Python object. All representations are calculated and supplied to the frontends by default. Frontends have control over which representations are utilised and how they are used. This is analogous to using print() in terminal IPython; for usage with richer frontends, see Jupyter notebook examples with extensive display logic.

- **from sklearn.feature_extraction.text import CountVectorizer**

The scikit-learn module in Python provides CountVectorizer, a useful utility for converting a given text into a vector based on the frequency (count) of each word that appears in the text.

- **from sklearn.feature_extraction.text import TfidfVectorizer**

TfidfVectorizer - Converts text into feature vectors that may be used as estimator input.

- **from sklearn, model_selection import train_test_split**

training and testing data will be splited.

- **from sklearn, model_selection import RandomizedSearchCV**

RandomizedSearchCV has two methods: "fit" and "score." If the estimator supports them, it also implements "score samples," "predict," "predict proba," "decision function," "transform," and "inverse transform."

- **from sklearn, model_selection import learning curve**

A quantifiable job, such as a factory worker learning to operate a new equipment that requires particular, repeated actions, is an example of where a learning curve might be used. The worker grows faster and more efficient as he learns to operate the equipment by following the procedural procedures.

- **from sklearn, decomposition import Truncated SVD**

For dimensional reduction, truncated SVD (also known as LSA) is utilised. This transformer uses truncated singular value decomposition to reduce linear dimensionality (SVD). This estimator, unlike PCA, does not centre the data prior to generating the singular value decomposition.

Confusion matrix:

A confusion matrix C is defined as the number of observations known to be in group I and predicted to be in group j, where Ci,j is the number of observations known to be in group I and projected to be in group j.

Genuine negatives have a count of C0,0, false negatives have a count of C1,0, true positives have a count of C1,1 and false positives have a count of C0,1.

ROC curve:

In binary classification, ROC curves are used to investigate a classifier's output. To apply the ROC curve and ROC area to multi-label classification, the result must be binarized.

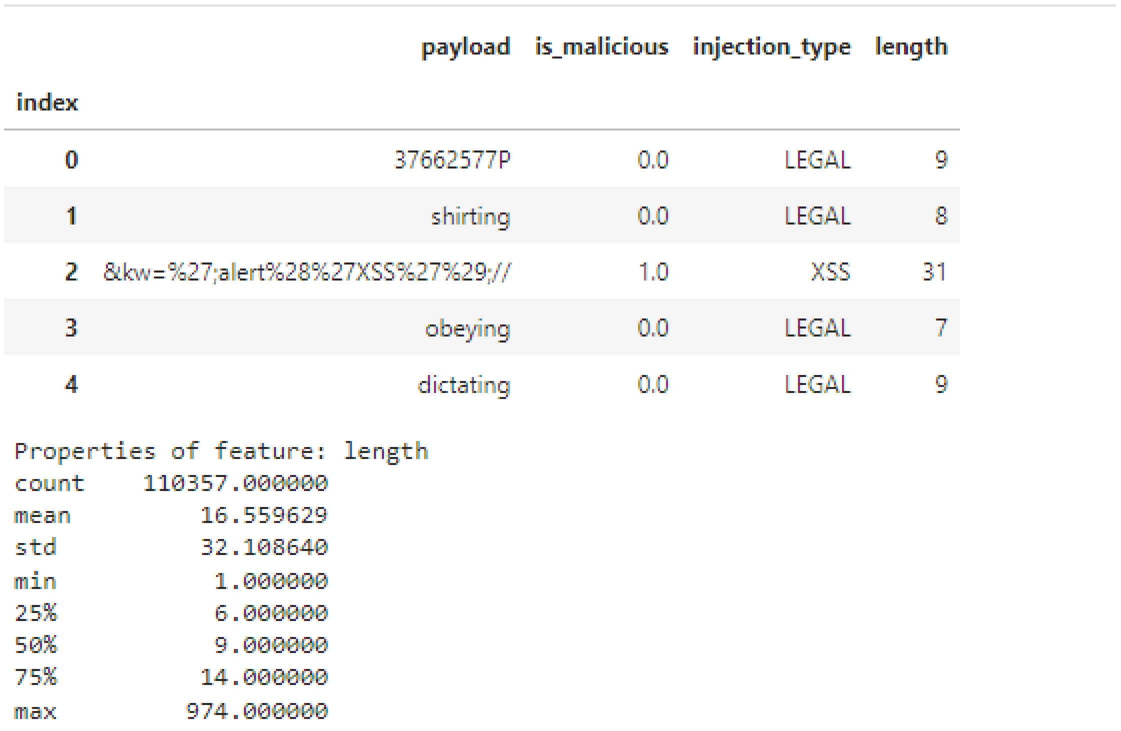
6.1.2 Step 2: Feature engineering custom features

- **length of payload**

Code

```
def create_feature_length(payloads):
  """
  function to create length feature
  """
  payloads['length'] = [len(str(row)) for row in payloads['payload']]
  return payloads

payloads = create_feature_length(payloads)
display(payloads.head())
#plot graph
plot_feature_distribution(payloads['length'])
```

Output



index	payload	is_malicious	injection_type	length
0	37662577P	0.0	LEGAL	9
1	shirting	0.0	LEGAL	8
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31
3	obeying	0.0	LEGAL	7
4	dictating	0.0	LEGAL	9

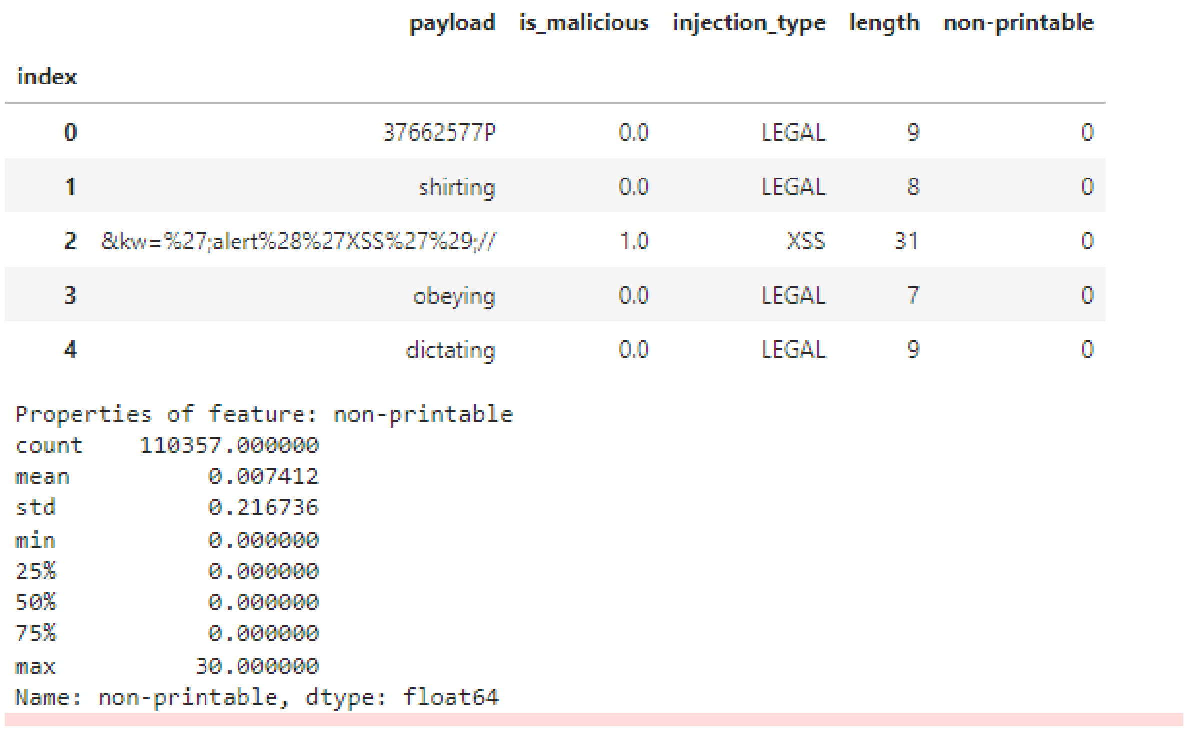
```
Properties of feature: length
count    110357.000000
mean         16.559629
std          32.108640
min           1.000000
25%           6.000000
50%           9.000000
75%          14.000000
max         974.000000
```

➢ **number of non-printable characters in payload**

Code

```
def create_feature_nonprintable(payloads):
  """
   function to find non- printable char in payload if present -->1 if absent --> 0
  """
  payloads['non-printable'] = [len([1 for letter in str(row) if letter not in string.printable])
for row in payloads['payload']]
  return payloads

payloads = create_feature_nonprintable(payloads)
display(payloads.head())

plot_feature_distribution(payloads['non-printable'])
```

Output



	payload	is_malicious	injection_type	length	non-printable
index					
0	37662577P	0.0	LEGAL	9	0
1	shirting	0.0	LEGAL	8	0
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31	0
3	obeying	0.0	LEGAL	7	0
4	dictating	0.0	LEGAL	9	0

