Software Requirement Specification Document for Secure Interactive Applications

Jumana Nehad , Abdallah Awny,
Nour Ahmed , Rawan Hesham ,
Supervised by:Dr. Ayman Taha, Eng. Verina Saber

May 2, 2023

Table 1: Document version history

Version	Date	Reason for Change
1.0	2-December-2022	SRS First version's specifications are defined.
2.0	2-May-2022	SRS Second version's specifications are defined.

GitHub: `https://github.com/JumanaNehad/Secure-Interactive-Application`

Contents

Abstract

Nowadays, interactive applications are witnessing a significant development .These applications are popular targets for cyberattacks as they are network accessible and contain a large amount of user data. To ensure user privacy protection, its important to develop privacy protection solutions that utilize network traffic features and artificial intelligence . Distributed Denial of Service (DDoS) attacks are one of the most significant threats to network security, causing severe financial losses and reputational damage. This paper provides detailed information about security in interactive applications and proposes an artificial intelligent based security system for detecting Distributed Denial-of-Service (DDoS) attacks on a network

1 Introduction

1.1 Purpose of this document

The objective of this Software Requirements Specification document is to lay out the requirements for Secure Interactive Applications (SIA) that aim to detect DDOS attacks and block them. Moreover, this paper will be useful to future projects that will be involved in the project's development and maintenance.

1.2 Scope of this document

In this document, the system's features and purpose will be outlined, and detailed functional and non-functional requirements are provided in addition to the main functionalities of our system, which detects malicious attacks while using interactive applications with the help of Deep learning to analyze metadata of the packets and classify IP addresses either it is a DDOS attack or not. Finally, the functions and diagrams of our proposed system will be characterized and modeled in this paper.

1.3 Business Context

Ensuring network security is crucial and should be considered a top priority in any organization. It is imperative to have a robust security system that can protect and monitor network access from Distributed Denial-of-Service (DDoS) attacks, which can be challenging to detect due to their varying appearances and scenarios. [1]. Without apply security measures in place, a network can be left vulnerable to unauthorized use and malicious activities, leading to disastrous outcomes such as revenue losses and the complete destruction of organizations. Our system provides comprehensive capabilities to prevent ddos attacks during the use of interactive applications with high performance and suitable costs. By implementing this solution, the businesses or organizations can ensure the availability of their services and prevent revenue loss due to downtime caused by DDoS attacks.

2 Similar Systems

2.1 Academic

2.1.1 Detecting web attacks with end-to-end deep learning [2]

Web applications are attractive targets for cyber criminals. SQL injection [3] , cross site scripting (XSS)[4] and remote code execution are common attacks that can disable web services, steal private Data from service providers and users , so an end-to-end deep learning technique applied to detect cyber-attacks automatically in real-time and adapt efficiently, scalably, and securely to thwart them.they provides three contributions to the study of autonomic intrusion detection systems, First They evaluate the feasibility of an unsupervised/semi-supervised approach for web attack detection based on the Robust Software Modeling Tool (RSMT), which monitors the runtime behavior of web applications. Second, They describe how RSMT trains an auto encoder to encode and reconstruct the call graph for end-to-end deep learning. Third they Analyze the results of testing RSMT on both synthetic datasets and production applications with intentional vulnerabilities. At the end, they detect attacks, including SQL injection, cross-site scripting, and deserialization, with minimal domain knowledge and little labeled training data.

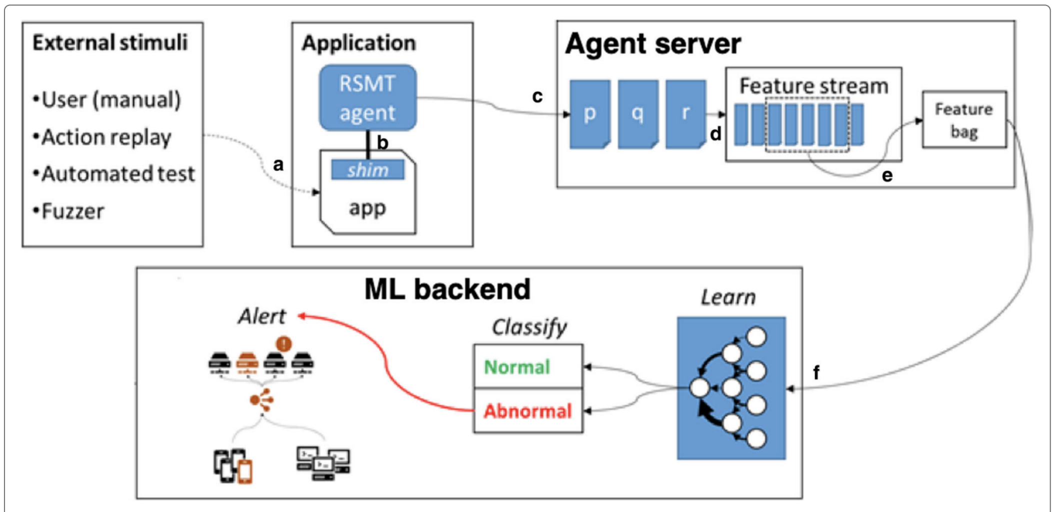


Fig. 1 The Workflow and Architecture of RSMT's Online Monitoring and Detection System

Figure 1: The Workflow and Architecture of RSMT's Online Monitoring and Detection System

2.1.2 Blacklist Based Malicious IP traffic Detection[5]

In this paper, they present their methodology for the detection of any connection going to or coming from a malicious IP address. They process the network traffic and compare the upcoming connection's Source and Destination IP addresses with IP Blacklist. Their detection is in real-time and blacklist are automatically updated every day. The first case, there's a connection to a malicious IP, the source IP is checked if it's hosted from their network by using "Local Addr" function that returns true if the IP address is hosted from their network and returns false if it's not, comparing the destination IP with the blacklist table if it is found then there's a connection to a malicious IP address and before they send an alert message they first check if they alerted about the same IP address in the current day, if this malicious ip is not detected in the alert table so they can add it to the blacklist detection table and also add it to the alert table that keeps the data for only one day to not alert on the same ip address twice per a day.

