Polymorphic Attack Feature Validation: Bridging the Gap Between Intrusion Detection and Evolving Threats.

by

Raksha Begwani

A Capstone Research Project submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of IT Security (AI)

Faculty of Business & IT

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

Aug 2024

CAPSTONE RESEARCH PROJECT REVIEW INFORMATION

Submitted by: **Raksha Begwani**

Masters in IT Security (AI)

Project title: Polymorphic Attack Feature Validation: Bridging the Gap Between Intrusion Detection and Evolving Threats.

The Project was approved on Aug 19th, 2024, by the following review committee:

Review Committee:

Research Supervisor	Prof. Shahram Heydari
Second Reader	Prof. Pooria Madani

The above review committee determined that the Project is acceptable in form and content and that a satisfactory knowledge of the field was covered by the work submitted. A copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

ABSTRACT

This project focuses on enhancing the detection of polymorphic attacks, which can evade traditional Intrusion Detection Systems (IDS) by changing their form with each attack. While IDS are crucial for network security, their effectiveness diminishes against such dynamic threats. The project aims to identify key features exploited by polymorphic attacks, enabling the creation of a feature list to improve detection. Using the SlowHTTP tool for generating attack profiles and the LycoStand tool for essential feature extraction, this research seeks to develop effective mechanisms to analyze polymorphic attacks and its features, addressing the limitations of IDS in identifying these attacks.

Keywords: Polymorphic attack; IDS; Feature Analysis; DDoS/DoS; Feature Extraction tool.

AUTHOR'S DECLARATION

I hereby declare that this project consists of original work of which I have authored. This is a true copy of the work, including any required final revisions, as accepted by my committee.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this work to other institutions or individuals for the purpose of scholarly research. I further authorize the University of Ontario Institute of Technology (Ontario Tech University) to reproduce this work by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my work may be made electronically available to the public.

Raksha Begwani

ACKNOWLEDGEMENTS

I would like to express my heartfelt thanks to Prof. Shahram S. Heydari for his consistent support and mentorship. His insights and direction were invaluable in shaping the development and completion of my project.

TABLE OF CONTENTS

STATEMENT OF CONTRIBUTIONS

I confirm that I am the sole author of this project, and that it has not been published or submitted for publication elsewhere. I have adhered to all standard referencing protocols to properly acknowledge contributions from other sources.

Table of Figures

1. Introduction

1.1 Intrusion Detection System

In basic terms, intrusion is defined as unwelcome or illegal interference, sometimes with malevolent intent, aimed at gathering sensitive information or beginning harms on an organization's internal networks or systems. Intrusions are primarily carried out by third parties; however, it can also include insiders misusing their authority. An Intrusion Detection System (IDS) is a hardware or software solution that detects and informs on intrusions, supplementing firewalls by extensively scrutinizing packet contents for malicious activities. While firewalls concentrate on control rules and traffic headers, IDS scans packet contents for possible threats, making it more time-consuming yet necessary for full network security. IDS is critical for quickly identifying and responding to suspicious actions, hence minimizing possible network harm **[1][5].**

Despite having comprehensive logging systems, manually distinguishing between malicious and legitimate network packets is impractical and computationally intensive. With the increasing connectivity in today's world, automated tools like Intrusion Detection Systems (IDS) are essential for monitoring and detecting attacks. The alerts generated through this system mostly fall under below listed categories **[4][5]:**

True Positives: These are warning signs that something is wrong when it is. For example, the IDS detects a packet containing malicious code, which was validated by inquiry.

True Negatives: These are indications that something is correct when it is. For example, the IDS identifies a packet as having no difficulties, when it genuinely has none.

False Positives: These are alarms that indicate that something is wrong with a packet when in fact it is correct. For example, the IDS identifies a packet as having malicious code while it
really contains legitimate code.

False negatives are warnings that something is correct when it is not. Example: The IDS finds that a packet does not include any harmful code, but it truly carries malicious code, as discovered during inspection.

Sundaram A [3], outlines six intrusion types, including attempted break-ins, masquerade attacks, security penetrations, data leakage, denial of service, and malicious use, all identified through abnormal behaviours or security breaches

The above-mentioned types of intrusions can be grouped under three different models of Intrusion detection mechanism:

Anomaly-based detection focuses on identifying abnormal behavior, often derived from historical or empirical data, which is continuously updated to include new behavior patterns.