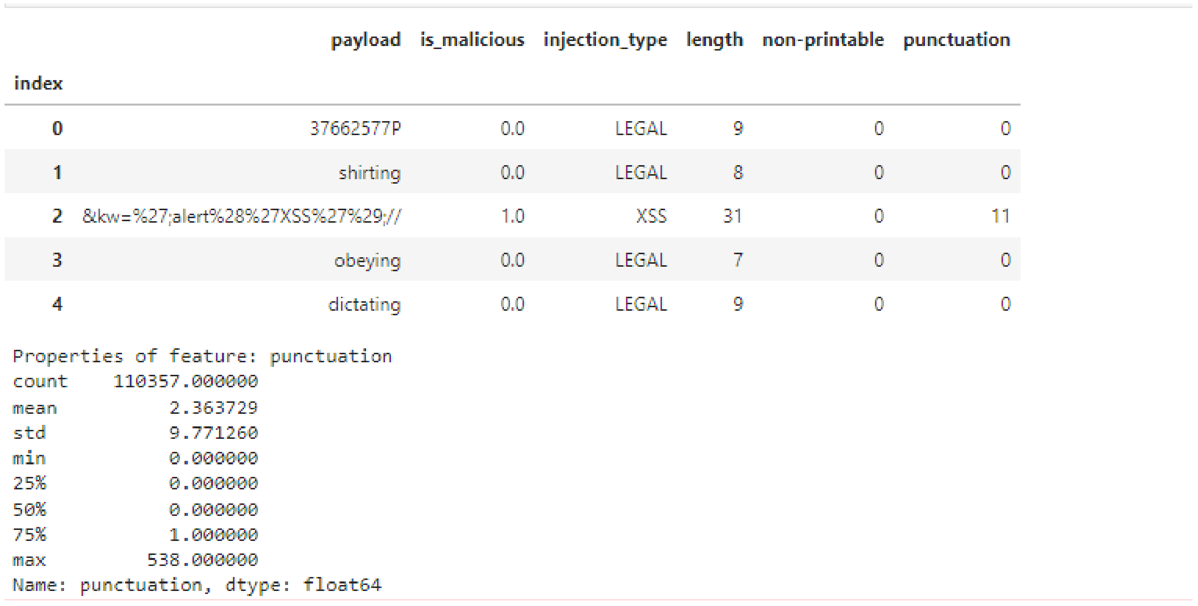
```
Properties of feature: non-printable
count    110357.000000
mean          0.007412
std           0.216736
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max          30.000000
Name: non-printable, dtype: float64
```

- **number of punctuation characters in payload**

Code

```
def create_feature_punchualchar(payloads):
  #adds new punctual cloumns with count of punctual characters inpayload
 payloads['punctuation'] = [len([1 for letter in str(row) if letter in string.punctuation])  for
row in payloads['payload']]
  return payloads

payloads = create_feature_punchualchar(payloads)
display(payloads.head())

plot_feature_distribution(payloads['punctuation'])
```

Output



index	payload	is_malicious	injection_type	length	non-printable	punctuation
0	37662577P	0.0	LEGAL	9	0	0
1	shirting	0.0	LEGAL	8	0	0
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31	0	11
3	obeying	0.0	LEGAL	7	0	0
4	dictating	0.0	LEGAL	9	0	0

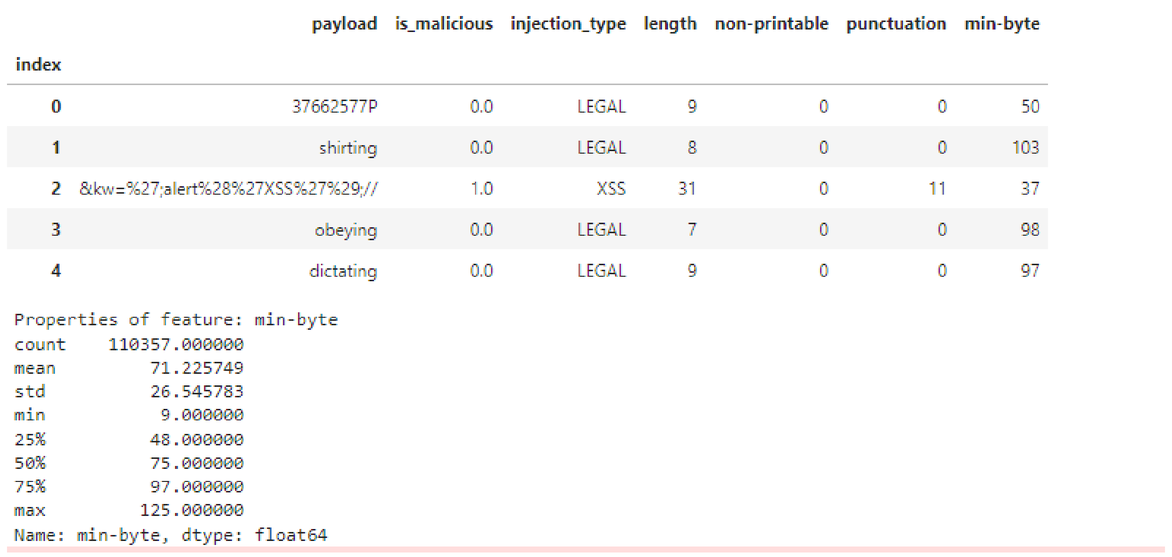
```
Properties of feature: punctuation
count    110357.000000
mean          2.363729
std           9.771260
min           0.000000
25%           0.000000
50%           0.000000
75%           1.000000
max         538.000000
Name: punctuation, dtype: float64
```

➢ **the minimum byte value of payload**

<u>Code</u>

```
def create_feature_minbyte(payloads):
  payloads['min-byte'] = [min(bytearray(str(row),'utf-8')) for row in payloads['payload']]
  return payloads

payloads = create_feature_minbyte(payloads)
display(payloads.head())
plot_feature_distribution(payloads['min-byte'])
```

<u>Output</u>



	payload	is_malicious	injection_type	length	non-printable	punctuation	min-byte
index							
0	37662577P	0.0	LEGAL	9	0	0	50
1	shirting	0.0	LEGAL	8	0	0	103
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31	0	11	37
3	obeying	0.0	LEGAL	7	0	0	98
4	dictating	0.0	LEGAL	9	0	0	97

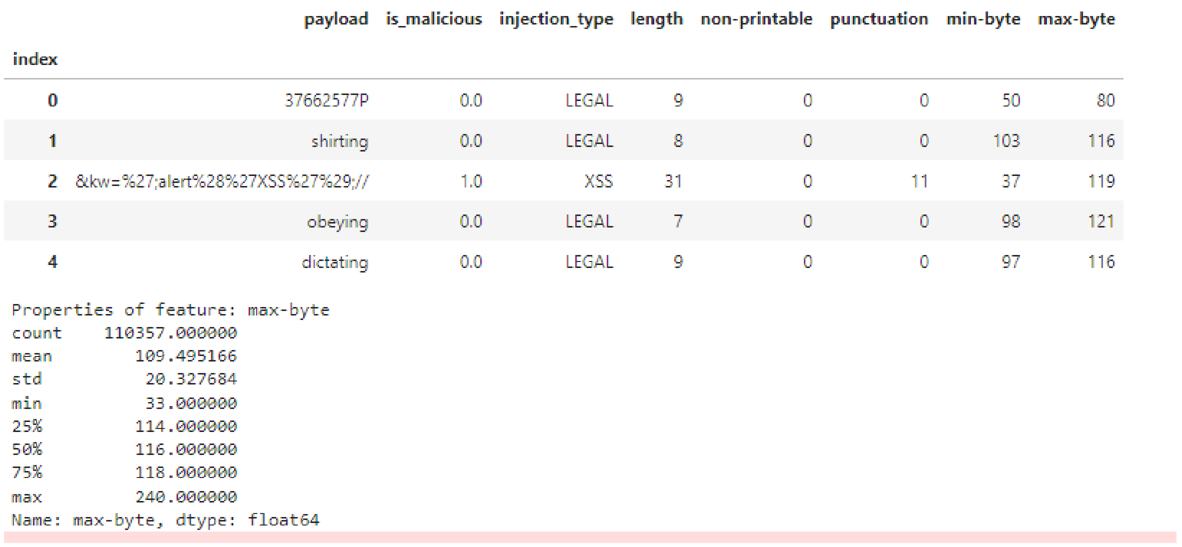
```
Properties of feature: min-byte
count    110357.000000
mean         71.225749
std          26.545783
min           9.000000
25%          48.000000
50%          75.000000
75%          97.000000
max         125.000000
Name: min-byte, dtype: float64
```

➢ **the maximum byte value of payload**

Code

```
#max byte feature# maximum byte value of payload
def create_feature_minbyte(payloads):
  payloads['max-byte'] = [max(bytearray(str(row),'utf-8')) for row in payloads['payload']]
  return payloads

payloads = create_feature_minbyte(payloads)
display(payloads.head())
plot_feature_distribution(payloads['max-byte'])
```