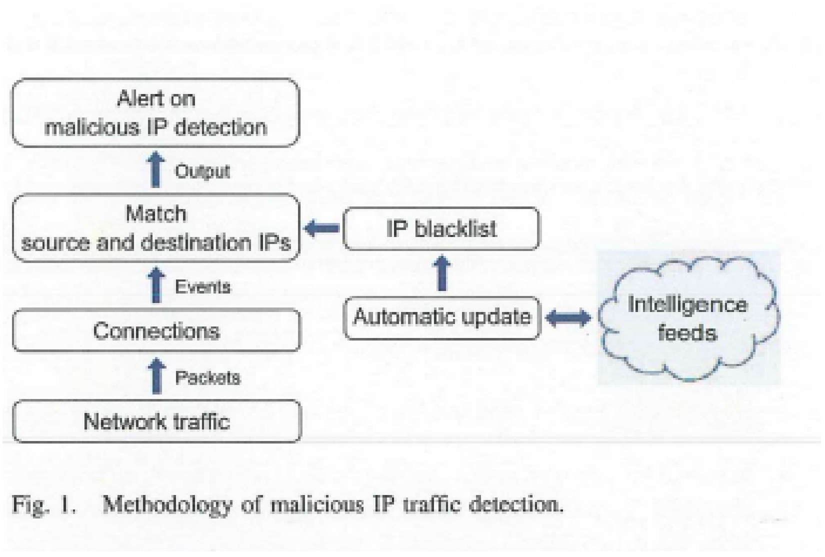

Alert on malicious IP detection
Output
Match source and destination IPs
IP blacklist
Events
Automatic update
Intelligence feeds
Connections
Packets
Network traffic
Fig. 1. Methodology of malicious IP traffic detection.

Figure 2: Methodology of malicious IP traffic detection

2.1.3 Classification of Application Traffic Using Tensorflow Machine Learning[6]

In this paper, The Application Traffic Classification method based on the machine learning was proposed to use in classification of network traffic instead of using The Payload Signature-based Classification method. The Payload Signature method has many Advantages like high accuracy and performance but it also has Many problems like it's computationally expensive for real-time To handle a large amount of traffic for high-speed networks, also It's too difficult for this method to cope with new patterns. For that reason, the application traffic classification method has Been approached because even new patterns of an application traffic has been found it classifies itself according to the patterns learned before this. Tensor flow has been used because of its simplicity to use. They are targeting to find more optimal learning features used in these experiments as future work.

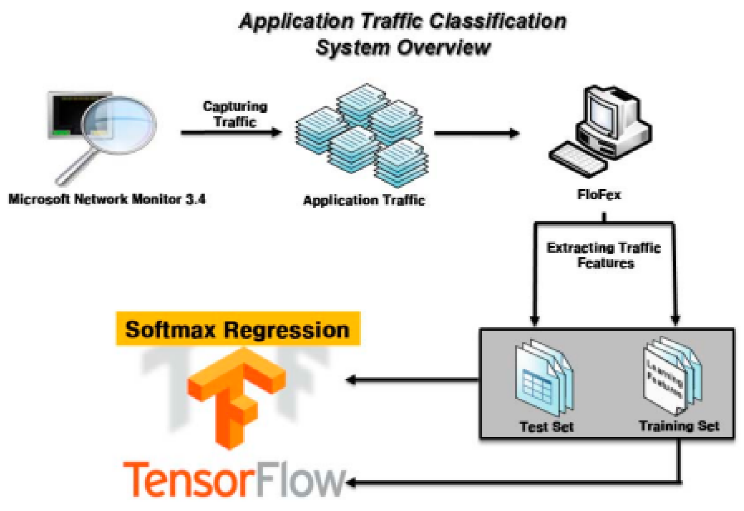

Application Traffic Classification System Overview
Capturing Traffic
Microsoft Network Monitor 3.4
Application Traffic
FloFex
Extracting Traffic Features
Softmax Regression
Test Set
Training Set
TensorFlow

Figure 3: Classification of Application Traffic Using Tensorflow Machine Learning

2.2 Business Applications

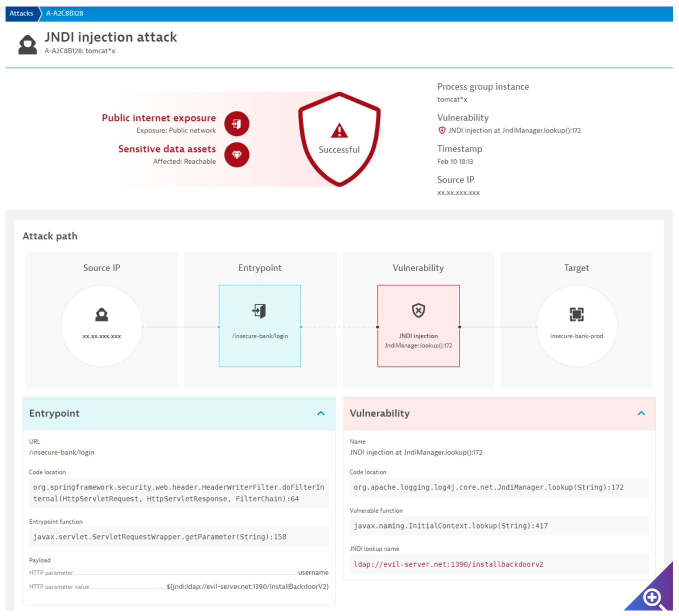
2.2.1 New Relic [7]

NETSCOUT

Figure 4

NETSCOUT is an application that protects network from DDoS attacks. It is a solution for network visibility and traffic intelligence that have been tested and proven in the world's largest, It is used for the most complex networks and helps in knowing who the threat actors are, their tools, behaviors and ongoing campaigns on a global basis.

2.2.2 Dyntrace



Process group instance
tomcat*x

Vulnerability
JNDI injection at JndiManager.lookup():172

Timestamp
Feb 10 18:13

Source IP
xx.xx.xxx.xxx

Entrypoint

URL
/insecure-bank/login

Code location
```
org.springframework.security.web.header.HeaderWriterFilter.doFilterIn
ternal(HttpServletRequest, HttpServletResponse, FilterChain):64
```

Entrypoint function
```
javax.servlet.ServletRequestWrapper.getParameter(String):158
```

Payload

HTTP parameter .... username

HTTP parameter value .... ${jndi:ldap://evil-server.net:1390/InstallBackdoorV2}

Vulnerability

Name
JNDI injection at JndiManager.lookup():172

Code location
```
org.apache.logging.log4j.core.net.JndiManager.lookup(String):172
```

Vulnerable function
```
javax.naming.InitialContext.lookup(String):417
```

JNDI lookup name
```
ldap://evil-server.net:1390/installbackdoorv2
```

Figure 5: Dyntrace

Dynatrace Application Security module includes real-time attack protection. Without any configuration, attacks can be detected and blocked based on code-level insights and transaction analysis.