Signature-based detection, on the other hand, relies on known signatures of malicious activities categorized into unauthorized access, modification, or denial of service. However, this model can only detect previously known attacks.

To address the limitations of both approaches, a hybrid model combining elements of both anomaly and signature-based detection is being developed. [3]

1.2 The System Intrusion Process

The attackers intrude into system through below process,

Reconnaissance - Before starting an attack, hackers conduct reconnaissance to learn about the vulnerabilities of a target system. They scan the system for vulnerabilities, a process called as vulnerability assessment that uses automated scanning to find flaws in operating systems, software, and protocols. As technology progresses, vulnerability assessment methodologies get more sophisticated, allowing attackers to successfully find and exploit system weaknesses.

Physical Intrusion - Intruders can get unauthorized access to an organization's network by scanning for information and impersonating genuine users. If security fixes are not installed, they can get special administrator privileges, low-privilege user accounts, and remote access privileges.

Denial of Service - Denial-of-service (DoS) attacks disrupt services or systems by crashing or overloading network links, CPU, or disk. Common attacks include Ping of Death, SYN flood, Land/Latierra, and WinNuke, exploiting vulnerabilities to disrupt service availability. [3]

1.3 Major setbacks of system intrusions

Loss of confidential, availability and integrity of critical data on computers can have various implications depending on the intrinsic value of the data. It's interesting that digital data loss differs from physical data loss, as precautions like reporting to the police and credit card issuers aren't always available.

The increased usage of online personal data, notably from credit and debit cards, as well as the storing of personal information by businesses and government agencies, might jeopardize privacy. Mortgage businesses, for example, might keep sensitive information, jeopardizing consumers' privacy and potentially harming their networks.

If the organization's network contains customer information, they may be held liable for damages caused by hackers. In two-level hacking, where hackers gain access to one network and target another, can result in liability [3].

1.4 Challenges in Anomaly Detection during Polymorphic Attack

Today, the misuse of the Internet is widespread, evident in various forms of malicious activities present in network traffic, including worms, port scans, and denial of service attacks. These anomalies not only consume network resources but also pose security threats to all Internet users by causing performance degradation and potential breaches. Consequently, accurately detecting such anomalies has become a pressing challenge for analysts. Anomaly detection involves identifying patterns in data that deviate from expected behavior, termed anomalies outliers, or aberrations [6].

Despite advancements in Intrusion Detection Systems (IDS), the anomaly detection during a polymorphic attack remains a challenge, leaving networks vulnerable to sophisticated cyber threats. Polymorphic attacks continuously mutate their characteristics, making them difficult for traditional IDS to identify accurately. While IDS are effective in detecting known patterns of Distributed Denial of Service (DDoS) or Denial of Service (DoS) attacks, they struggle to adapt to the evolving nature of these polymorphic attacks. Current detection methods often rely on analysing a predefined set of traffic features, including traffic volume, packet attributes, flow patterns, rate of traffic, behavioural analysis, protocol behaviour, and anomaly detection. However, these methods are insufficient in capturing the dynamic variations introduced by polymorphic attacks, leading to false negatives, and leaving networks susceptible to exploitation. As a result, there is a pressing need to enhance IDS capabilities to effectively detect and mitigate polymorphic attacks by developing more robust and adaptive detection methodologies.