Output



index	payload	is_malicious	injection_type	length	non-printable	punctuation	min-byte	max-byte
0	37662577P	0.0	LEGAL	9	0	0	50	80
1	shirting	0.0	LEGAL	8	0	0	103	116
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31	0	11	37	119
3	obeying	0.0	LEGAL	7	0	0	98	121
4	dictating	0.0	LEGAL	9	0	0	97	116

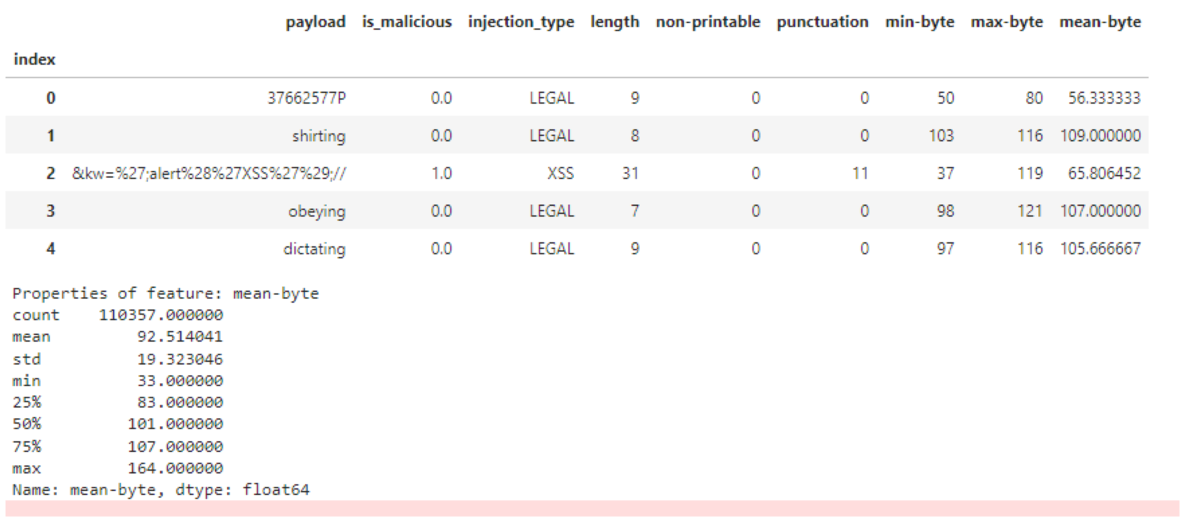
```
Properties of feature: max-byte
count    110357.000000
mean        109.495166
std          20.327684
min          33.000000
25%         114.000000
50%         116.000000
75%         118.000000
max         240.000000
Name: max-byte, dtype: float64
```

➢ the mean byte value of payload

Code

```
#mean byte value of payload
def create_feature_minbyte(payloads):
  payloads['mean-byte'] = [np.mean(bytearray(str(row),'utf-8')) for row in
payloads['payload']]
  return payloads

payloads = create_feature_minbyte(payloads)
display(payloads.head())
plot_feature_distribution(payloads['mean-byte'].astype(int))
```

Output



index	payload	is_malicious	injection_type	length	non-printable	punctuation	min-byte	max-byte	mean-byte
0	37662577P	0.0	LEGAL	9	0	0	50	80	56.333333
1	shirting	0.0	LEGAL	8	0	0	103	116	109.000000
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31	0	11	37	119	65.806452
3	obeying	0.0	LEGAL	7	0	0	98	121	107.000000
4	dictating	0.0	LEGAL	9	0	0	97	116	105.666667

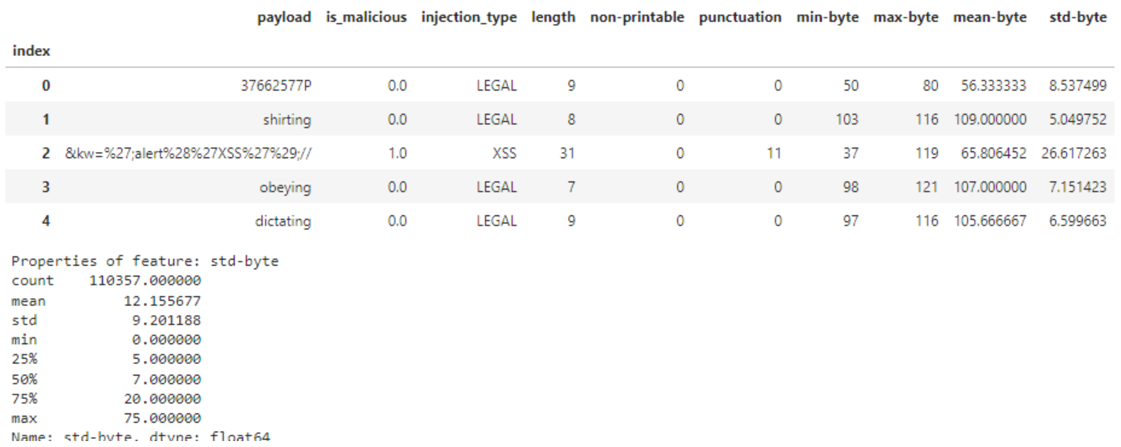
```
Properties of feature: mean-byte
count    110357.000000
mean         92.514041
std          19.323046
min          33.000000
25%          83.000000
50%         101.000000
75%         107.000000
max         164.000000
Name: mean-byte, dtype: float64
```

➢ **the standard deviation of payload byte values**

Code

```
#standard deviation of byte value of payload
def create_feature_minbyte(payloads):
  payloads['std-byte'] = [np.std(bytearray(str(row),'utf-8')) for row in payloads['payload']]
  return payloads

payloads = create_feature_minbyte(payloads)
display(payloads.head())
plot_feature_distribution(payloads['std-byte'].astype(int))
```

Output



index	payload	is_malicious	injection_type	length	non-printable	punctuation	min-byte	max-byte	mean-byte	std-byte
0	37662577P	0.0	LEGAL	9	0	0	50	80	56.333333	8.537499
1	shirting	0.0	LEGAL	8	0	0	103	116	109.000000	5.049752
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31	0	11	37	119	65.806452	26.617263
3	obeying	0.0	LEGAL	7	0	0	98	121	107.000000	7.151423
4	dictating	0.0	LEGAL	9	0	0	97	116	105.666667	6.599663

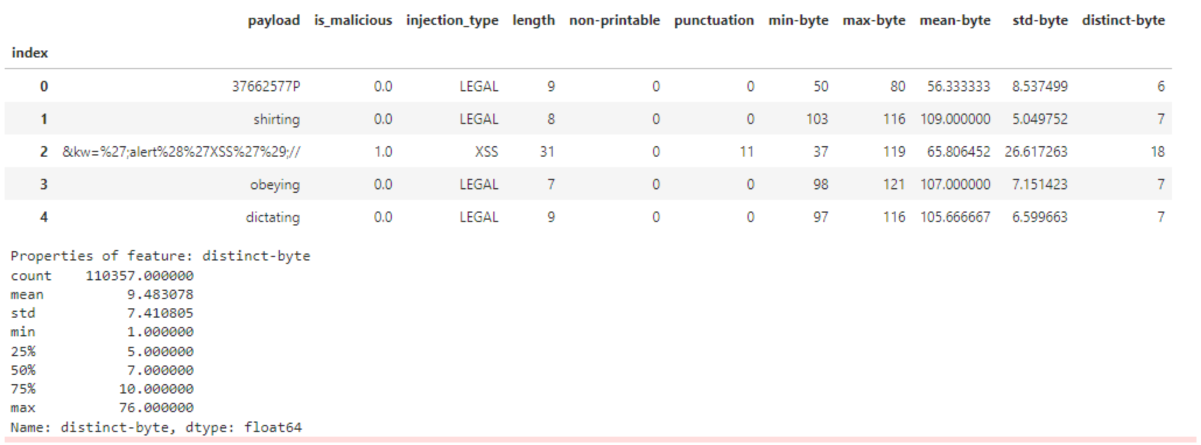
```
Properties of feature: std-byte
count    110357.000000
mean         12.155677
std           9.201188
min           0.000000
25%           5.000000
50%           7.000000
75%          20.000000
max          75.000000
Name: std-byte, dtype: float64
```

- **number of distinct bytes in payload**

Code

```
#distinct byte value of payload
def create_feature_minbyte(payloads):
payloads['distinct-byte'] = [len(set(bytearray(str(row),'utf-8'))) for row in
payloads['payload']]
  return payloads

payloads = create_feature_minbyte(payloads)
display(payloads.head())
plot_feature_distribution(payloads['distinct-byte'].astype(int))
```