3 System Description

3.1 Problem Statement

The problem addressed by this project is the need to develop an effective and accurate deep learning-based system that can detect Distributed Denial-of-Service (DDoS) attacks on a network. DDoS attacks are a significant threat to network security and can result in Website or application downtime, Security vulnerabilities, Increased infrastructure costs, and financial losses for organizations. There is a critical need for a reliable security system that can quickly and accurately identify and prevent DDoS attacks, thereby ensuring the security and integrity of the network. The goal of this project is to develop a robust binary classifier that can accurately distinguish between normal network traffic and traffic associated with DDoS attacks, thereby mitigating the effects of such attacks on the network.

3.2 System Overview

1. Traffic Capture: The system shall capture network traffic using (Scapy). The traffic is captured in real-time and processed to extract relevant information from the packets such as IP addresses, port numbers, and packet size.

2. Preprocessing and Analysis: The system preprocesses the extracted information and uses a pre-trained deep learning model to detect malicious packets. The extracted features from the packets are preprocessed and input into the model.

3. AI model: A deep learning model is trained on historical data to detect DDoS attacks. The model takes packet metadata as input and classifies them as normal traffic or DDoS traffic(malicious).

4. Blocking: If the deep learning model detects a DDoS attack, it automatically blocks the offending IP address using the firewall. The blocked IP address is added to a blacklist stored in the database.

5. Database: The system uses a MySQL database to store the list of known benign and malicious IP addresses and list them to there their corresponding labels (white-listed or black-listed). The system can retrieve IP addresses from the database when needed to determine if an incoming packet is benign or malicious.

6. User Interface: The system will provide a user-friendly web-based interface that displays history, makes the user able to start and stop the program, re-train the dataset, and block and unblock previous IPs. This makes it easy for users to monitor, control and understand the system's performance.

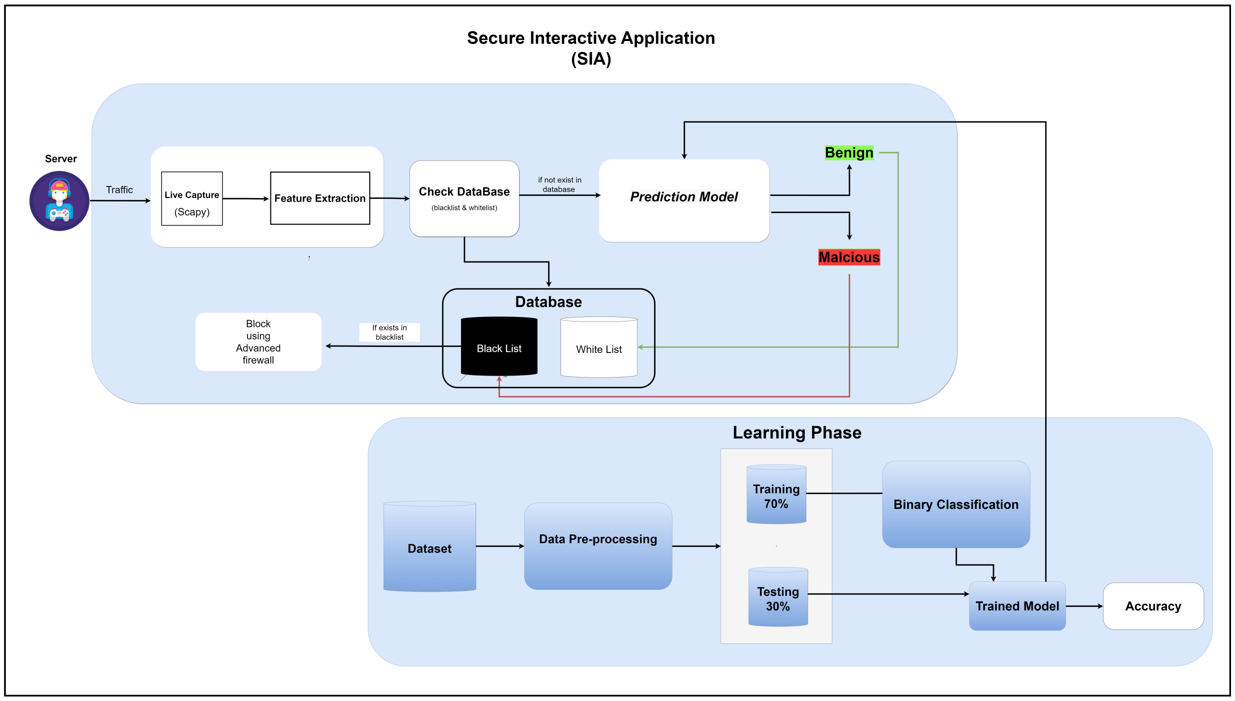


Figure 6: System overview

3.3 System Scope

The System will include

- innovative secure-oriented software that manages security features to protect the servers from being attacked.
- A traffic capture module that uses Scapy to capture network traffic in real time.
- A traffic preprocessing model that extracts features from the captured traffic data, such as Frequency of requests, Header Length, Protocol, and Packet Size.
- An AI classification model that uses a trained deep learning model to classify the preprocessed traffic as either benign or malicious.
- A MySQL database for storing the preprocessed traffic data and the results of the AI analysis.
- If the preprocessed traffic has been benign for more than time, the IP will be stored in the whitelist.
- The system shall block the malicious IP using Advanced Firewall and store it in the blacklist in Database.
- A user interface for displaying system logs.

