APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 7 No. 1 (2026) p. 1000-1014
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Network Intrusion Detection System (NIDS) to Detect Botnet Attack using Rule-based and Signature-based Approach

Nurqaseh Jeffy Irana Abdul Hakim[1], Isredza Rahmi A Hamid[1*]

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: rahmi@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2026.07.01.054

Article Info

Received: 3 August 2025
Accepted: 15 June 2026
Available online: 30 June 2026

Keywords

Botnet detection, Network Intrusion Detection System (NIDS), Network Traffic Analysis

Abstract

Botnets are a serious cybersecurity threat that allows Denial of Service (DOS) attacks and distribution of malwares. The existing Intrusion Detection System (IDS) is uncapable to identify the evolving and modern botnet threats. Therefore, a Network Intrusion Detection System (NIDS) to detect botnet using rule-based and signature-based approach is developed. This project aims to detect botnets using rule-based and signature-based detection by analysing the behaviour of the network traffic based on the predefined rules and signatures. We design the NIDS using Agile methodology. The NIDS can identify, alert and generate report for users that handles sensitive information. This tool provides valuable insights to regular users that wants to mitigate any future cyberattacks.

1. Introduction

Botnets known for executing multiple types of attacks such as DOS attacks and malware distribution making it a significant threat. The current security measures such as encryption are irrelevant as botnets can mimic as an authentic traffic[1]. The architecture of botnets heavily relies on the interaction over communication channels also known as C&C (Command and Control) channel. Bot masters utilise these channels to communicate and control the bots as long as possible. Botnets utilise Peer-to-peer (P2P) network to serve as both client and server, not requiring a centralized point. This type of botnet is more resilient even though nodes are offline allowing network to continuously operate under the attacker's control[2].

Network Intrusion Detection System (NIDS) is system based on data mining to detect network any network intrusions[3]. NIDS works by analysing real network traffic data as shown in Fig. 1. Data captured usually contains packet header information and will be stored in flat files. NIDS will analyse these flat files in batches. A crucial step for NIDS is the data filtering step that is done before feeding the data into anomaly detection module. Data filtering step ensures only important network traffic will be thoroughly analysed. NIDS are typically placed near firewall within the network perimeter to analyse traffic. In general, NIDS detects for hostile actions within network traffic such as Denial of Service (DOS) attacks, port scanning, and packet with malicious payload. NIDS is responsible to detect and alert suspicious activities throughout the network using both rule-based and signature-based detection [4][5]. Existing NIDS have difficulty in detecting botnets due to high volume traffic, produce high false alarm for any unusual patterns, and unable to detect encrypted traffic [6]. This project examines traffic analysis to detect botnet attacks using rule-based and signature-based approach. Rule-based detection identifies attacks based on the predefined rules while, signature-based detection matches known threat pattern for detection. Therefore, both mitigations can reduce the risk of cyber threat immensely. Traffic analysis for botnet

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1000-1014

detections is based on analysing payload, TCP or UDP contents, traffic behaviour, and the time frame of packets sent between hosts[7]. The objectives of this are:

1. To design a network intrusion detection system (NIDS) based on a static approach.
2. To develop a NIDS by introducing rule-based and signature-based approach to detect botnet in network traffic.
3. To implement alpha and beta testing on the developed system to target users.

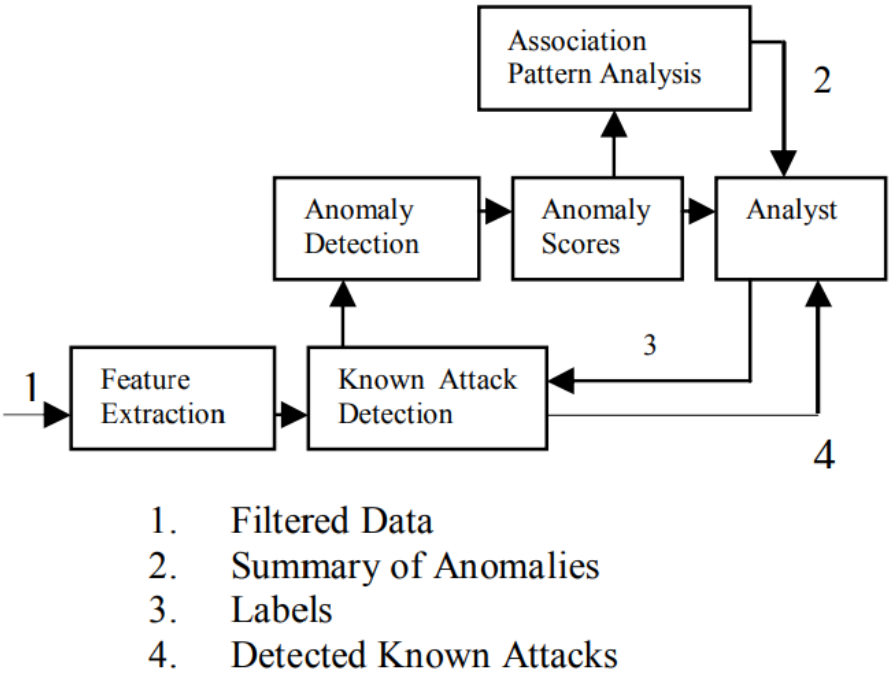


Fig. 1 *NIDS System[3]*

The proposed NIDS is straightforward and easy to use. The user interface (UI) allows user to easily configure, monitor, upload files, and system maintenance. The system involves a comprehensive report function and logs for view logs for any auditing purposes. Furthermore, report generation allows NIDS to produce detailed reports and maintain logs when botnets are identified. Reports consist of valuable insights such as attack types, date and time of attack and type of activity considered as suspicious. The following information are important for analysing threats and improve any functions of the system in the future. Lastly, it is crucial for this NIDS tool to be scalable for easy configuration. It will be easier to add new rules and signatures to the system for higher rate of botnet identification. A scalable NIDS tool ensures system to adapt to the constant evolving cyber-attacks in this digital era. This tool uses the Agile methodology ensuring a iterative development for continuous improvement of the system. This system is tested in multiple scenarios and controlled environment to make sure an accurate botnet detection. Besides botnets, this system can also be used to detect any related behaviours and provide a robust defence against botnet threats.

The remainder of this paper is structured as follows: Section 2 covers related work, Section 3 outlines the methodology, Section 4 details the design and analysis, and Section 5 presents the results and implementation. The conclusion highlights key findings and future directions.

2. Related Work

This section discusses the existing NIDS tools such Suricata, Zeek and, Snort. Besides that, discussion about the mechanism of botnets.