Output



index	payload	is_malicious	injection_type	length	non-printable	punctuation	min-byte	max-byte	mean-byte	std-byte	distinct-byte
0	37662577P	0.0	LEGAL	9	0	0	50	80	56.333333	8.537499	6
1	shirting	0.0	LEGAL	8	0	0	103	116	109.000000	5.049752	7
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31	0	11	37	119	65.806452	26.617263	18
3	obeying	0.0	LEGAL	7	0	0	98	121	107.000000	7.151423	7
4	dictating	0.0	LEGAL	9	0	0	97	116	105.666667	6.599663	7

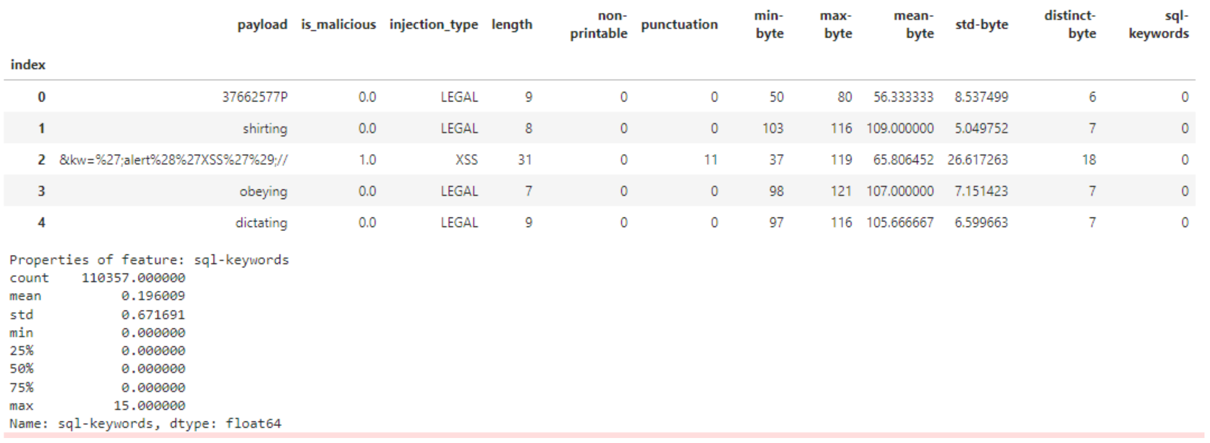
```
Properties of feature: distinct-byte
count    110357.000000
mean          9.483078
std           7.410805
min           1.000000
25%           5.000000
50%           7.000000
75%          10.000000
max          76.000000
Name: distinct-byte, dtype: float64
```

- **number of SQL keywords in payload**

Code

```
#number of SQL keywords in payload
sql_keywords = pd.read_csv('https://trello-attachments.s3.amazonaws.com/5ed2d4107c349c221194b608/5ed2d453f0e5a45bcd8cf16c/435e639346787ce2b495a16e9f690ef5/SQLKeywords.txt', index_col=False)

def create_feature_sql_keywords(payloads):

    '''
    Feature
    Number of SQL keywords within payload
    '''
    payloads['sql-keywords'] = [ len([1 for keyword in sql_keywords['Keyword'] if str(keyword).lower() in str(row).lower()]) for row in payloads['payload']]
    return payloads

create_feature_sql_keywords(payloads)
display(type(sql_keywords))
display(payloads.head())
plot_feature_distribution(payloads['sql-keywords'])
```

Output



index	payload	is_malicious	injection_type	length	non-printable	punctuation	min-byte	max-byte	mean-byte	std-byte	distinct-byte	sql-keywords
0	37662577P	0.0	LEGAL	9	0	0	50	80	56.333333	8.537499	6	0
1	shirting	0.0	LEGAL	8	0	0	103	116	109.000000	5.049752	7	0
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31	0	11	37	119	65.806452	26.617263	18	0
3	obeying	0.0	LEGAL	7	0	0	98	121	107.000000	7.151423	7	0
4	dictating	0.0	LEGAL	9	0	0	97	116	105.666667	6.599663	7	0

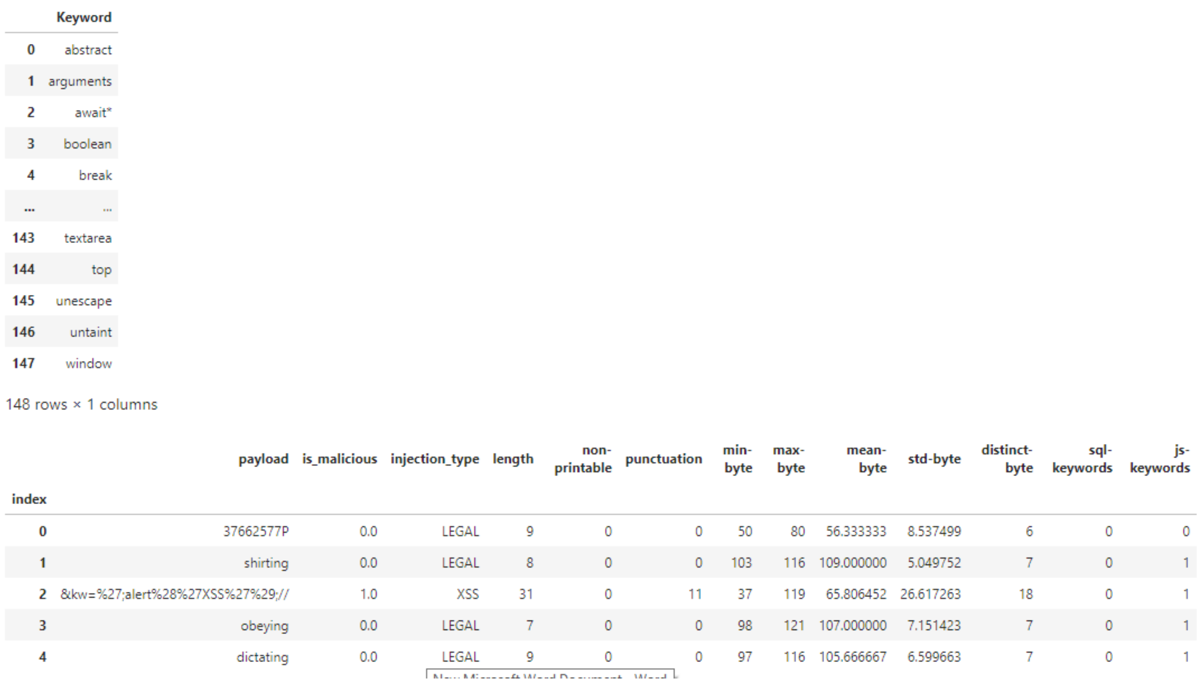
```
Properties of feature: sql-keywords
count    110357.000000
mean          0.196009
std           0.671691
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max          15.000000
Name: sql-keywords, dtype: float64
```

➢ **number of javascript keywords in payload**

Code

```
# javascript key word is presten or not in payload
js_keywords = pd.read_csv("https://trello-attachments.s3.amazonaws.com/5ed2d4107c349c221194b608/5ed2d453f0e5a45bcd8cf16c/dedc7eb9846a30c252cd950a0e2153d9/JavascriptKeywords.txt",index_col=False)
def create_feature_javascript_keywords(payloads):
  payloads['js-keywords'] = [len([1 for keyword in js_keywords['Keyword'] if str(keyword).lower() in str(row).lower()]) for row in payloads['payload']]
  return payloads
display(js_keywords)
payloads = create_feature_javascript_keywords(payloads)
display(payloads.head())
plot_feature_distribution(payloads['js-keywords'])
```

Output



	Keyword
0	abstract
1	arguments
2	await*
3	boolean
4	break
...	...
143	textarea
144	top
145	unescape
146	untaint
147	window