3.4 System Context

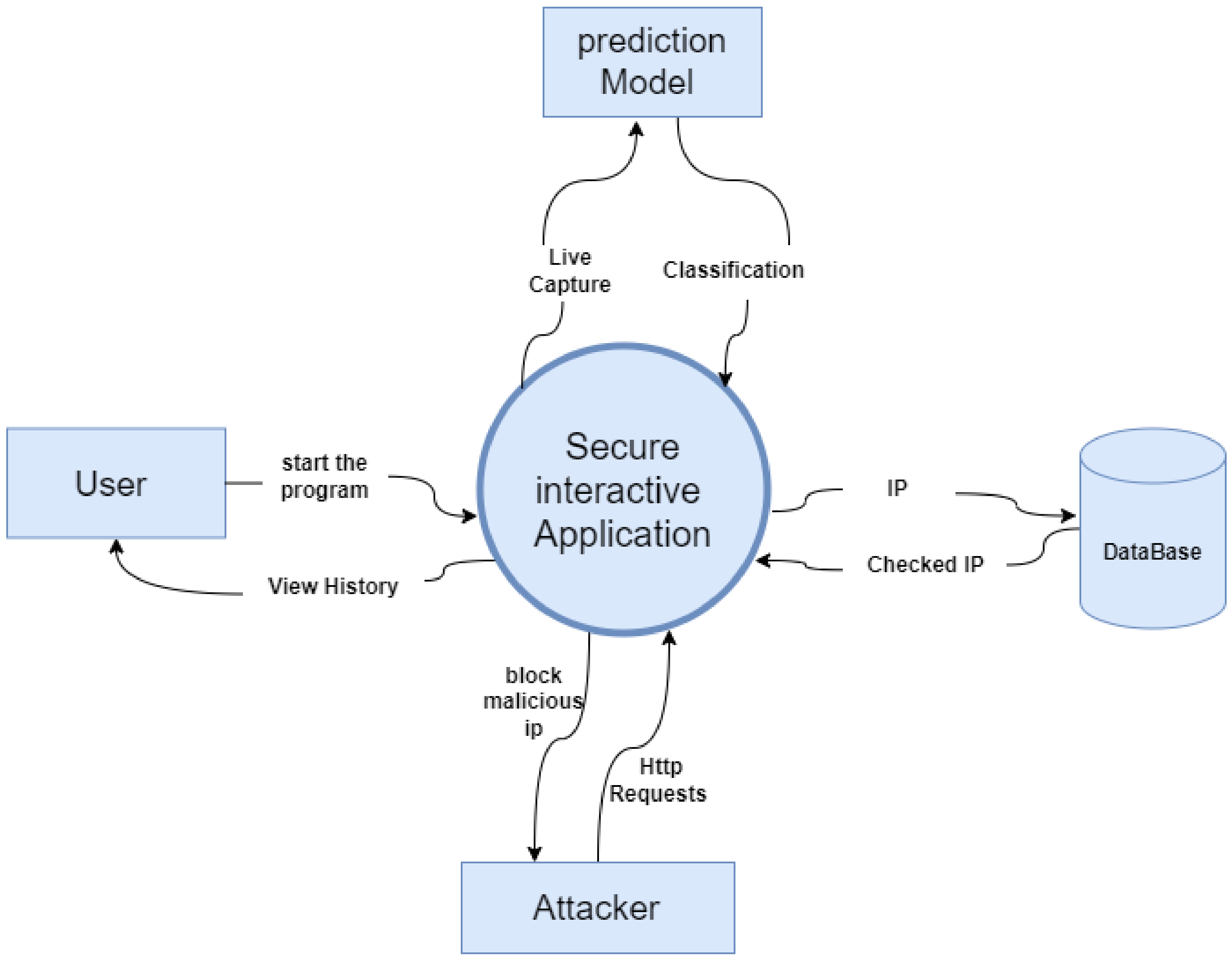


Figure 7: System context

3.5 Objectives

- Classifying and detecting DDOS attacks in real-time using AI and network traffic analysis.
- The system will provide real-time monitoring of network traffic.
- Building a prediction model to classify the incoming network traffic as either normal or malicious during the use of an interactive application.
- Providing a user interface to allow administrators to monitor and manage the system.
- Build MySQL database to store the list of known benign and malicious IPs that can be blocked without querying the database on subsequent requests.
- Implementing a firewall to automatically block incoming traffic from identified malicious IP addresses.

3.6 User Characteristics

- The users shall be network administrators, security analysts, or any individuals or organizations that are responsible for maintaining the security of their networks.

- These users need to have a basic understanding of network security and be familiar with the tools and techniques used to protect against DDoS attacks.
- These users need to have also some knowledge of machine learning and be able to interpret the results generated by the system.

4 Functional Requirements

4.1 System Functions

- ID:01 The user shall be able to start the program.
- ID:02 The user shall be able to stop the program.
- ID:03 The user shall be able to view the Attacks history.
- ID:04 The user shall be able to block an IP address from the whitelist manually.
- ID:05 The user shall be able to unblock an IP address from the blacklist manually.
- ID:06 The system shall be able to live capture the metadata.
- ID:07 The system shall be able to pre-process the captured data to send it to the model.
- ID:08 The system shall be able to check if the IP is in either the blacklist or whitelist.
- ID:09 The system shall be able to classify metadata.
- ID:10 The system shall be able to block the malicious IP.