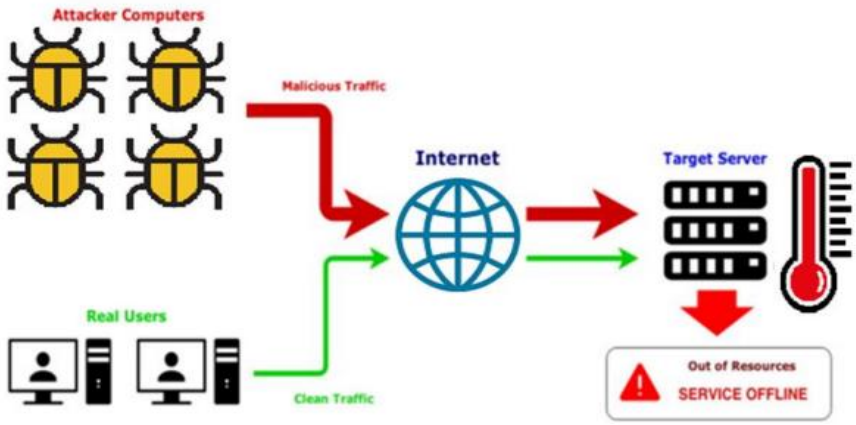


Figure 1. Malicious and clean traffic

Figure 1: Malicious & Clean traffic

[2] Figure 1. Malicious and Clean traffic, Rasheed, M. M., Faieq, A. K., & Hashim, A. A. (2021). Development of a new system to detect denial of service attack using machine learning classification. *Indonesian Journal of Electrical Engineering and Computer Science*, *23*(2), 1068-1072.

1.5 DDoS Attack: Range and Types

In the research survey [7] on DDoS attacks, the findings indicate that these attacks are difficult to counter and trace because of their distributed nature and the use of spoofed IP addresses, which conceal the attacker's identity. . Additionally, many internet hosts have security vulnerabilities that can be exploited, and attacks targeting the application layer are rapidly increasing. Hence, understanding all aspects of DDoS attacks is crucial for deploying effective IDS. It classifies the DDoS flooding attacks based on the protocol level they target, mainly network/transport-level and application-level **[7]**.

1.5.1 Network/Transport-Level DDoS Flooding Attacks

These attacks, often using TCP, UDP, ICMP, and DNS protocols, can be categorized into **[7]**:

- Flooding Attacks: Disrupt connectivity by exhausting network bandwidth (e.g., UDP flood, ICMP flood).
- Protocol Exploitation Flooding Attacks: Exploit protocol features or bugs to consume resources (e.g., TCP SYN flood).
- Reflection-Based Flooding Attacks: Send forged requests to reflectors, which then overload the victim (e.g., Smurf attacks).
- Amplification-Based Flooding Attacks: Use services to amplify traffic toward the victim, often combined with reflection techniques (e.g., Botnets).

1.5.2 Application-Level DDoS Flooding Attacks

These attacks exhaust server resources by mimicking legitimate traffic, making them stealthier. Common types include **[7]**:

1. Reflection/Amplification-Based Attacks: Like network-level, but using application protocols (e.g., VoIP flooding).

2. HTTP Flooding Attacks: Include session flooding, request flooding, asymmetric attacks, and slow request/response attacks. Examples are:

- Session Flooding: Overwhelms servers with high connection request rates (e.g., HTTP GET/POST flood).
- Request Flooding: Sends multiple requests within a single session to bypass defenses.
- Asymmetric Attacks: High-workload requests that are hard to detect.
- Slow Request/Response Attacks: Hold connections open for long periods, exhausting server resources (e.g., Slowloris, Slowpost).

DDoS attacks are multidimensional, requiring preparation to detect both aggressive and stealthy attacks to protect systems effectively **[7]**

2. Problem Statement

The standard approach for improving Intrusion Detection Systems (IDS) involves analyzing attack traffic data and updating the IDS with the identified attack features. This enables the IDS to recognize the attack when it occurs. However, this approach struggles with detecting polymorphic attacks, where attackers continuously modify their attack features to evade detection.

In their research work **[8]**, the authors highlight the complexities of detecting network attacks that deviate from typical patterns, including novel (unknown), unusual (atypical), and continuously mutating (polymorphic) attacks, discussing the limitations of existing methods. Understanding these nuances is crucial for developing robust intrusion detection systems capable of handling dynamic cyber threats.

To tackle the challenge of detecting polymorphic attacks, it is essential to analyze and correlate a diverse range of features from attack flows to identify potential indicators. This research aims to enhance the detection of polymorphic attacks, such as Denial of Service (DoS) attacks, by scrutinizing the network traffic they generate. Utilizing brute force techniques with a DoS attack tool and various arguments, the study will uncover critical features and their interrelationships, offering deeper insights into these attacks. By extracting 82 different parameters using a feature extraction tool, we can effectively monitor both benign and malicious traffic. The research will focus on examining one feature at a time that could be affected by the attack, analyzing change in one feature value impacts other important parameters across iterations. This thorough analysis can contribute to a better understanding of polymorphic attack characteristics, ultimately improving the Intrusion Detection System's ability to detect such attacks.