148 rows × 1 columns

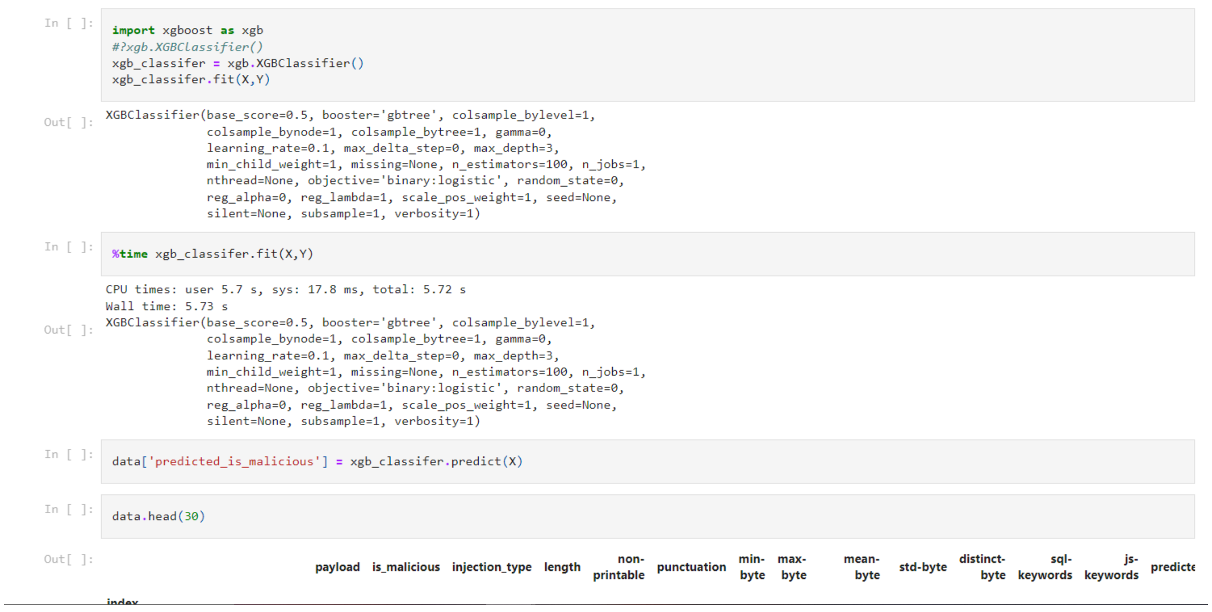
index	payload	is_malicious	injection_type	length	non-printable	punctuation	min-byte	max-byte	mean-byte	std-byte	distinct-byte	sql-keywords	js-keywords
0	37662577P	0.0	LEGAL	9	0	0	50	80	56.333333	8.537499	6	0	0
1	shirting	0.0	LEGAL	8	0	0	103	116	109.000000	5.049752	7	0	1
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31	0	11	37	119	65.806452	26.617263	18	0	1
3	obeying	0.0	LEGAL	7	0	0	98	121	107.000000	7.151423	7	0	1
4	dictating	0.0	LEGAL	9	0	0	97	116	105.666667	6.599663	7	0	1

6.2 XGB booster

Extreme Gradient Boosting is abbreviated as XGBoost. It's a gradient-boosted decision tree (GBDT) machine learning package that's scalable and distributed. It is the top machine learning package for regression, classification, and ranking tasks, and it includes parallel tree boosting.

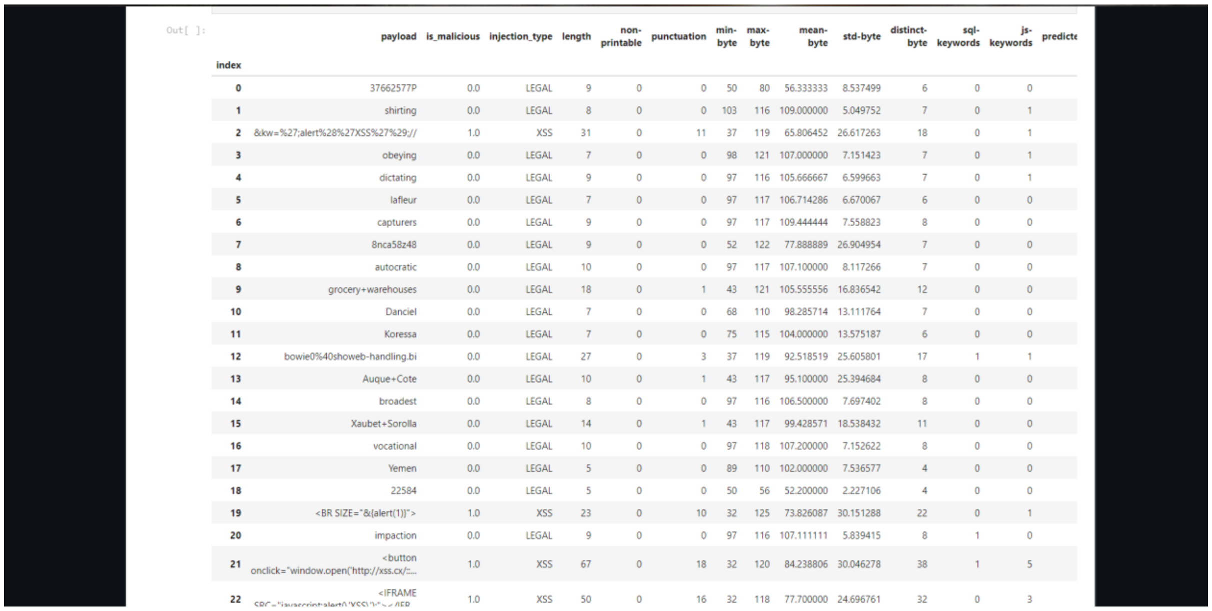
Gradient boosting refers to the process of "boosting" or enhancing a single weak model by merging it with numerous other weak models to get a collectively strong model. The method of additively producing weak models is formulated as a gradient descent algorithm over an objective function in this application of boosting. To reduce mistakes, gradient boosting establishes intended outcomes for the following model. The gradient of the mistake with regard to the prediction determines the targeted outcomes for each case.

The model I used allows for multiple results of packet characteristics to be examined.



```
In [ ]: import xgboost as xgb
        #?xgb.XGBClassifier()
        xgb_classifer = xgb.XGBClassifier()
        xgb_classifer.fit(X,Y)

Out[ ]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=1, gamma=0,
                      learning_rate=0.1, max_delta_step=0, max_depth=3,
                      min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                      nthread=None, objective='binary:logistic', random_state=0,
                      reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                      silent=None, subsample=1, verbosity=1)

In [ ]: %time xgb_classifer.fit(X,Y)

        CPU times: user 5.7 s, sys: 17.8 ms, total: 5.72 s
        Wall time: 5.73 s
Out[ ]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=1, gamma=0,
                      learning_rate=0.1, max_delta_step=0, max_depth=3,
                      min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                      nthread=None, objective='binary:logistic', random_state=0,
                      reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                      silent=None, subsample=1, verbosity=1)

In [ ]: data['predicted_is_malicious'] = xgb_classifer.predict(X)

In [ ]: data.head(30)

Out[ ]:
```

	payload	is_malicious	injection_type	length	non-printable	punctuation	min-byte	max-byte	mean-byte	std-byte	distinct-byte	sql-keywords	js-keywords	predicte
index														

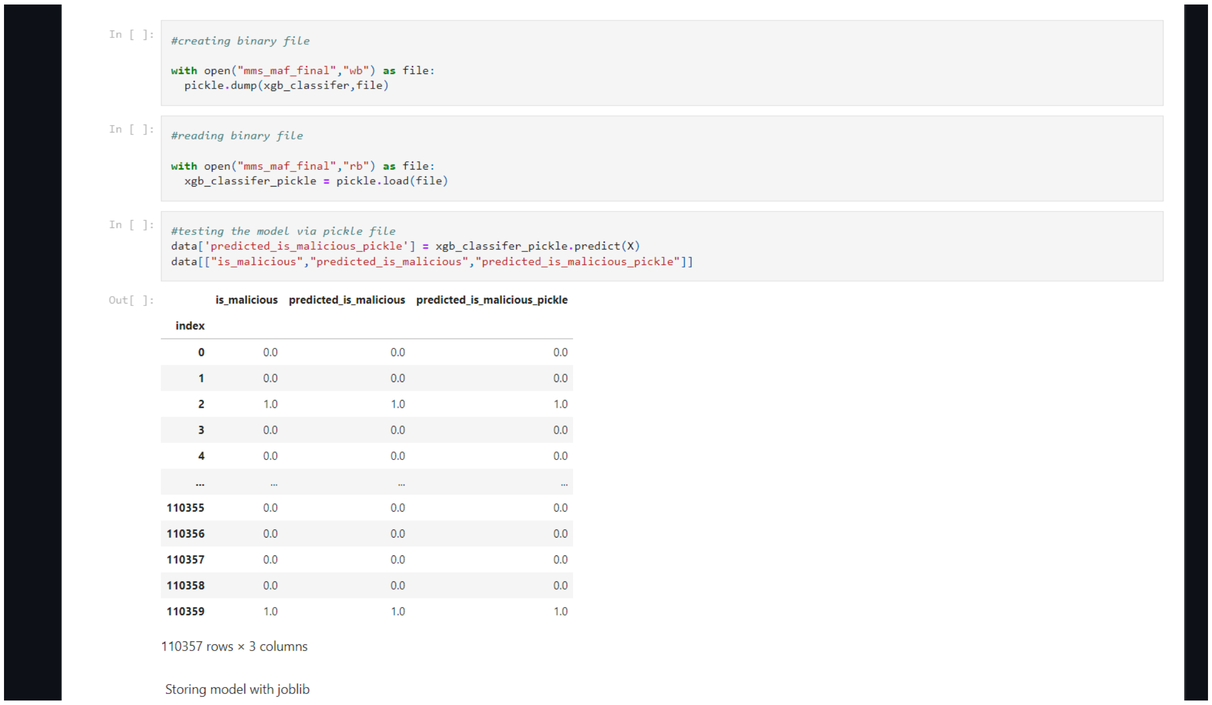


Out[ ]:

index	payload	is_malicious	injection_type	length	non-printable	punctuation	min-byte	max-byte	mean-byte	std-byte	distinct-byte	sql-keywords	js-keywords	predicte
0	37662577P	0.0	LEGAL	9	0	0	50	80	56.333333	8.537499	6	0	0	
1	shirting	0.0	LEGAL	8	0	0	103	116	109.000000	5.049752	7	0	1	
2	&kw=%27;alert%28%27XSS%27%29;//	1.0	XSS	31	0	11	37	119	65.806452	26.617263	18	0	1	
3	obeying	0.0	LEGAL	7	0	0	98	121	107.000000	7.151423	7	0	1	
4	dictating	0.0	LEGAL	9	0	0	97	116	105.666667	6.599663	7	0	1	
5	lafleur	0.0	LEGAL	7	0	0	97	117	106.714286	6.670067	6	0	0	
6	capturers	0.0	LEGAL	9	0	0	97	117	109.444444	7.558823	8	0	0	
7	8nca58z48	0.0	LEGAL	9	0	0	52	122	77.888889	26.904954	7	0	0	
8	autocratic	0.0	LEGAL	10	0	0	97	117	107.100000	8.117266	7	0	0	
9	grocery+warehouses	0.0	LEGAL	18	0	1	43	121	105.555556	16.836542	12	0	0	
10	Danciel	0.0	LEGAL	7	0	0	68	110	98.285714	13.111764	7	0	0	
11	Koressa	0.0	LEGAL	7	0	0	75	115	104.000000	13.575187	6	0	0	
12	bowie0%40showeb-handling.bi	0.0	LEGAL	27	0	3	37	119	92.518519	25.605801	17	1	1	
13	Auque+Cote	0.0	LEGAL	10	0	1	43	117	95.100000	25.394684	8	0	0	
14	broadest	0.0	LEGAL	8	0	0	97	116	106.500000	7.697402	8	0	0	
15	Xaubet+Sorolla	0.0	LEGAL	14	0	1	43	117	99.428571	18.538432	11	0	0	
16	vocational	0.0	LEGAL	10	0	0	97	118	107.200000	7.152622	8	0	0	
17	Yemen	0.0	LEGAL	5	0	0	89	110	102.000000	7.536577	4	0	0	
18	22584	0.0	LEGAL	5	0	0	50	56	52.200000	2.227106	4	0	0	
19	<BR SIZE="&{alert(1)}">	1.0	XSS	23	0	10	32	125	73.826087	30.151288	22	0	1	
20	impaction	0.0	LEGAL	9	0	0	97	116	107.111111	5.839415	8	1	0	
21	<button onclick="window.open('http://xss.cx/::...	1.0	XSS	67	0	18	32	120	84.238806	30.046278	38	1	5	
22	<IFRAME SRC="javascript:alert('XSS');"></IFR	1.0	XSS	50	0	16	32	118	77.700000	24.696761	32	0	3	

6.3 Pickle

For serialising and de-serializing a Python object structure, the Python pickle package is utilised. Pickling an object in Python allows it to be stored on a disc. Before writing to file, Pickle "serialises" the item. It's a method for converting a Python object (such a list or dict) into a character stream. This character stream provides all of the information required to recreate the object in a subsequent Python script.



```
In [ ]: #creating binary file

        with open("mms_maf_final","wb") as file:
          pickle.dump(xgb_classifer,file)
```

```
In [ ]: #reading binary file

        with open("mms_maf_final","rb") as file:
          xgb_classifer_pickle = pickle.load(file)
```

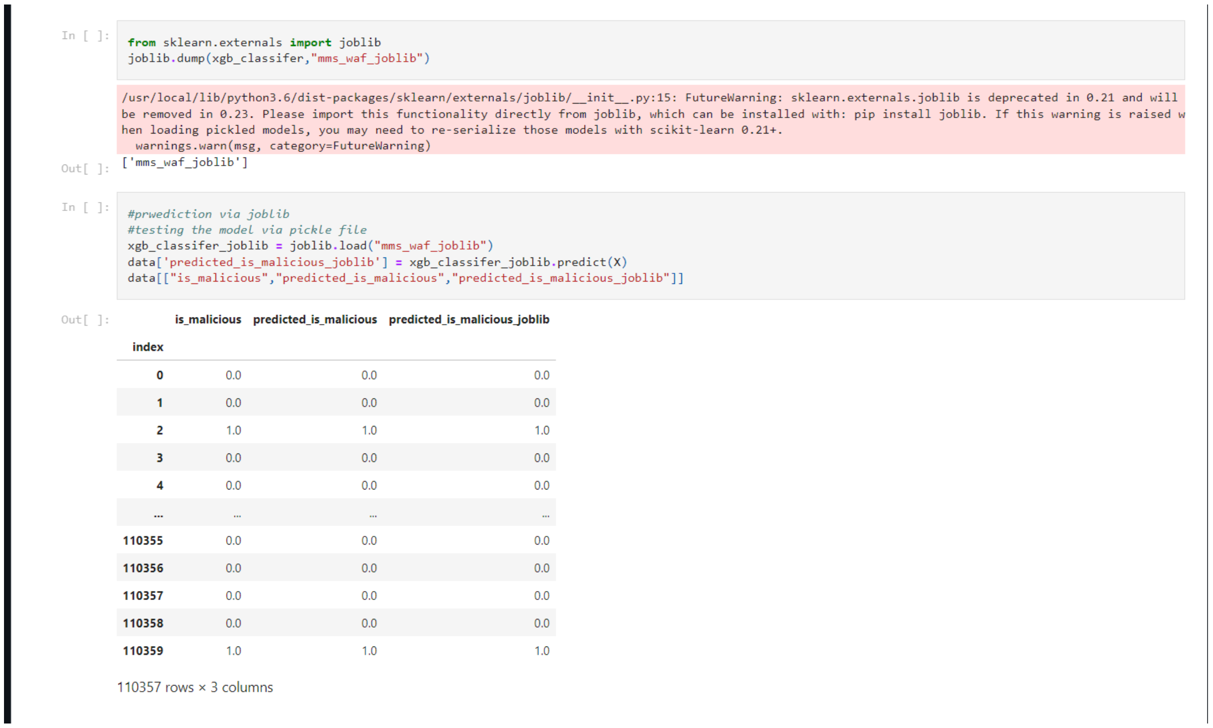
```
In [ ]: #testing the model via pickle file
        data['predicted_is_malicious_pickle'] = xgb_classifer_pickle.predict(X)
        data[["is_malicious","predicted_is_malicious","predicted_is_malicious_pickle"]]
```

Out[ ]:

index	is_malicious	predicted_is_malicious	predicted_is_malicious_pickle
0	0.0	0.0	0.0
1	0.0	0.0	0.0
2	1.0	1.0	1.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0
...	...	...	...
110355	0.0	0.0	0.0
110356	0.0	0.0	0.0
110357	0.0	0.0	0.0
110358	0.0	0.0	0.0
110359	1.0	1.0	1.0

110357 rows × 3 columns

Storing model with joblib



```
In [ ]:  from sklearn.externals import joblib
         joblib.dump(xgb_classifer,"mms_waf_joblib")
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/externals/joblib/__init__.py:15: FutureWarning: sklearn.externals.joblib is deprecated in 0.21 and will
be removed in 0.23. Please import this functionality directly from joblib, which can be installed with: pip install joblib. If this warning is raised w
hen loading pickled models, you may need to re-serialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=FutureWarning)
```

```
Out[ ]:  ['mms_waf_joblib']
```

```
In [ ]:  #prwediction via joblib
         #testing the model via pickle file
         xgb_classifer_joblib = joblib.load("mms_waf_joblib")
         data['predicted_is_malicious_joblib'] = xgb_classifer_joblib.predict(X)
         data[["is_malicious","predicted_is_malicious","predicted_is_malicious_joblib"]]
```

Out[ ]:

	is_malicious	predicted_is_malicious	predicted_is_malicious_joblib
index			
0	0.0	0.0	0.0
1	0.0	0.0	0.0
2	1.0	1.0	1.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0
...	...	...	...
110355	0.0	0.0	0.0
110356	0.0	0.0	0.0
110357	0.0	0.0	0.0
110358	0.0	0.0	0.0
110359	1.0	1.0	1.0

110357 rows × 3 columns

7. References and Work Cited

1. Clincy, Victor, and Hossain Shahriar. "Web application firewall: Network security models and configuration." *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. IEEE, 2018

2. Gupta, Namit, Abakash Saikia, and D. Sanghi. "Web application firewall." *Indian Institute of Technology, Kanpur* 61 (2007): 62.

3. N. Moustafa, B. Turnbull and K. R. Choo, "An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things," in *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815-4830, June 2019, doi: 10.1109/JIOT.2018.2871719.

4. Tekerek, Adem, Cemal Gemci, and Omer Faruk Bay. "Development of a hybrid web application firewall to prevent web based attacks." *2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT)*. IEEE, 2014.