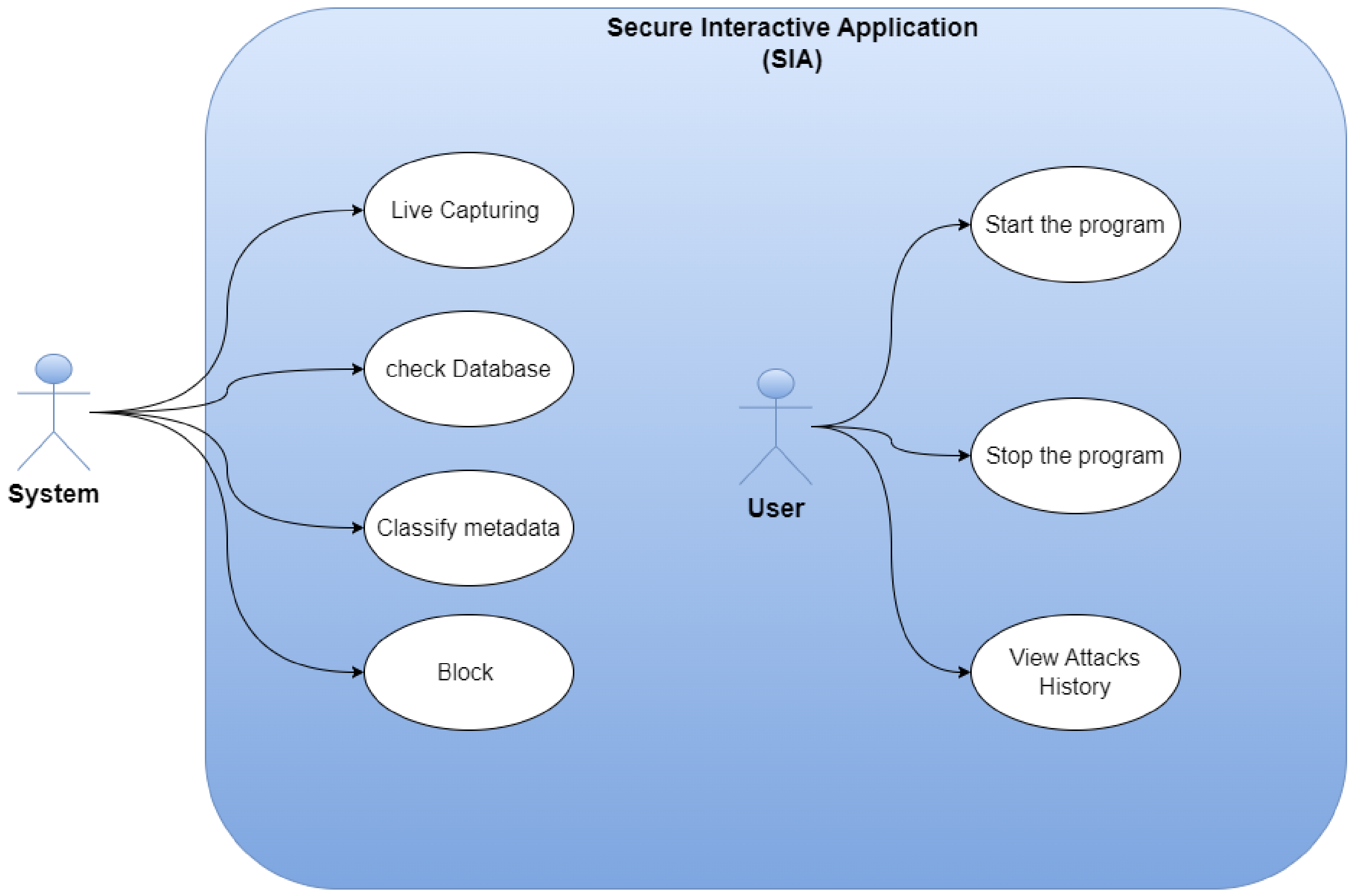


Figure 8: Use case Diagram

4.2 Detailed Functional Specification

Table 2: Live Capture

Name	sniff()
Code	ID:06
Priority	Extreme
Critical	The system won't be able to check metadata without capturing the traffic.
Description	Allow the system to capture the traffic.
Input	Network Traffic .
Output	captured packets details
Pre-condition	Connected to the interface
Post-condition	system shall sniff all the users traffic while using any interactive application the system.
Dependency	None
Risk	No internet connection

Table 3: Check Blacklist

Name	blacklisthas()
Code	ID:08
Priority	High
Critical	It's important to check whether the ip exists in the blacklist or not before passing it to the model.
Description	check captured IP addresses if exist in blacklist
Input	The IP address captured
Output	True or False
Pre-condition	live capturing.
Post-condition	IP address will not pass as it will be blocked
Dependency	ID:06
Risk	No database

Table 4: unblock

Name	unblock()
Code	ID:05
Priority	Medium
Critical	The user could unblock a selected IP
Description	Deletes the corresponding firewall rule that blocks the IP address, and updates the user interface grids to reflect the changes.
Input	Selected IP address by the user
Output	None
Pre-condition	An ip adress is blocked by the firewall
Post-condition	The selected ip adress will be unblocked and removed from the blacklist
Dependency	User shall select an ip address
Risk	None

Table 5: block

Name	block()
Code	ID:04
Priority	Medium
Critical	The user could unblock a selected IP
Description	Inserts the firewall rule that blocks the IP address, and updates the user interface grids to reflect the changes.
Input	Selected IP address by the user
Output	Remove ip from whitelist
Pre-condition	The IP should be in whitelist
Post-condition	The selected IP address shall be blocked and added to the blacklist
Dependency	User shall select an IP address
Risk	None

Table 6: Pre-process the captured data

Name	preprocess-data()
Code	ID:07
Priority	High
Critical	pre-processing the captured data before feeding it to the model
Description	It takes a datagram as input and converts the source and destination IP addresses to integer arrays, converts the source and destination ports to integers and creates an input array for the model.
Input	The input array includes source IP, destination IP, source port, destination port, packet size, and header size.
Output	returns the input array.
Pre-condition	Captured datagram should contain the required fields (source IP, destination IP, source port, destination port, packet size, and header size)
Post-condition	the output array can be fed to the machine learning model for classification.
Dependency	Pyshark and Scapy libraries,
Risk	None

5 Design Constraints

5.1 Standards Compliance

- Operating system: Windows 7, 8 and 10
- High-speed Internet Connection

6 Non-functional Requirements

6.1 Maintainability

The system shall be practically maintainable since it can be easily improved as it adapts to changes. Also, the code is refactored and this makes it easier to be used and understood.

6.2 Usability

The system shall be user-friendly as its interface shall be easy for any user to use.

6.3 Security

Our main goal is to keep the user secure from DDoS attacks by handling a high volume of traffic during a DDoS attack.

6.4 Performance

The program shall be able to accurately and quickly capture any traffic.