2.1 How Botnet Works

Botnets are self-propagating were infecting vulnerabilities found during the initial phase. Infected hosts are exploited to carry out malicious activities such as DoS attacks and malware injection. Botnet attacks consist of multiple phases, starting from its birth to its spread across networks. Fig. 2 illustrates botnet lifestyle consisting of its multiple phases before, during, and after botnet attack. Initial phase known as the Initial Injection, where the attacker scans vulnerability of target such as unpatched updates or backdoors. Second phase, hosts are infected and will run shellcode to identify malware in network database. By File Transfer Protocol (FTP), Hypertext

Transfer Protocol (HTTP) or Peer-to-peer (P2P)[8]. The bot will be installed on the targeted machine and behaves as bot or zombie.

During connection phase, bot establishes a connection with C&C server for updates and instructions by the bot master. This phase occurs every time host restarts ensuring bot master can provide commands to botnet. Next phase, the C&C is fully functioning and can be used to distribute commands to the army. Commands received by bots are carried out under the attacker's supervision. Lastly, this phase focuses on the maintaining and updating the bots by installing an update binary.

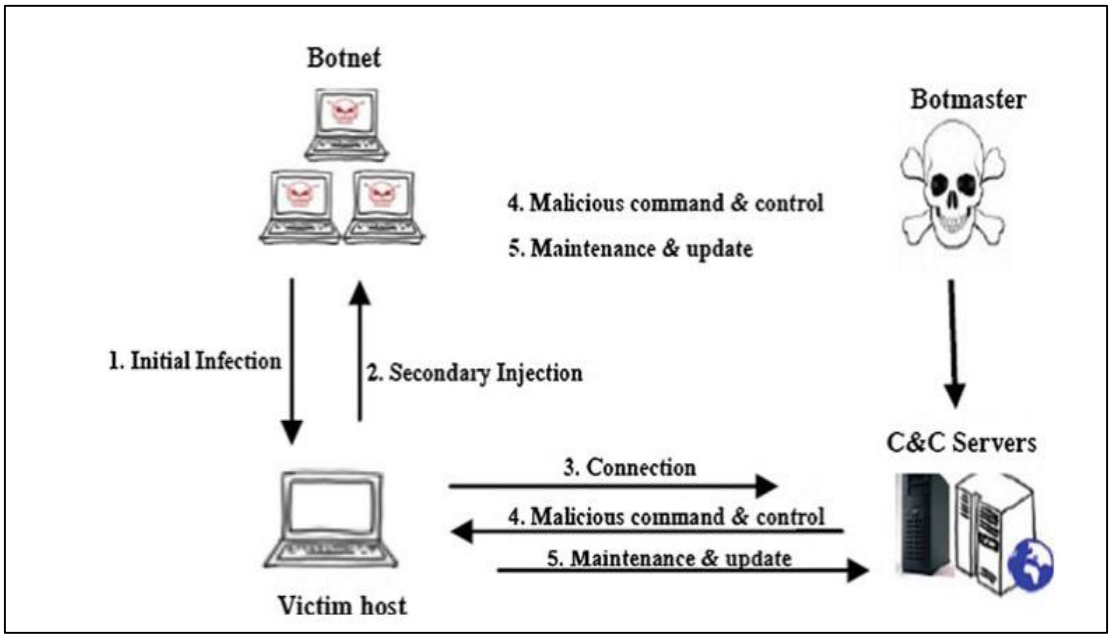


Fig. 2 *Botnet lifestyle [8]*

2.2 Botnet Detection Approach

There is various approach used to detect botnet within the network. The examples are rule-based detection and signature-based detection. Both approaches enable an easy yet functional botnet detection tool The implementation of both approaches, enables the proposed project to identify botnets accurately. Rule-based typically means that botnet will be identified based on the packet-level rules. Using only rule-based for this project will not guarantee the accuracy of botnet detection. Rule-based depends highly on predefined rules. Next, signature-based will match traffic against a database of known botnet attacks such as Mirai botnet[9]. Signature consists of strings or any expression presenting malicious patterns. It might be inefficient or slow when it detects new attacks. The combination of both approach enables a higher probability of threat coverage, reduce any false positive during the analysis process and enables the NIDS to be adaptable.[10]

The features from rule-based are explicit rules that operates with "if-then". It helps to detect any suspicious network behaviours such as "IF packet size > 1500 bytes AND protocol is ICMP THEN alert" Besides that, rule-based enables to detect more than signature-based such Zero-day attack[11]. Next, signature-based is based on its pattern matching feature, which is highly efficient in analysing byte sequence, malicious code snippets and known botnet attack[12]. Therefore, it is precise when analysing botnet attack within network traffic based on the stored signatures in database.

2.3 Existing Network Intrusion Detection System (NIDS)

This section discusses the existing NIDS tools that are used to detect botnets such as Suricata, Snort, and Zeek.

2.3.1 Suricata

Suricata is optimal when placed after Firewall and before Demilitarized Zone (DMZ) within the network. This setup allows effective monitoring of traffic (incoming and outgoing) from the DMZ. It is important to external-facing servers such as web and mail servers. Suricata's limit is placing behind the firewall because it is unable to detect lateral traffic. In general, Suricata can handle diverse network traffic because it has multi-threaded processing and advanced rule syntax including various keywords and modifiers[13]. It enables Suricata to read more than simple packet header by matching a deeper inspection of packets to increase positive detection rate.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1000-1014

2.3.2 Zeek

Zeek leans toward passive monitoring, suited for segmented network traffic analysis by a network TAP and SPAN ports. Zeek focuses on high-level network event and analysing packet-by-packet. Zeek use its own domain-specific scripting language for complex development for specific network and requirements. Zeek's is efficient in logging and contextualizing traffic analysis based on cybersecurity setups[14].

2.3.3 Snort

Snort's latest version Snort3.0 is known for its passive IDS. Its optimal position is similar to Suricata for a comprehensive network coverage. Like Suricata, Snort implements multi-threaded processing to support high demands of modern networks. The predefine rules are accurate for higher detection rate. The Snort can handle large volume of traffic incoming and outgoing.

Table 1 displays the comparison between Snort, Zeek, Suricata and the proposed NIDS tool. Each tool has its own advantages. The primary use of Snort and Suricata is both Intrusion Prevention System (IPS) and Intrusion Detection System (IDS) to detect and mitigate threats. Zeek's main use is to monitor network and highly used in network forensics along with Suricata. The proposed tool focuses primarily on detecting botnets while providing valuable insights by report and logs. Next, the approach used in Snort, Suricata, and proposed tool are similar as these tools used Rule-based and Signature-based to identify malicious patterns. However, Suricata consists of Advanced Rule Syntax that enables an accurate detection of cyber-attacks. Zeek implements behavioural analysis for a robust event log of network activity. Zeek allows deeper analysis and detection of anomaly based on its scripting language.