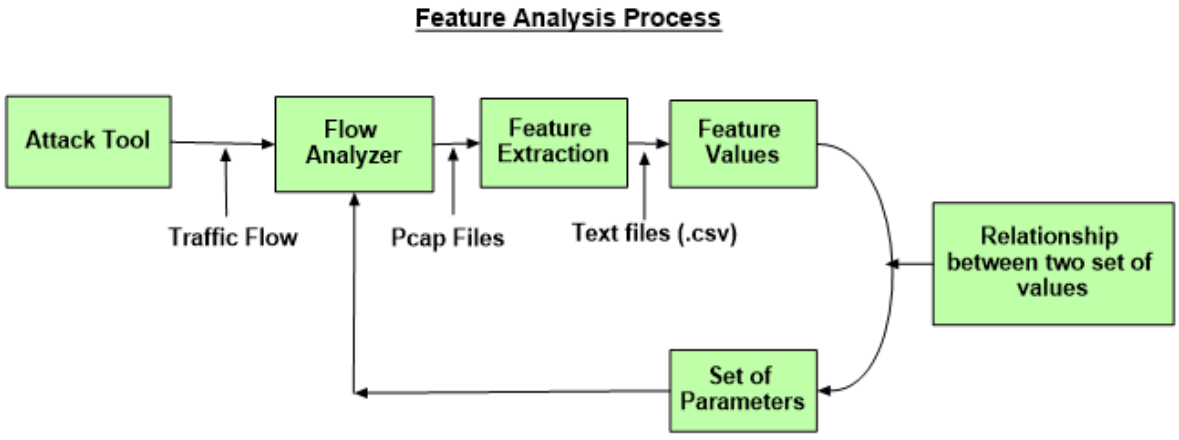


Figure 2: Feature Analysis

The primary goal of the research is to establish a systematic and automated approach for analyzing and optimizing network traffic generated by polymorphic attack tools. This involves leveraging machine learning to gain insights and improve the tool's effectiveness through an iterative feedback loop.

We assume the attack tool has a set of parameters designated as P´ and is related to a subset F´ of all network features extracted from the traffic

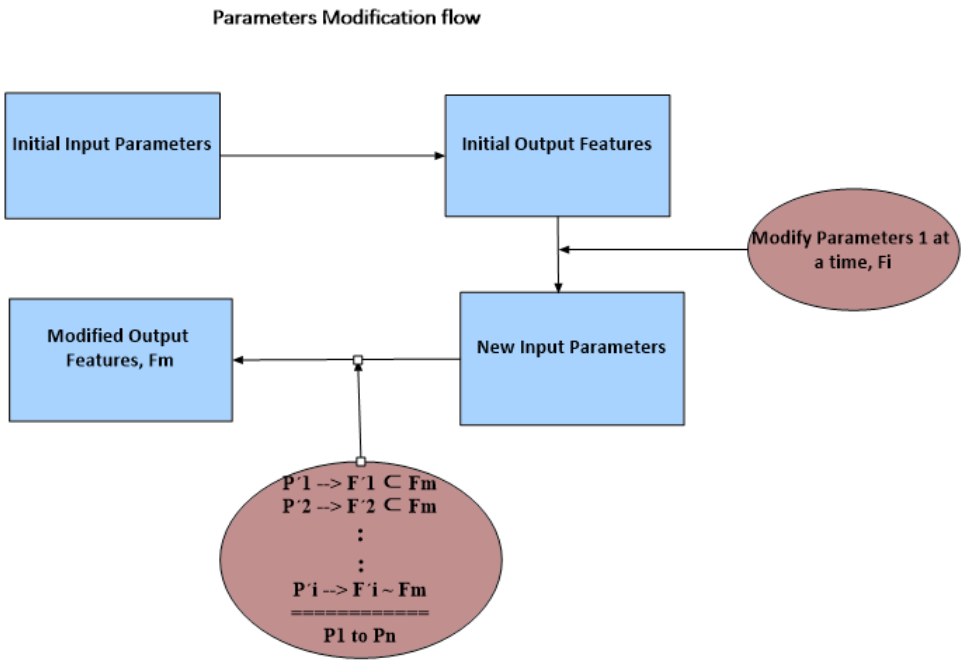


Figure 3:Parameters Modifications

The process to conduct the feature analysis involves the following steps:

1. Generation of Polymorphic Attacks: A polymorphic attack tool, such as Slowloris, is utilized to create diverse patterns of network traffic. This tool is configured with a specific set of parameters that dictate its behavior and attack methods.