6.5 Reliability

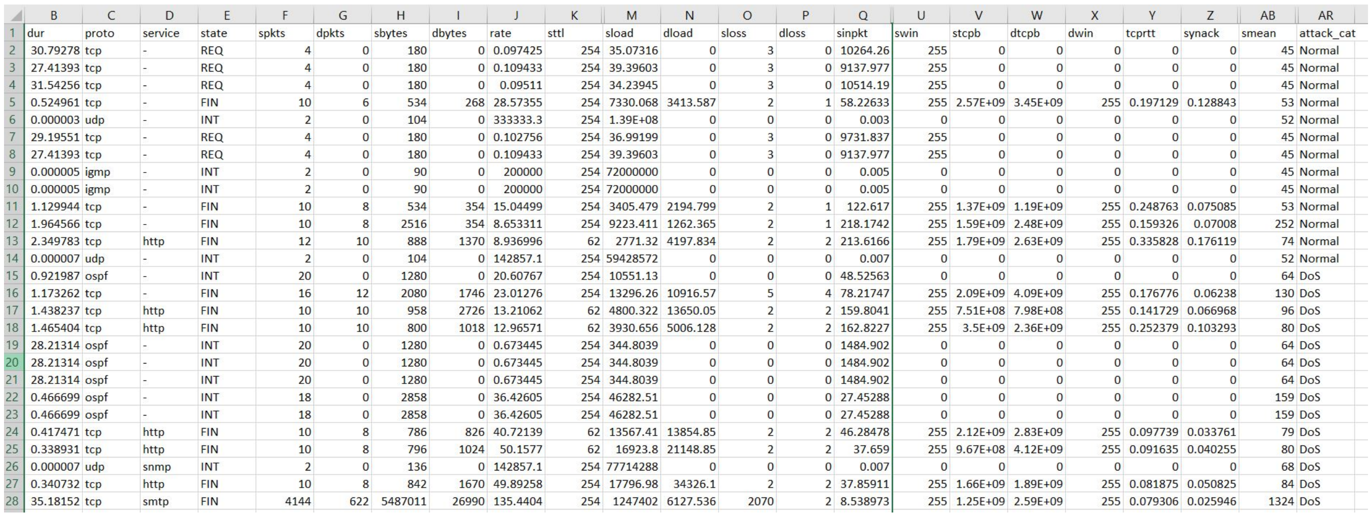
The program shall provide a high detection rate of whether an IP address is benign or malicious.

6.6 Availability

The system shall be available all the time whenever the user wants to capture and detect malicious activity

7 Data Design

We have utilized the UNSW-NB15 dataset to develop a predictive model for network security purposes. The dataset comprises over 40,000 instances of network traffic data with 47 features, which includes labels indicating whether the traffic is a normal connection or an attack. More than 24,000 instances are labeled as a DDos attack. The UNSW-NB15 dataset is a widely used resource in cybersecurity research, primarily due to the availability of real-world network traffic data and the presence of both normal and attack instances. However, it's crucial to keep in mind the limitations and biases of any dataset used for machine learning and evaluate the model's performance on new, unseen data to ensure its generalizability to real-world scenarios.



	B	C	D	E	F	G	H	I	J	K	M	N	O	P	Q	U	V	W	X	Y	Z	AB	AR
1	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	sload	dload	sloss	dloss	sinpkt	swin	stcpb	dtcpb	dwin	tcprtt	synack	smean	attack_cat
2	30.79278	tcp	-	REQ	4	0	180	0	0.097425	254	35.07316	0	3	0	10264.26	255	0	0	0	0	0	45	Normal
3	27.41393	tcp	-	REQ	4	0	180	0	0.109433	254	39.39603	0	3	0	9137.977	255	0	0	0	0	0	45	Normal
4	31.54256	tcp	-	REQ	4	0	180	0	0.09511	254	34.23945	0	3	0	10514.19	255	0	0	0	0	0	45	Normal
5	0.524961	tcp	-	FIN	10	6	534	268	28.57355	254	7330.068	3413.587	2	1	58.22633	255	2.57E+09	3.45E+09	255	0.197129	0.128843	53	Normal
6	0.000003	udp	-	INT	2	0	104	0	333333.3	254	1.39E+08	0	0	0	0.003	0	0	0	0	0	0	52	Normal
7	29.19551	tcp	-	REQ	4	0	180	0	0.102756	254	36.99199	0	3	0	9731.837	255	0	0	0	0	0	45	Normal
8	27.41393	tcp	-	REQ	4	0	180	0	0.109433	254	39.39603	0	3	0	9137.977	255	0	0	0	0	0	45	Normal
9	0.000005	igmp	-	INT	2	0	90	0	200000	254	72000000	0	0	0	0.005	0	0	0	0	0	0	45	Normal
10	0.000005	igmp	-	INT	2	0	90	0	200000	254	72000000	0	0	0	0.005	0	0	0	0	0	0	45	Normal
11	1.129944	tcp	-	FIN	10	8	534	354	15.04499	254	3405.479	2194.799	2	1	122.617	255	1.37E+09	1.19E+09	255	0.248763	0.075085	53	Normal
12	1.964566	tcp	-	FIN	10	8	2516	354	8.653311	254	9223.411	1262.365	2	1	218.1742	255	1.59E+09	2.48E+09	255	0.159326	0.07008	252	Normal
13	2.349783	tcp	http	FIN	12	10	888	1370	8.936996	62	2771.32	4197.834	2	2	213.6166	255	1.79E+09	2.63E+09	255	0.335828	0.176119	74	Normal
14	0.000007	udp	-	INT	2	0	104	0	142857.1	254	59428572	0	0	0	0.007	0	0	0	0	0	0	52	Normal
15	0.921987	ospf	-	INT	20	0	1280	0	20.60767	254	10551.13	0	0	0	48.52563	0	0	0	0	0	0	64	DoS
16	1.173262	tcp	-	FIN	16	12	2080	1746	23.01276	254	13296.26	10916.57	5	4	78.21747	255	2.09E+09	4.09E+09	255	0.176776	0.06238	130	DoS
17	1.438237	tcp	http	FIN	10	10	958	2726	13.21062	62	4800.322	13650.05	2	2	159.8041	255	7.51E+08	7.98E+08	255	0.141729	0.066968	96	DoS
18	1.465404	tcp	http	FIN	10	10	800	1018	12.96571	62	3930.656	5006.128	2	2	162.8227	255	3.5E+09	2.36E+09	255	0.252379	0.103293	80	DoS
19	28.21314	ospf	-	INT	20	0	1280	0	0.673445	254	344.8039	0	0	0	1484.902	0	0	0	0	0	0	64	DoS
20	28.21314	ospf	-	INT	20	0	1280	0	0.673445	254	344.8039	0	0	0	1484.902	0	0	0	0	0	0	64	DoS
21	28.21314	ospf	-	INT	20	0	1280	0	0.673445	254	344.8039	0	0	0	1484.902	0	0	0	0	0	0	64	DoS
22	0.466699	ospf	-	INT	18	0	2858	0	36.42605	254	46282.51	0	0	0	27.45288	0	0	0	0	0	0	159	DoS
23	0.466699	ospf	-	INT	18	0	2858	0	36.42605	254	46282.51	0	0	0	27.45288	0	0	0	0	0	0	159	DoS
24	0.417471	tcp	http	FIN	10	8	786	826	40.72139	62	13567.41	13854.85	2	2	46.28478	255	2.12E+09	2.83E+09	255	0.097739	0.033761	79	DoS
25	0.338931	tcp	http	FIN	10	8	796	1024	50.1577	62	16923.8	21148.85	2	2	37.659	255	9.67E+08	4.12E+09	255	0.091635	0.040255	80	DoS
26	0.000007	udp	snmp	INT	2	0	136	0	142857.1	254	77714288	0	0	0	0.007	0	0	0	0	0	0	68	DoS
27	0.340732	tcp	http	FIN	10	8	842	1670	49.89258	254	17796.98	34326.1	2	2	37.85911	255	1.66E+09	1.89E+09	255	0.081875	0.050825	84	DoS
28	35.18152	tcp	smtp	FIN	4144	622	5487011	26990	135.4404	254	1247402	6127.536	2070	2	8.538973	255	1.25E+09	2.59E+09	255	0.079306	0.025946	1324	DoS