Snort, Zeek and Suricata utilises Command Line Interface (CLI) and external User interface (UI) or Application Programming Interface (API). Users are not required to login into the web application but must access the tools by CLI, SSH or integrate any UI such as Splunk. In contrast, proposed NIDS tool allows user to login and register with a security implementation such as hash password, session management and password management. The proposed tool is suitable with users that are not familiar with CLI or new to NIDS tools. The dashboard of proposed NIDS enables users to view their past analysis and upload PCAP file[15].

Snort is highly customizable because of its rule language. Zeek is known to be flexible because of the Zeek scripting language allowing high customization of detection and analysis. Suricata mainly use the multi-threaded processing and advanced rule syntax which provided high customization. The proposed NIDS tool is scalable for any new rules and change in architecture. Snort, Zeek, and Suricata provides configurable alerts integrated with Security Information and Even Management System. However, proposed NIDS tool provides insightful report generation and maintain logs of any detected botnet.

Table 1 *Comparison between Snort, Zeek, Suricata, and proposed NIDS tool*

NIDS Tools	Snort	Zeek	Suricata	NIDS Tool (Proposed)
Primary Use	Intrusion Detection (IDS and IPS)	Network Security Monitor and Network Forensics	IDS, IPS, and Network Forensics	IDS specifically for Botnet Attacks and provide valuable insights
Detection Technique	Rule-based and Signature-based	Behavioural Analysis, Anomaly Detection, and Scripting-based	Rule-based, Signature-based, and Advanced Rule Syntax	Static Approach (Rule-based and Signature-based)
How to Access	Command-line (CLI) or external UI or API	CLI or external UI or API	CLI or external UI and API	User-friendly interface for registration, login, dashboard and monitoring logs.
Customization Rules	Rule language and multi-threaded	Highly customizable (independent scripting language)	Advanced rule syntax and multi-threaded	Scalable for modification and rules
Alert	Configurable alert	Robust logging of event	Configurable alert	Report generation and log valuable insights
Report Generation	No	No	No	Yes

3. Methodology

Agile methodology was used in developing the NIDS tool. This method emphasises an iterative development, constant update based on feedback and user collaboration. Agile allows continuous improvement based on user's feedback that aligns with dynamic requirement of the tool to detect botnets. The workflow of the system development system includes six phases which are requirement gathering, design, implementation, testing, deployment and feedback as listed in Table 2. These phases ensure adaptable and effective tool for users.

The development process begins with gathering the requirements such as the problems with current tool to create a proper problem statement. This phase allows the preparation of proposal that outlines the NIDS's tool objectives, scope and expected results. Next, the architecture and UI of the NIDS will be designed by Draw.io. Unified Modelling Language (UML) were used to present the NIDS's functions and interaction between different modules. For implementation phase, the NIDS tool is developed with Phyton featuring rule-based and signature-based detection modules. Rule-based detection will be implemented by inserting predefined rules from Snort.

Moreover, testing phase ensures the NIDS tool performs flawlessly under various scenarios including real-time attacks. Deployment phase involves executing the tool to users with detailed guide and manual. Lastly, feedback phase allows users to provide insights and suggestion for further improvements to ensure the NIDS to remain relevant with the evolving threats.

Table 2 *System development workflow*

Phase	Task	Output
Requirement Gathering and Analysis	1. Comparing the existing tools to identify the concerns 2. Research to find the problem and a way to solve the issue 3. Preparing a proposal consisting of problem statement, objectives, scope and expected results 4. Develop a Gantt chart based on the Agile method	1. Proposal 2. Gantt Chart
Design	1. Create an architecture of the tool 2. Design UI (User Interface) 3. Define software and hardware requirements 4. Define user, functional and non-functional requirements	1. Architecture design of tool 2. UI Design 3. Software and hardware requirements 4. User, functional and non-functional requirements
Implementation	1. Write source code for tool 2. Implement features and functions 3. Set predefined rules and signatures of botnet	1. NIDS tool to detect botnet that can identify, detect and log or generate reports of the attacks. 2. Rule-based detection and Signature-Based detection module
Testing	1. Implement test plan 2. Evaluate tool's response when botnet is detected. 3. Solve errors during iterative process	1. Results from test plan. 2. Results from detecting on multiple scenarios. 3. Error detection and problem-solving solutions
Deployment	1. Deliver solution to potential users 2. Provide guides for installation and use 3. Modify and configure tool based on users	1. Installation and usage guide. 2. Modified and optimized NIDS tool
Feedback	1. Receive feedback or evaluation from users 2. Analyse and document the feedback for future optimization 3. Improve tool based on the analysed feedback from users	1. User acceptance testing form 2. Upgraded tool

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1000-1014

3.2 Software and Hardware Requirements

The software requirements for the proposed NIDS tool are listed in Table 3. The operating system (OS) used is windows as it has strong network capabilities, stable, and it is a security focused OS. The language used for this project is Python along with Flask framework. Python has an extensive library for network programming such as Scapy. It is suitable for analysing network traffic and packet inspection. Moreover, Flask provides a framework that enables easier backend developing. Front end development consists of HTML, CSS, and JSON. The proposed tool has database to manage logs, user accounts, and event of detected botnet. MySQL is used along with PhpMyAdmin for logging event meta data and its structure rule sets. For network analysis, libcap is used included in Python library. For providing insights and report generation, PDF generation are implemented into the backend development. It allows visualization of the logs and any detected events.

Table 3 *Software Requirements for proposed tool*

Software requirement	Software	Description
Operating system	Windows	Windows 11 (Latest version)
Programming language and Framework	Python language and Flask framework	Python has comprehensive libraries, and Flask is Python's web framework
Database	MySQL and PhpMyAdmin	Log event metadata, user accounts, and activity log
Libraries	Scapy and flask_mail	To analyse packets and receive emails from proposed tool
Report tool	PDF generation	To generate PDF file of reported events

The hardware requirements for this project are listed in Table 4. The device's processor (CPU) is Intel i5 and 18GB of RAM. It is essential for processing packets and to run the NIDS tool, running rule-based and signature-based detection simultaneously. 8GB of RAM is sufficient to store network traffic buffers, load both rules and signatures, run database and the tool itself on Visual Studio Code. The Huawei Matebook D14 consists of SSD for quick application loading and boot. This device has 512GB of SSD storage, crucial for performance of tool and operations of database.