2. Traffic Capture: The generated traffic is captured and forwarded to a flow analyzer for detailed examination and analysis.

3. Flow Analysis: The flow analyzer processes the incoming traffic data, extracting relevant metrics and information. The output is recorded in an Excel file for further processing.

4. Data Extraction: From the Excel file, key features and values are identified and extracted. These features represent significant attributes of the traffic data that can be used for deeper analysis.

5. Machine Learning Analysis: A machine learning model is applied to the extracted features and values to identify patterns and relationships between different sets of values. This analysis helps in understanding how various parameters affect the behavior of the polymorphic attack tool and the resulting traffic.

6. Optimization of Parameter Set: Based on insights from the machine learning analysis, the parameter set of the attack tool is adjusted and optimized. This iterative process helps refine the tool's configuration to achieve desired outcomes.

7. Feedback Loop: The optimized parameter set is fed back into the polymorphic attack tool, and the process repeats, enabling continuous improvement and fine-tuning of the tool’s performance and the traffic analysis process.

This version emphasizes the use of polymorphic attack tools and the iterative nature of optimizing their parameters for improved traffic analysis.

3 Related Work:

3.1 Detection Methods

This section of research highlights the existing related works in detection methods of different polymorphic attacks and improving the IDS. Most of the current research works use either ML/DL, Statistical or Algorithm methods on datasets made available to the public for research purpose.

In **[9],** the authors propose a method for detecting multi-class distributed reflection denial of service (DRDoS) attacks. Their methodology involves analyzing attack behavior, selecting a feature subset of 24 relevant features, constructing an XGBoost model, and evaluating its performance. Performance evaluation is conducted using metrics such as accuracy, precision, recall, and F1-score. The research paper faces challenges in feature selection complexity, feature engineering from network traffic data, handling imbalanced data, selecting, and optimizing machine learning algorithms, and choosing appropriate evaluation metrics, thus highlighting the complexity of detecting DRDoS attacks accurately and efficiently.

The study **[10]** addresses the increasing threat of DDoS attacks, and propose a method that leverages natural selection(biological) principles to dynamically select both features and machine learning models for DDoS attack detection. By adapting to changes in attack patterns and network conditions, the proposed method aims to improve detection accuracy and adaptability. Inspired by genetic algorithms, the process iteratively refines the feature set and model ensemble, adapting to changes in attack patterns and network conditions.

It **[11]** introduces a novel method for detecting Distributed Denial of Service (DDoS) attacks, leveraging active and idle features extracted from a revised version of the CICFlowMeter. By analyzing both active (during an attack) and idle (normal) states of network traffic, the method aims to identify anomalous patterns indicative of DDoS attacks. The paper contributes to the advancement of DDoS attack detection techniques by combining flow-based measurement with statistical analysis, providing insights into improving network security and resilience against DDoS attacks.

The study **[12]** compares different AI methods for spotting overload attacks and finds that all three methods tested (RNN, LSTM, GRU) work very well. All three were almost perfect (99% accuracy) at finding attacks in a recent dataset, GRU being the fastest. The study **[13]** compares different Machine Learning (ML) techniques for finding these DDoS attacks. It finds that some methods, like Natural Gradient Boosting and Convolutional Neural Networks, work particularly well on data organized in tables. This research provides valuable insights into protecting networks from DDoS attacks and helps choose the best approach for SDN.

Generative Adversarial Networks (GAN) as proposed in **[14],** are considered effective at creating traffic that mimics legitimate data. This study explores ML/DL-based DDoS detection and counter-detection techniques using Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Network (RNN) that learns long-term dependencies, achieving high accuracy. However, testing against GAN-generated adversarial attacks showed the LSTM method's inefficiency and then was further improved for more accurate results.

The paper **[15]** aims to detect and mitigate both known and unknown DDoS attacks in real-time environments. To achieve this, an Artificial Neural Network (ANN) algorithm is employed to identify DDoS attacks by distinguishing specific patterns that differentiate attack traffic from legitimate traffic.