Figure 9: Dataset2

8 Preliminary Object-Oriented Domain Analysis

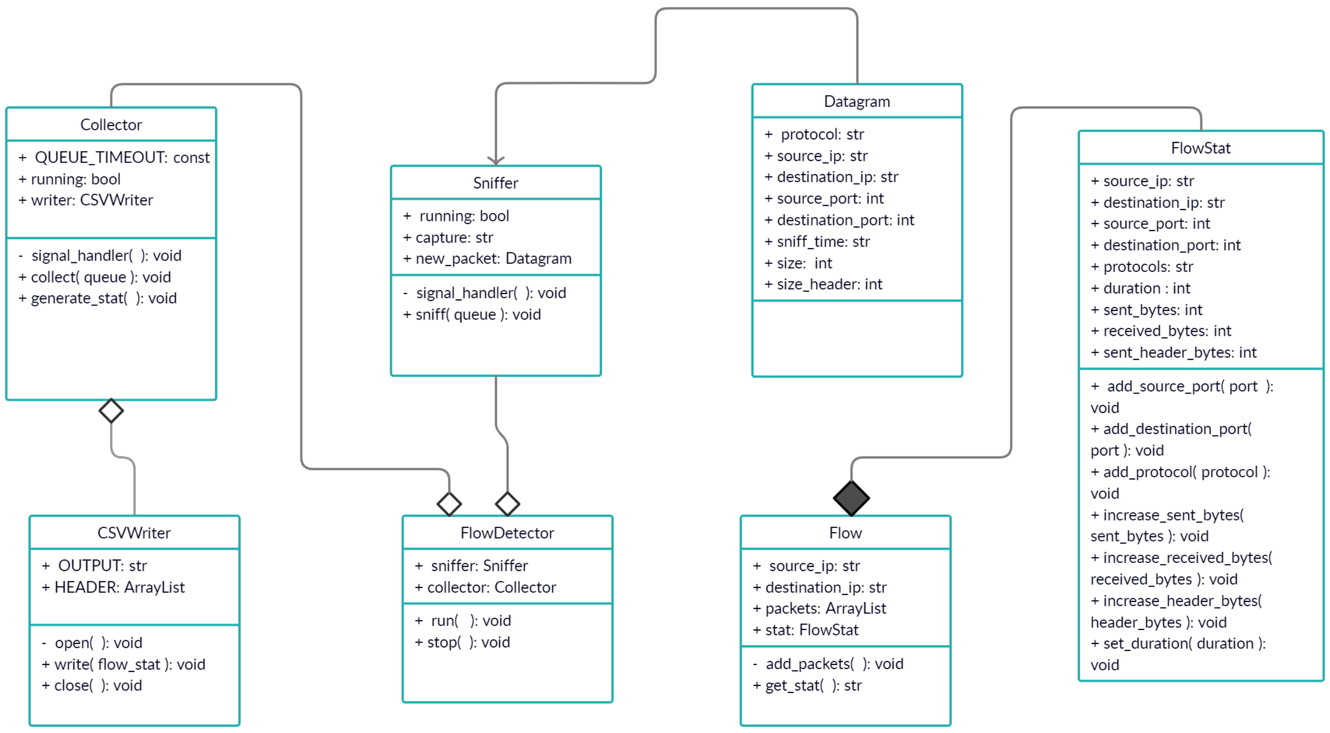


Figure 10: UML Diagram

9 Operational Scenarios

- **Scenario 1:** The user shall start the program to begin live capturing of received packets. Then use different interactive applications while the software is running in the background. The system shall check if any of the captured metadata exists in the blacklist or the whitelist, since it's found in the blacklist, it will block the source IP address. Moreover, the user will be able to check the history of blocked IP addresses.
- **Scenario 2:** When the user starts the program, the program will check the blacklist, if the program didn't find the IP in the blacklist or the whitelist, the metadata will enter the prediction model to be classified, if metadata is benign nothing shall happen and the process will be continued normally.
- **Scenario 3:** The user shall start executing the application, and the application starts by live capturing the upcoming packets, then someone is trying to DDoS attack by sending a lot of packets till the server is down, so the application starts to classify these packets by passing them to the prediction model, finally, they are classified as malicious packets and these malicious IP addresses are blocked using the advanced firewall and added to the blacklist
- **Scenario 4:** The user shall start the program to begin live capturing of received packets. Then use different interactive applications while the software is running in the background. The system shall check if any of the captured metadata exists in the blacklist or the whitelist, since it's found in the whitelist, the packet will pass normally without passing by the prediction model.