Table 4 *Hardware Requirements for proposed tool*

Hardware Requirements	Specification	Description
CPU	Intel i5	Efficient for packet processing and support during high traffic
RAM	8GB	For small networks
Storage type	SSD (512GB)	To store database operations and process analysis

4. Analysis and Design

This section discusses the architecture of proposed project, functional and non-functional requirements, diagrams and the interface of the system.

4.1 System Architecture

Fig. 3 illustrates the architecture of the system. After user upload PCAP file, the NIDS will start analysing it based on the detection engine. The engine consists of rule-based and signature-based for PCAP file analysis. Rule-based will detect any files with pattern matching and rules set from the Snort 3.0. Rule-based includes "if-then" and analysing packet size.

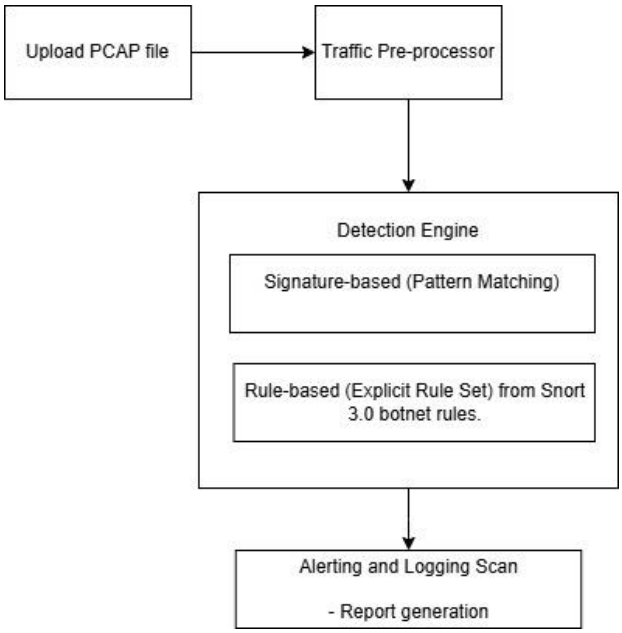


Fig. 3 *System architecture of NIDS*

4.2 System Requirements

Functional requirement are the specific features, functionalities and behaviours of the NIDS tool that must considered to achieve its purpose to detect botnets by analysing the behaviour of the network traffic. These requirements can be taken from analysing the tool's tasks and function to ensure an effective identification of botnet attacks. Table 5 illustrates the functional requirement for the NIDS tool.

Table 5 *Functional requirement for NIDS tool*

Functional Requirement	Description
Real-time Traffic Analysis Module	NIDS tool will analyse network traffic in real time to detect anomalies and potential botnet attacks.
Rule-based Detection Module	NIDS tool will allow configuration of predefined rules to identify botnet behaviours.
Signature-based Detection Module	NIDS tool will implement signature-based detection to detect known botnet signatures.
Notification Module	NIDS tool provides real-time alert to users when suspicious activity is detected.
Log and Report Generation Module	NIDS tool will generate detailed report and log any attacks detected for future use.

Non-functional requirements refer to the overall usability and security of the tool including its scalability and compatibility of the NIDS tool. These requirements ensure the NIDS to operate effectively under various scenarios to meet users' expectations. Table 6 shows the non-functional requirements that plays a crucial role in defining the quality of the NIDS tool.

Table 6 *Non-Functional requirement for NIDS tool*

Non-functional Requirement	Description
Performance	NIDS toll will process and analyse the network traffic efficiently to detect any botnet threats
Usability	NIDS tool consists of an easy to navigate user interface for users.
Scalability	NIDS tool can handle increasing size of network traffic sizes.
Compatibility	NIDS tool compatible with various Operating Systems (OS) such Windows OS
Security	NIDS tool includes a security measure to protect tool's integrity.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1000-1014

4.3 System Analysis

This section showcases the Unified Modelling Language (UML) of the NIDS tool interactions with the users.

4.3.1 Use Case Diagram

Fig. 4 shows the user case diagram consists of only users and the features of the NIDS tool. However, this project does not specific admin account. New rules and signatures must be added manually into the backend development. User able to view report and analyse network, email and system notifications and report generation or viewing logs of attacks.

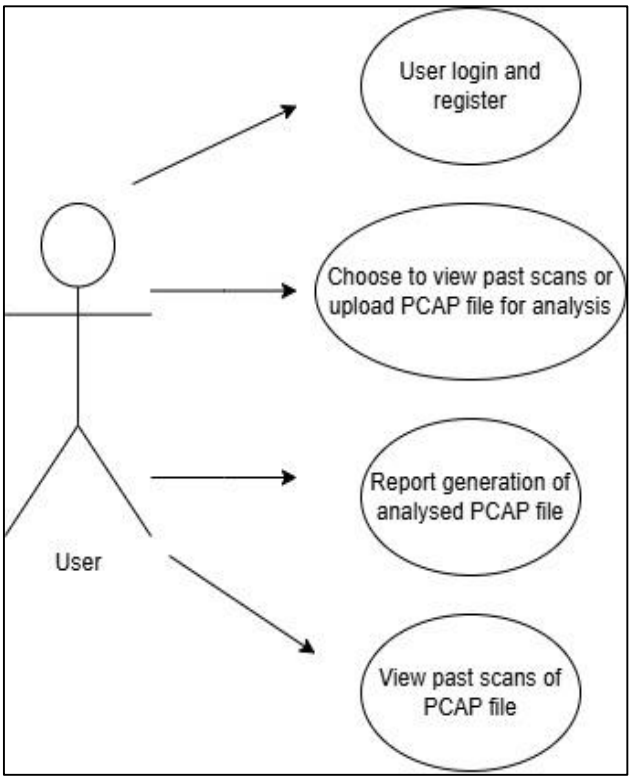


Fig. 4 *User case diagram*

4.3.2 Flowchart Diagram

Fig. 5 illustrates the flowchart of the proposed NIDS tool. User will have to login or register if they have no account. After registration, user is required to login before accessing other pages of the NIDS. First page that will be shown after logging in is the dashboard. Dashboard has two main option which are view past scans and upload PCAP file for analysis. User can choose whichever activity they wish to carry out. If past scans are selected, user can view past logs and generate a PDF file containing the details of the event. User can sort based on the date. If user chooses to upload PCAP file, user is required to upload a PCAP file containing botnet traffics and wait for analysis to complete. User can view the details of the scan go back to dashboard. The uploaded PCAP file will be analysed by predefined rules and signatures of botnets. The NIDS identifies payload content and packet size related to botnet attack. The NIDS detects for any known IRC Botnet that uses C&C channel. Botnets are also analysed by recognizing the port used whether if it's unusual for regular traffic behaviours.