5. Elmasri, B. (2015). *Detection of denial of service attacks on application layer protocols* (Order No. 10085278). Available from ProQuest Dissertations & Theses Global. (1780275442).

6. Mereani, F. A. (2021). *Investigating the detection of stored scripting attacks using machine learning*. (Unpublished Doctoral thesis, City, University of London)

7. Moosa, Asaad. "Artificial neural network based web application firewall for SQL injection." *International Journal of Computer and Information Engineering* 4.4 (2010): 610-619.

8. Tekerek, A. D. E. M., and O. F. Bay. "Design and implementation of an artificial intelligence-based web application firewall model." *Neural Network World* 29.4 (2019): 189-206.

9. Singh, Jatesh Jagraj, Hamman Samuel, and Pavol Zavarsky. "Impact of paranoia levels on the effectiveness of the modsecurity web application firewall." *2018 1st International Conference on Data Intelligence and Security (ICDIS)*. IEEE, 2018.

10. Pałka, Dariusz, and Marek Zachara. "Learning web application firewall-benefits and caveats." *International Conference on Availability, Reliability, and Security*. Springer, Berlin, Heidelberg, 2011.

11. Torrano-Gimenez, C., Perez-Villegas, A., Alvarez, G., Fernández-Medina, E., Malek, M., & Hernando, J. (2009, July). An Anomaly-based Web Application Firewall. In *SECRYPT* (pp. 23-28).

12. Muzaki, Rizki Agung, et al. "Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall." *2020 International Workshop on Big Data and Information Security (IWBIS)*. IEEE, 2020.

13. Hacker, Andrew J., and ISSAP CISSP. "Importance of web application firewall technology for protecting web-based resources." *ICSA Labs an Independent Verizon Business* (2008): 7.

14. Epp, Nico, et al. "Anomaly-based web application firewall using http-specific features and one-class svm." *Workshop Regional de Segurança da Informação e de Sistemas Computacionais*. 2017.

15. Appelt, Dennis, Cu D. Nguyen, and Lionel Briand. "Behind an application firewall, are we safe from SQL injection attacks?." *2015 IEEE 8th international conference on software testing, verification and validation (ICST)*. IEEE, 2015.

16. Akbar, M., & Ridha, M. A. F. (2018). Sql injection and cross site scripting prevention using owasp modsecurity web application firewall. *JOIV: International Journal on Informatics Visualization*, *2*(4), 286-292.

17. Mukhtar, B. I., & Azer, M. A. (2020, December). Evaluating the modsecurity web application firewall against SQL injection attacks. In *2020 15th International Conference on Computer Engineering and Systems (ICCES)* (pp. 1-6). IEEE.

18. Nagendran, K., Balaji, S., Raj, B. A., Chanthrika, P., & Amirthaa, R. G. (2020, March). Web application firewall evasion techniques. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)* (pp. 194-199). IEEE.

19. Shaheed, A., & Kurdy, M. H. D. (2022). Web Application Firewall Using Machine Learning and Features Engineering. Security and Communication Networks, 2022.

20. Yuan, H., Zheng, L., Dong, L., Peng, X., Zhuang, Y., & Deng, G. (2019, February). Research and implementation of WEB application firewall based on feature matching. In *International Conference on Application of Intelligent Systems in Multi-modal Information Analytics* (pp. 1223-1231). Springer, Cham.

21. Khamdamov, R. K., Kerimov, K. F., & ugli Ibrahimov, J. O. (2019). Method of Developing a Web-Application Firewall. *Journal of Automation and Information Sciences*, *51*(6).

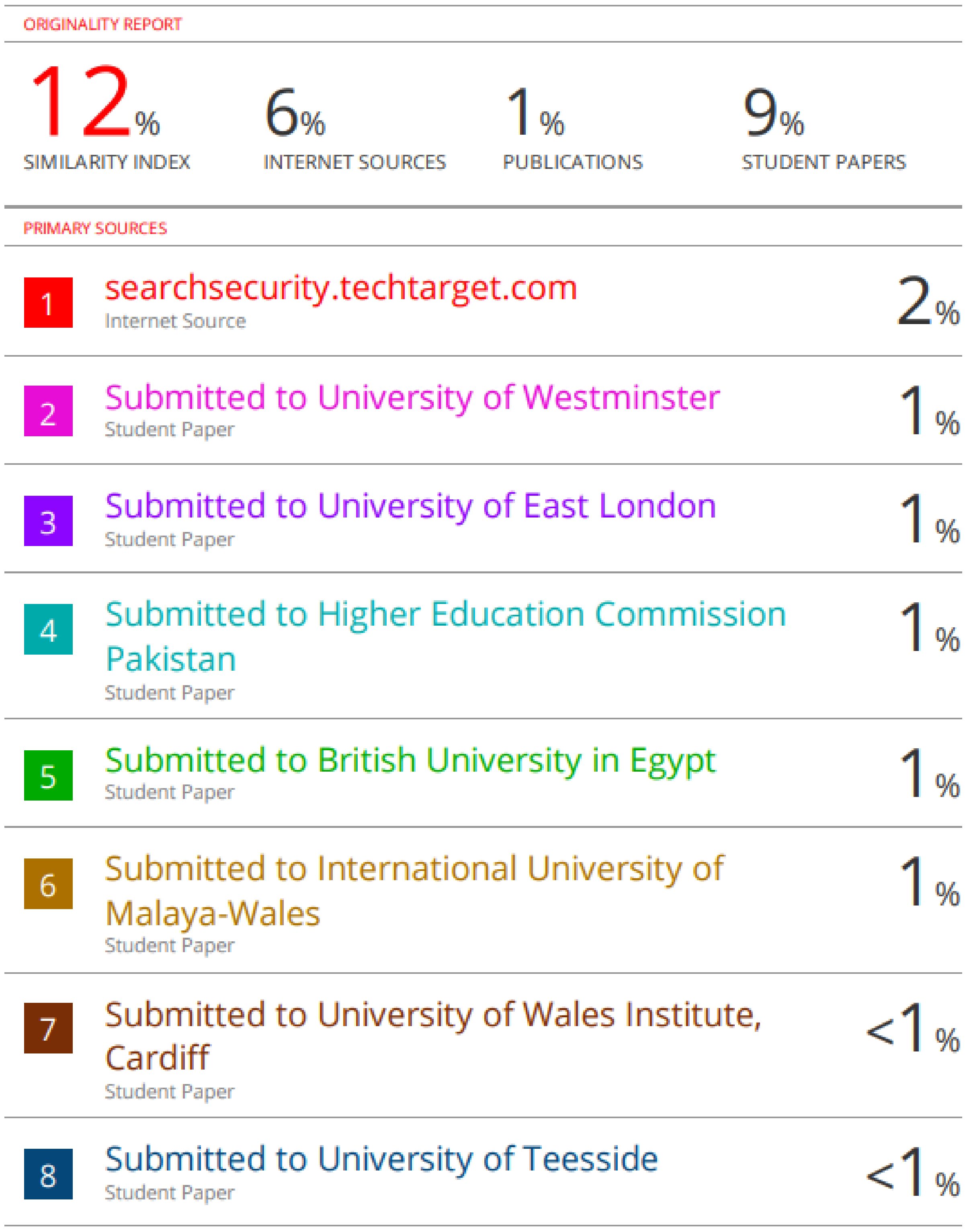
22. Thang, N. M. (2020). Improving efficiency of web application firewall to detect code injection attacks with random forest method and analysis attributes HTTP request. *Programming and Computer Software*, *46*(5), 351-361.

23. Manaseer, S., & Al Hwaitat, A. K. (2018). Centralized Web Application Firewall Security System. *Modern Applied Science*, *12*(10), 164.

24. Lee, H. Y., & Yang, H. S. (2017). Construction of Security Evaluation Criteria for Web Application Firewall. *Journal of digital Convergence*, *15*(5), 197-205.

25. Malekar, V., & Waghmare, J. M. (2013). Web application firewall to protect against web application vulnerabilities: A survey and comparison. *International Journal of Computer Technology and Applications*, *4*(1), 141.