10 Project Plan

Table 7: Project Plan

Task Name	Start date	Due date	Duration
SRS Document	12/2/2022	12/17/2022	15 days
Finding Dataset	12/3/2022	12/3/2022	1 day
Implementation: Check Blacklist	12/12/2022	12/13/2022	2 days
Implementation: Alert Message	12/14/2022	12/14/2023	1 day
Working on GUI	12/15/2022	1/1/2023	16 days
SDD Document	12/23/2022	2/22/2023	29 days

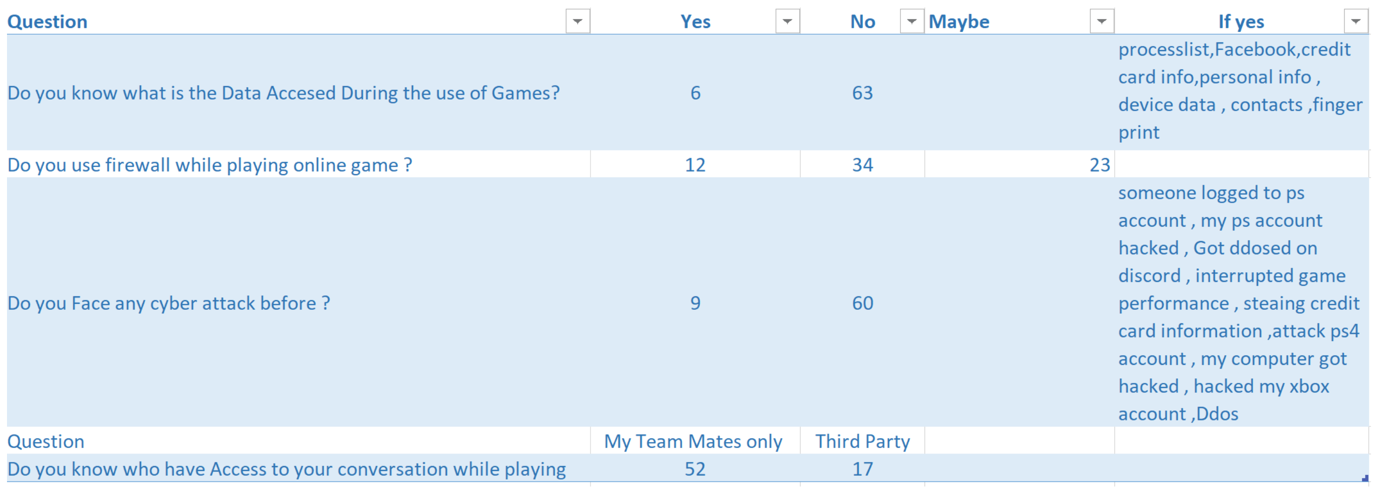
11 Appendices

11.1 Definitions, Acronyms, Abbreviations

- **DDOS Attack [8]:**A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt normal traffic to a specific server, service, or network by flooding the target or its surrounding infrastructure with Internet traffic.
- **IP:** internet protocol

11.2 Supportive Documents

- **High orbit ion Cannon[9] :**The High Orbit Ion Cannon (HOIC) is a network stress application created by the hacktivist group Anonymous, intended to replace the Low Orbit Ion Cannon (LOIC). This open-source software is used to perform denial of service (DoS) and distributed denial of service (DDoS) attacks by inundating the target system with excessive amounts of HTTP GET and POST requests.
- **TensorFLow [10]:**is an end-to-end open-source platform for machine learning, that provides stable Python and C++ APIs. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML, and developers easily build and deploy ML-powered applications.
- **Scapy:[11]**Scapy is a powerful Python-based interactive packet manipulation program and library that allows users to sniff, dissect, and forge network packets. It permits the production of new packets from scratch, packet sniffing on a network interface, packet decoding into raw data, and packet manipulation for a variety of uses, such as testing, troubleshooting, and security analysis.



Question	Yes	No	Maybe	If yes
Do you know what is the Data Accesed During the use of Games?	6	63		processlist,Facebook,credit card info,personal info , device data , contacts ,finger print
Do you use firewall while playing online game ?	12	34	23	
Do you Face any cyber attack before ?	9	60		someone logged to ps account , my ps account hacked , Got ddosed on discord , interrupted game performance , steaing credit card information ,attack ps4 account , my computer got hacked , hacked my xbox account ,Ddos
Question	My Team Mates only	Third Party		
Do you know who have Access to your conversation while playing	52	17		

Figure 11: Survey

Survey

References

[1] Anteneh Girma, Moses Garuba, Jiang Li, et al. "Analysis of DDoS Attacks and an Introduction of a Hybrid Statistical Model to Detect DDoS Attacks on Cloud Computing Environment". In: *2015 12th International Conference on Information Technology - New Generations*. 2015, pp. 212–217. DOI: 10.1109/ITNG.2015.40.

[2] Yao Pan, Fangzhou Sun, Zhongwei Teng, et al. "Detecting web attacks with end-to-end deep learning". In: *Journal of Internet Services and Applications* 10 (Dec. 2019). DOI: 10.1186/s13174-019-0115-x.

[3] William G. J. Halfond, Jeremy Viegas, and Alessandro Orso. "A Classification of SQL-Injection Attacks and Countermeasures". In: 2006.

[4] Shashank Gupta and Brij B Gupta. "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art". In: *International Journal of Systems Assurance Engineering and Management* 8 (Sept. 2015). DOI: 10.1007/s13198-015-0376-0.

[5] Ibrahim Ghafir and Vaclav Prenosil. "Blacklist-based malicious IP traffic detection". In: *2015 Global Conference on Communication Technologies (GCCT)*. 2015, pp. 229–233. DOI: 10.1109/GCCT.2015.7342657.

[6] Jee-Tae Park, Kyu-Seok Shim, Sung-Ho Lee, et al. "Classification of application traffic using tensorflow machine learning". In: *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2017, pp. 391–394. DOI: 10.1109/APNOMS.2017.8094156.

[7] Jaspreet Singh and Karuna Ghai. "Comparing New Relic with other Performance Monitoring Tools". In: *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. 2022, pp. 1–5. DOI: 10.1109/ICRITO56286.2022.9964706.

[8] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, et al. “Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy”. In: *2019 International Carnahan Conference on Security Technology (ICCST)*. 2019, pp. 1–8. DOI: 10.1109/CCST.2019.8888419.

[9] Mahadev, Vinod Kumar, and Krishan Kumar. “Classification of DDoS attack tools and its handling techniques and strategy at application layer”. In: *2016 2nd International Conference on Advances in Computing, Communication, Automation (ICACCA) (Fall)*. 2016, pp. 1–6. DOI: 10.1109/ICACCAF.2016.7749002.

[10] Li-Der Chou, Chia-Wei Tseng, Meng-Sheng Lai, et al. “Classification of Malicious Traffic Using TensorFlow Machine Learning”. In: *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. 2018, pp. 186–190. DOI: 10.1109/ICTC.2018.8539685.

[11] Rohith Raj S, Rohith R, Minal Moharir, et al. “SCAPY- A powerful interactive packet manipulation program”. In: *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*. 2018, pp. 1–5. DOI: 10.1109/ICNEWS.2018.8903954.