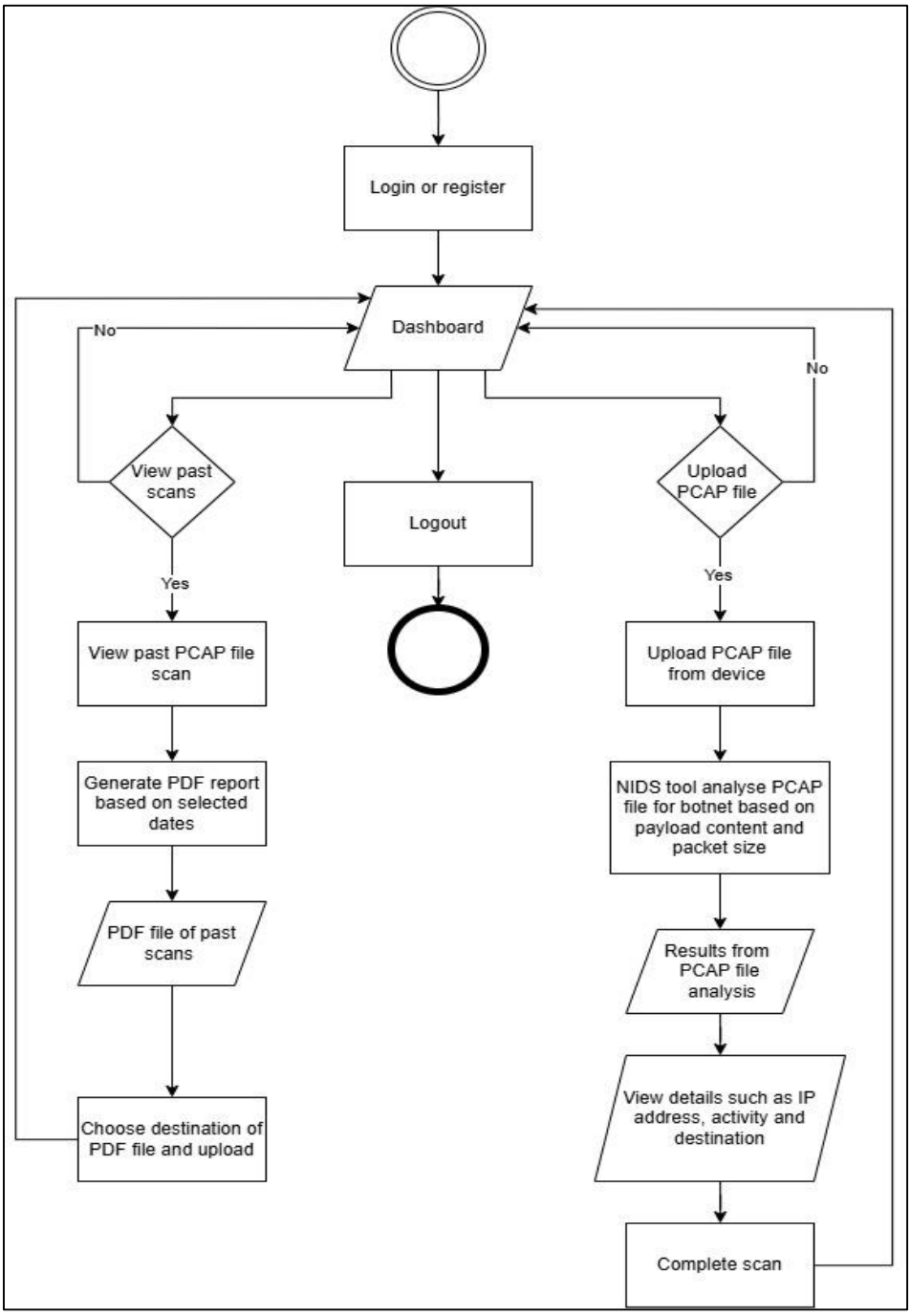


Fig. 5 *Flowchart of proposed NIDS tool*

4.4 System Design

This section discusses the general functions of the tool and modules used to detect botnet attacks. Modules used in this NIDS tool are PCAP file analysis module, report generation module, login/registration module, and view details of scans module.

4.4.1 Dashboard Module

In Fig 6, the of dashboard's UI is shown. The dashboard consists of two main menus which user can choose between view past scan or upload PCAP file. Top of page shows the username of user and other options of this system.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1000-1014

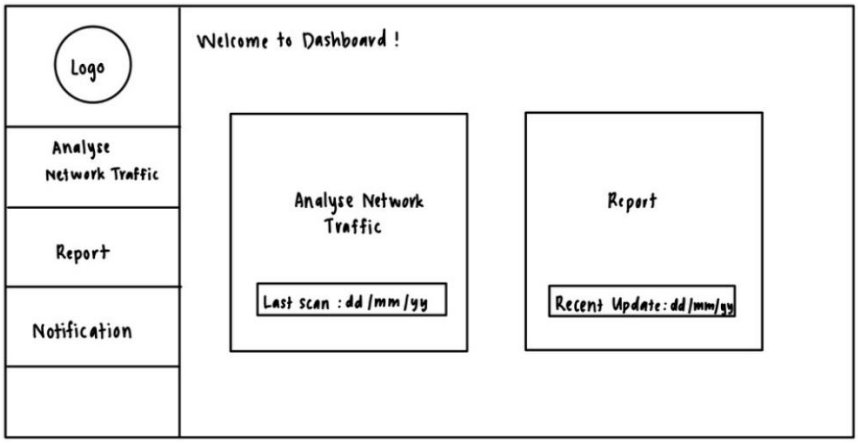


Fig. 6 *UI of dashboard page*

4.4.2 Scanning Module

Fig. 7 illustrates the page for users to upload their PCAP file from their device. Users can upload PCAP or PCAPNG files from their device and upload to the tool. When user click start scan, user will have to wait for the scanning process to finish to view the details of any finding from the uploaded file. Fig. 8 illustrates the past scan pages. This page consists of any logged botnet attacks gained from the analysis of the uploaded PCAP file. It has the relevant details such as time, date, activity, and sender/receiver IP address. User can select date to download the following PDF file for report generation. The report allows user to do deeper analysis or auditing purposes.