8. Plagiarism Report

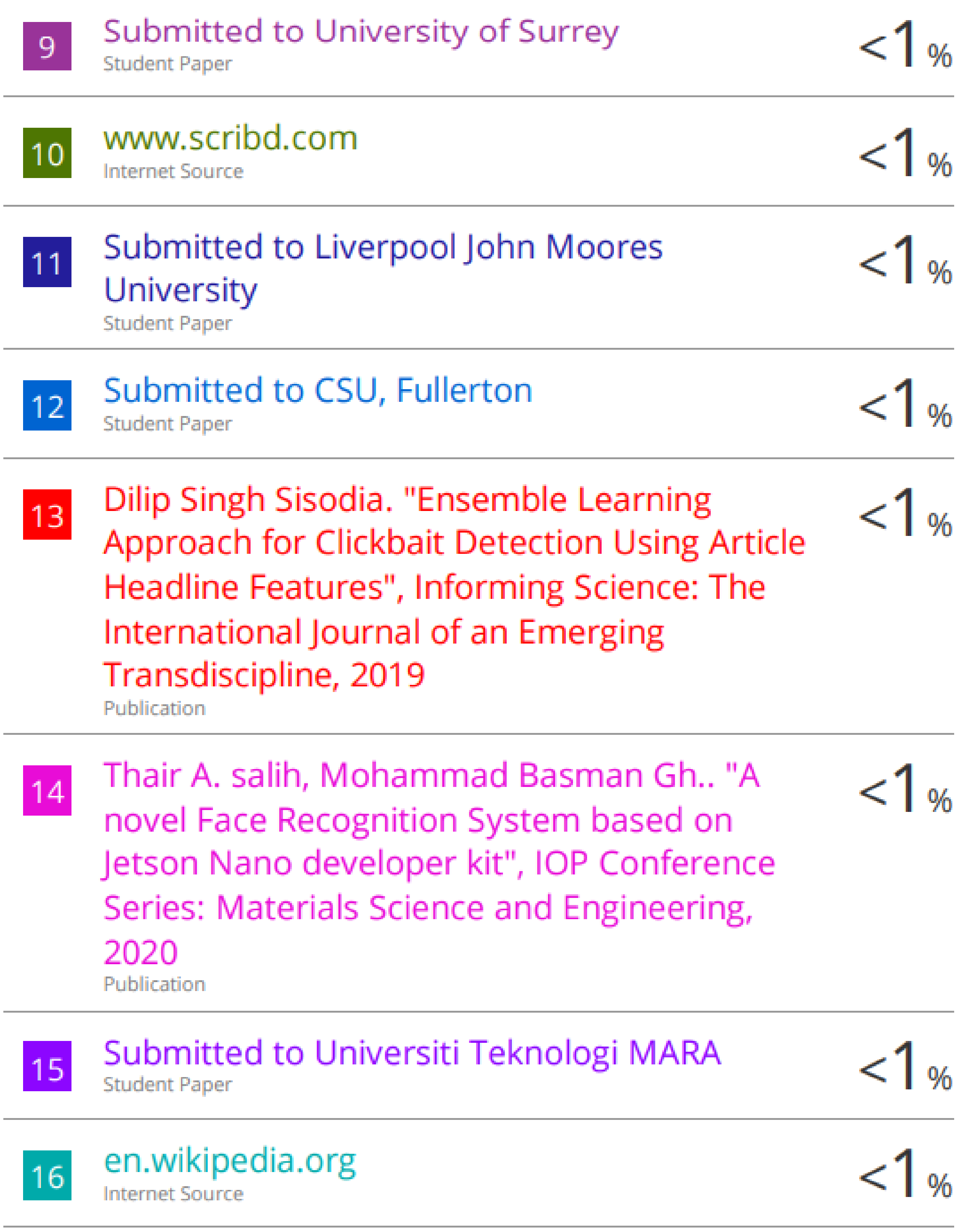


ORIGINALITY REPORT

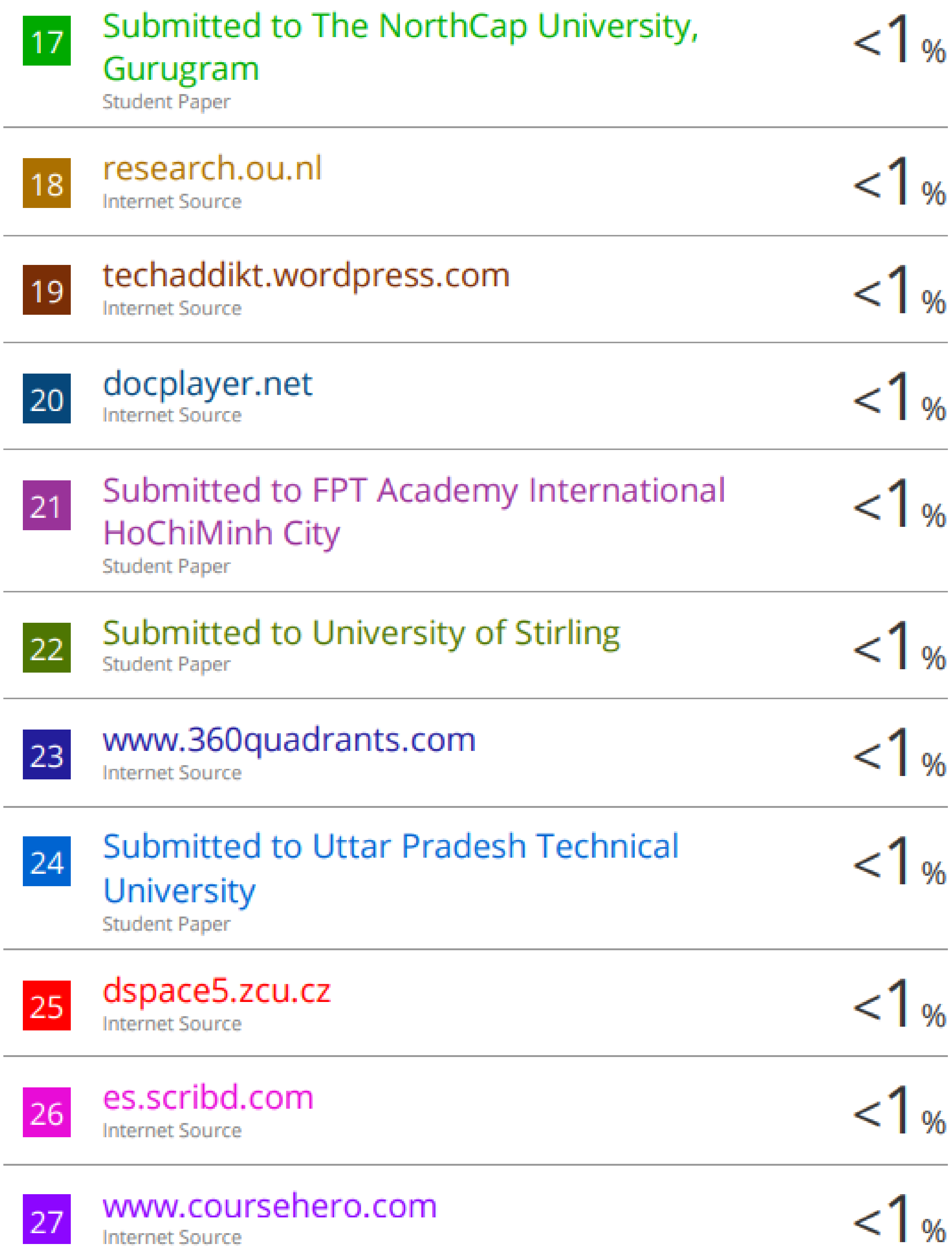
12%	6%	1%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	searchsecurity.techtarget.com Internet Source	2%
2	Submitted to University of Westminster Student Paper	1%
3	Submitted to University of East London Student Paper	1%
4	Submitted to Higher Education Commission Pakistan Student Paper	1%
5	Submitted to British University in Egypt Student Paper	1%
6	Submitted to International University of Malaya-Wales Student Paper	1%
7	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
8	Submitted to University of Teesside Student Paper	<1%



9	Submitted to University of Surrey Student Paper	<1 %
10	www.scribd.com Internet Source	<1 %
11	Submitted to Liverpool John Moores University Student Paper	<1 %
12	Submitted to CSU, Fullerton Student Paper	<1 %
13	Dilip Singh Sisodia. "Ensemble Learning Approach for Clickbait Detection Using Article Headline Features", Informing Science: The International Journal of an Emerging Transdiscipline, 2019 Publication	<1 %
14	Thair A. salih, Mohammad Basman Gh.. "A novel Face Recognition System based on Jetson Nano developer kit", IOP Conference Series: Materials Science and Engineering, 2020 Publication	<1 %
15	Submitted to Universiti Teknologi MARA Student Paper	<1 %
16	en.wikipedia.org Internet Source	<1 %



17	Submitted to The NorthCap University, Gurugram Student Paper	<1 %
18	research.ou.nl Internet Source	<1 %
19	techaddikt.wordpress.com Internet Source	<1 %
20	docplayer.net Internet Source	<1 %
21	Submitted to FPT Academy International HoChiMinh City Student Paper	<1 %
22	Submitted to University of Stirling Student Paper	<1 %
23	www.360quadrants.com Internet Source	<1 %
24	Submitted to Uttar Pradesh Technical University Student Paper	<1 %
25	dspace5.zcu.cz Internet Source	<1 %
26	es.scribd.com Internet Source	<1 %
27	www.coursehero.com Internet Source	<1 %

28 Dhagash Mehta, Dhruv Desai, Jithin Pradeep. "Machine learning fund categorizations", Proceedings of the First ACM International Conference on AI in Finance, 2020 <1%

Publication