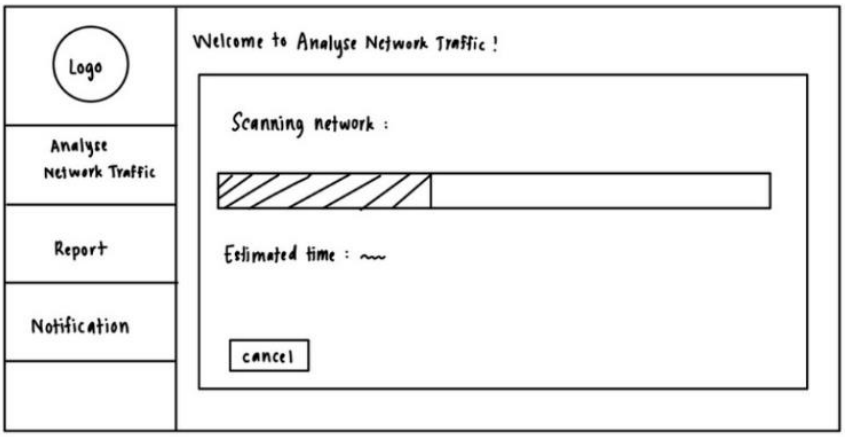


Fig. 7 *UI of upload PCAP file page*

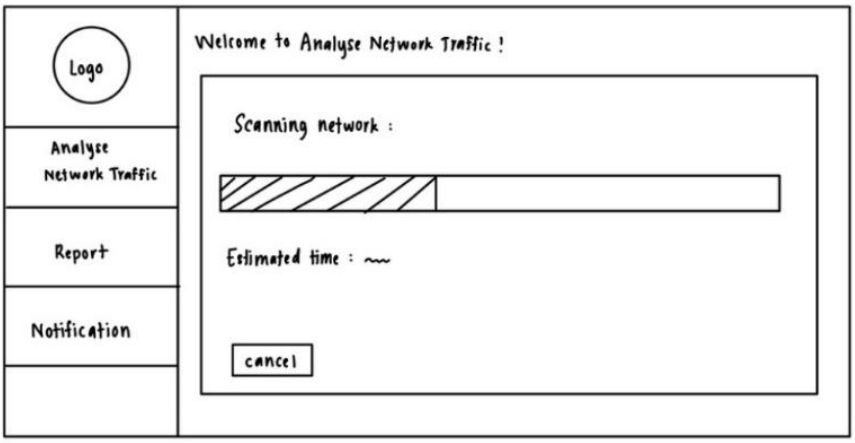


Fig. 8 *UI of past scan page*

4.5 Test Plan

This section outlines the test plan for the proposed NIDS tool primarily detecting botnet attacks using rule-based and signature-based approach. This test plan aims to ensure the NIDS can accurately detect botnet attacks from the uploaded PCAP file. It also aims to provide a positive User Experience (UX) based on the easy and user-friendly UI of the tool. This tool will also be testes to discover if it generates an informational report of scans. The functionality of the tool including, functional UI, accurate botnet detection, informational report, and scalability of the tool. Software used for application testing are Windows 11 OS, Wireshark (capture botnet traffic), traffic generator and PCAP file containing botnet attacks. This test implements the Agile methodology emphasising on the accuracy and functionality of both approaches.

Table 7 shows the categories of the testing phase of the tool. The categories are detection function, UI and reporting function. The detection function will only detect botnet attacks or any malicious behaviours of the uploaded PCAP file. It can also produce minimal false positive results and differentiate between benign and an attack. The UI ensures that user can easily view past scans, upload PCAP file, register account, and filter logs based on date. Furthermore, in the reporting category user can generate report consisting of valuable insights and ensuring it is a PDF format. The report generated must align with the scan results and ensures it is accurate.

Table 7 *Test plan*

Category	Description	Key Testing Focus Areas
Detection function	Able to identify botnet attacks accurately based on the uploaded PCAP file	Verify its accuracy and detection of any known botnet activity such as C&C server and malware transfers.
High accuracy	Ensures proposed NIDS to accurately detect botnet attacks and minimise false-positive rate.	Only malicious traffic trigger alert and differentiate between benign and real attack.
User interface	Functioning interface for proposed NIDS such as dashboard, upload PCAP file, view scans and generate report	Easy to navigate around the web application and ability to easily access all pages with minimal effort.
Report generation	Able to generate report with valuable insights and accurate details from the scan.	Ensure successful PDF file generated filled with information of the scan.

5. Result and Discussion

This section explains the focus of the NIDS tool with using static approach based on rule and signature. Along with both approaches is the user-friendly UI and informational reporting functionality. The NIDS is developed with the architecture focusing on botnet attacks, maintaining or updating rules. The upload PCAP module allows the NIDS to analyse traffic for any botnet attacks and log into the database. The module implements rule-based and signature-based approach to capture any botnet related packets. The UI module ensures easy use of the NIDS including monitoring, reporting and uploading file. Report module is crucial to gain valuable insights from data stored in database. The security module for this system includes password hashing for database, session management and required login before accessing any page.

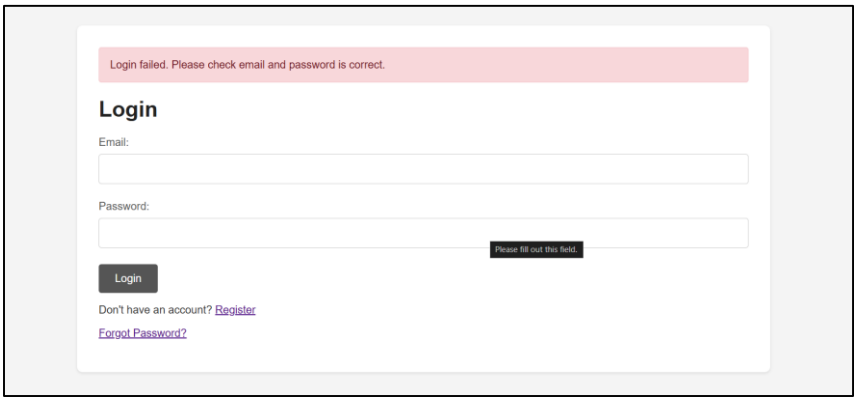
5.1 Login Module with Security Mechanism

Fig. 8 displays the coding in Python for the login required. User must register account before accessing any page. This ensures authorized access only to the NIDS dashboard. The source code explains how all pages except login, registration, and reset password page does not require any logins.

```
@app.route("/logout")
@login_required
def logout():
    user_id=get_user_id()
    session.pop('user_id', None)
    session.pop('username', None)
    flash('Logged out. Login again', 'info')
    return redirect(url_for('login'))

@app.route("/dashboard")
@login_required
def dashboard():
    username=session.get('username', 'Guest')
    user_id=get_user_id()
    return render_template('dashboard.html', username=username)
```

(a)



(b)

Fig. 8 *(a) Login required to access NIDS dashboard (b) Login page UI*

For forgot password features, user will be directed to the reset password page exclusively for forgot password purposes. User is required to input the email address used and wait for an email. The email has the reset password link for user to insert a new password. Fig. 9 shows the reset link being sent to the input email address. The password link is valid for 5 minutes. After 5 minutes is up, link is expired and useless for password reset. Fig. 10 shows that it will be sent immediately to the user's email address. Fig. 11 shows that user has received the link in the email.

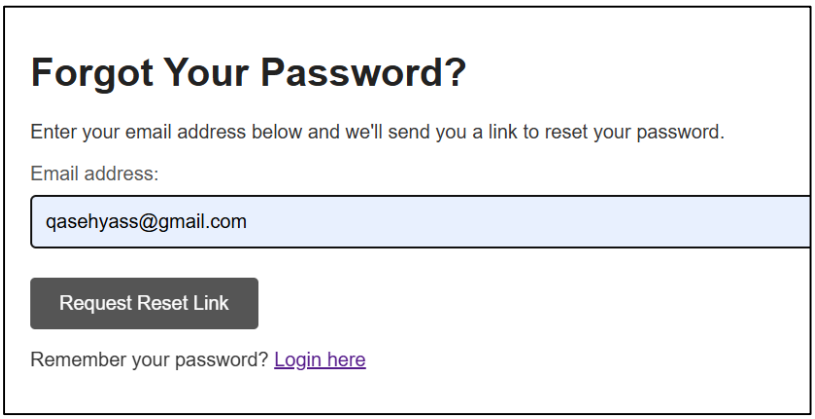


Fig. 9 *Forgot password page*

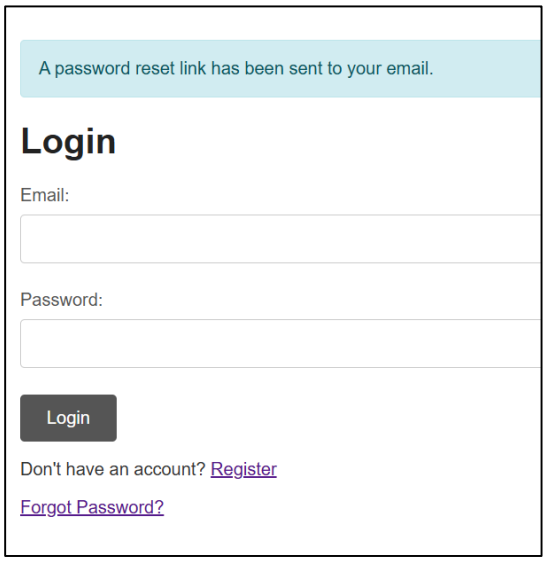


Fig. 10 *Reset password link sent to email*

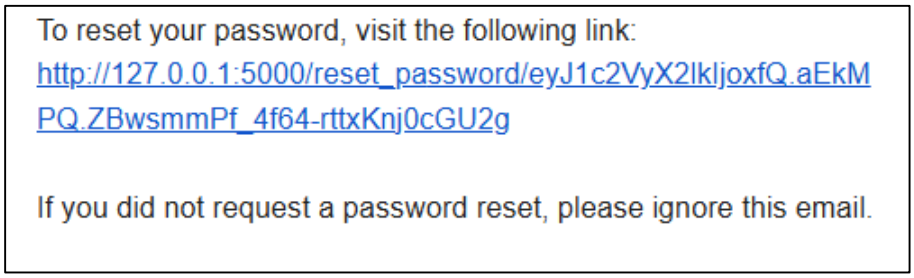
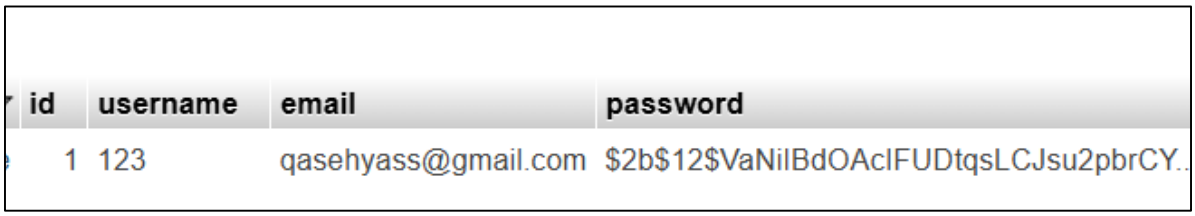


Fig. 11 *Reset password link*

Lastly, the hashed password in the database can be seen in Fig. 12. Hashed passwords with salting is crucial in this proposed NIDS tool. It is because, this ensures no unauthorized access and easy guessing of user's password based on the rainbow table. Password are hashed in the database ensuring the inability to bypass security and access someone else's account. For password change, user will need to change their password by inserting their old password.



id	username	email	password
1	123	qasehyass@gmail.com	$2b$12$VaNilBdOAclFUDtqsLCJsu2pbrCY..

Fig. 12 *Hashed and salting password in database*

5.2 User Acceptance Testing (UAT) Result

This section discusses about the results of UAT of the proposed NIDS. There are 21 respondents of the google form. Users consist of regular computer users trying to learn about botnet attacks. This project targets user with minimal knowledge of cybersecurity expanding its experience in botnet attacks. The scale of the UAT is from 1 to 7 (Really Dissatisfied to Really Satisfied). Majority are satisfied with the overall proposed NIDS.

Fig. 13 shows the NIDS to be overall satisfied reaching 10 out of 21 users with a score of 7. The proposed NIDS was easy to navigate, detect accurately based on rule and signature approach, UI that was functional, functional PCAP file upload, functional view past scans page and lastly able to provide report with valuable insights.

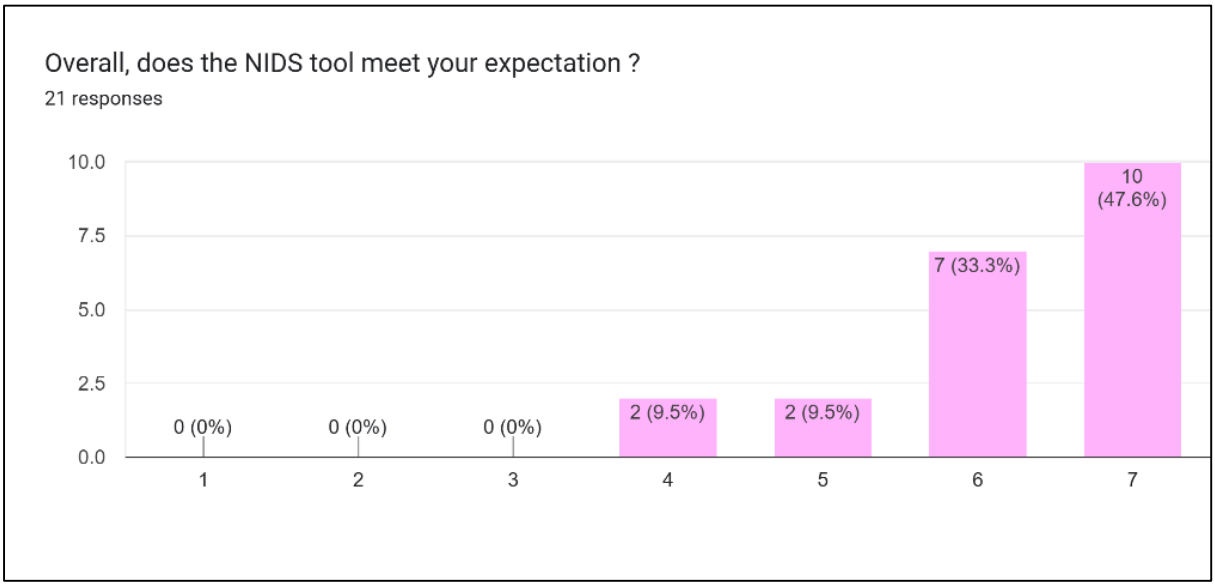


Fig. 13 *Conclusion from the UAT*

6. Conclusion

In conclusion, this project focuses on the UX, accuracy y of both approaches and the information contained in the report. The employment of rule-based and signature-based static approach enables the NIDS tool to identify and log malicious traffic indicating botnet attacks. One of the main highlights of this project is to deliver a useful yet simple NIDS tool for regular users. It is crucial to ensure the system can detect botnet attacks and produce insightful reports. This NIDS tool is suitable for post-incident analysis and making security decisions of the system. This NIDS tool enables anyone understand botnet and view botnet's behaviours within traffic. This tool is useful in research about evolving botnets and identify new rules or signatures. Future work for this project is integrating Machine Learning (ML) for a robust detection of any anomaly behaviours within the traffic. Besides that, real-time alert systems will be implemented in the future to minimize attacks and damage of the attack. Lastly, implementing a mechanism that enables automatic update of rules and signatures to keep up with the emerging botnet threats.

Acknowledgment

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

This journal requires that all authors take public responsibility for the content of the work submitted for review. The contributions of all authors must be described in the following manner:

The authors confirm contribution to the paper as follows: ***study conception and design:*** *N. J. I. Abdul Hakim, I. R. A. Hamid;* ***data collection:*** *N. J. I. Abdul Hakim, I. R. A. Hamid;* ***analysis and interpretation of results:*** *N. J. I. Abdul Hakim, I. R. A. Hamid;* ***draft manuscript preparation:*** *N. J. I. Abdul Hakim, I. R. A. Hamid. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] H. Awan, "A Comprehensive Guide to Botnet," Efani. Accessed: Dec. 30, 2024. [Online]. Available: https://www.efani.com/blog/botnet

[2] D. Zhao *et al.*, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput Secur*, vol. 39, pp. 2–16, Nov. 2013, doi: 10.1016/j.cose.2013.04.007.

[3] R. Bane and N. Shivsharan, "Network Intrusion Detection System (NIDS)," *Emerging Trends in Engineering & Technology, International Conference on*, vol. 0, pp. 1272–1277, Jan. 2008, doi: 10.1109/ICETET.2008.252.

[4] K. Alieyan, A. Almomani, M. Anbar, M. Alauthman, R. Abdullah, and B. B. Gupta, "DNS rule-based schema to botnet detection," *Enterp Inf Syst*, vol. 15, no. 4, pp. 545–564, Apr. 2021, doi: 10.1080/17517575.2019.1644673.

[5] A. Shaikh and P. Gupta, "Advanced Signature-Based Intrusion Detection System," in *Intelligent Communication Technologies and Virtual Mobile Networks*, G. Rajakumar, K.-L. Du, C. Vuppalapati, and G. N. Beligiannis, Eds., Singapore: Springer Nature Singapore, 2023, pp. 305–321.

[6] H. M. Elshafie, T. M. Mahmoud, and A. A. Ali, "Improving the Performance of the Snort Intrusion Detection Using Clonal Selection," in *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, 2019, pp. 104–110. doi: 10.1109/ITCE.2019.8646601.

[7] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013, doi: https://doi.org/10.1016/j.comnet.2012.07.021.

[8] G. K. Venkatesh and R. Anitha, "Botnets: A Study and Analysis," *Advances in Intelligent Systems and Computing*, vol. 246, pp. 203–214, Nov. 2014, doi: 10.1007/978-81-322-1680-3_23.

[9] N. Duffield, P. Haffner, B. Krishnamurthy, and H. Ringberg, *Rule-Based Anomaly Detection on IP Flows*. 2009. doi: 10.1109/INFCOM.2009.5061947.

[10] A. Shaikh and P. Gupta, "Advanced Signature-Based Intrusion Detection System," 2023, pp. 305–321. doi: 10.1007/978-981-19-1844-5_24.

[11] R. Bane and N. Shivsharan, "Network Intrusion Detection System (NIDS)," *Emerging Trends in Engineering & Technology, International Conference on*, vol. 0, pp. 1272–1277, Jan. 2008, doi: 10.1109/ICETET.2008.252.

[12] A. J. Alzahrani and A. A. Ghorbani, "Real-time signature-based detection approach for SMS botnet," in *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, 2015, pp. 157–164. doi: 10.1109/PST.2015.7232968.

[13] A. Gupta and L. Sen Sharma, "Performance Evaluation of Snort and Suricata Intrusion Detection Systems on Ubuntu Server," 2020, pp. 811–821. doi: 10.1007/978-3-030-29407-6_58.

[14] A. A. E. Boukebous, M. I. Fettache, G. Bendiab, and S. Shiaeles, "A Comparative Analysis of Snort 3 and Suricata," in *2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*, 2023, pp. 1–6. doi: 10.1109/GlobConET56651.2023.10150141.

[15] A. Waleed, A. F. Jamali, and A. Masood, "Which open-source IDS? Snort, Suricata or Zeek," *Computer Networks*, vol. 213, p. 109116, 2022, doi: https://doi.org/10.1016/j.comnet.2022.109116.