The research **[16]** explores a new system that combines different machine learning (ML) techniques for better DDoS detection. The system mixes supervised (trained with labeled data) and unsupervised (learns patterns without labels) methods, including clustering (K-Means), classification (SVM), and deep learning. By combining these techniques, the system can more accurately spot malicious traffic and reduce false alarms, making DDoS detection overall more effective.

The above-mentioned research works discusses various methods to detect DDoS attacks and help in improving IDS. The method of generating attack and collecting datasets for feature analysis is also an important aspect of this research work.

3.2 Feature Engineering

Azam, **[17]** mentions that feature selection reduces computational complexity, removes redundant data, enhances machine learning accuracy, simplifies datasets, and lowers false alarm rates, aiding lightweight intrusion detection systems (IDSs). With the rise in data generation across various fields, feature engineering has emerged as a solution to the curse of dimensionality, managing high-dimensional data and improving classification performance through refined data representations. Numerous feature selection algorithms exist, categorized into filter-based, wrapper, embedded, and hybrid methods.

Feature extraction reduces data attributes by mapping high-dimensional features to a lower-dimensional space, enhancing model performance but altering feature meaning **[18]**, **[19].** In contrast, feature selection retains the original meaning by choosing the most relevant features **[20].** Both methods improve learning efficiency, reduce computational costs, and mitigate overfitting. **Feature selection**, an NP-hard problem, involves selecting, evaluating, and validating optimal feature subsets based on criteria like distance and consistency **[21], [22], [23], [24].**

In **[25],** the research work proposes a feature selection-based approach for DDoS attack flow classification, emphasizing the importance of identifying relevant features for accurate detection at both packet and flow levels. They experiment with various deep

learning models and find that Long Short-Term Memory (LSTM) performs well without overly complex models. However, applying deep learning to flow datasets poses challenges due to abstraction differences between packets and flows. This paper **[26]** introduced a heuristic method for detecting DoS attacks by reducing feature vector size using Information Gain (IG), Gain Ratio (GR), and Correlation (CR). IG measures entropy, GR reduces bias in multivalued attributes, and CR evaluates attributes value by value. Features are ranked and divided into seven subsets, which are processed using the C4.5 classification algorithm. If the accuracy of subset Fi surpasses the previous subset Fi-1, it is included in the reduced feature set (RF). This method is applied to all feature selection techniques to achieve the results. This method is performed on KDD datasets.

[27] mentions that although Machine learning is used to fight these attacks but picking the right data points (features) is even more important than the machine learning model itself. This research proposes a new approach that groups similar network traffic together to create a more abstract view of the data. This allows the models to better identify attacks that mimic normal behavior. The study tests this approach on a standard dataset (CICIDS2017) and shows it works well. This new method can improve attack detection and open doors for better ways to find features for complex attacks.

In **[28]** flow-based intrusion detection offers an innovative approach for identifying intrusions in high-speed networks by inspecting only the packet header and not the packet payload. It discusses the available flow-based datasets used to evaluate these systems and proposes a taxonomy based on the techniques used to detect malicious activity in flow records.

The paper **[29]** focuses on application layer DDoS and compares with the existing solutions for application layer DDoS detection which either have limited coverage or high computational complexity. To address these limitations, it proposes a deep learning-based approach using Stacked AutoEncoders to learn deep features of application layer DDoS attacks. From the datasets generated in their lab 8 features are extracted – Source IP Address, TimeStamp, TimeZone, URL, response code, number of bytes send, refer page used by the user, browsing information of the user.

Jazi, **[30]** research proposes a new detection method using a nonparametric CUSUM algorithm and evaluates its effectiveness against different attack types. It analyzed various network and application-level features to detect DoS attacks, focusing on attributes like HTTP requests/responses, connection counts, and packet payload sizes. The detection algorithm primarily used two features: the number of application-layer requests and packets with zero payload size. It calculated a detection ratio R^ to capture the variability of these attacks. However, common sampling techniques used in production environments can significantly hinder the detection capabilities of this approach.

Tang [31] introduces the SADBSCAN algorithm, an adaptation of the DBSCAN algorithm optimized for detecting low-rate DoS attacks. It dynamically adjusts its parameters to identify anomalies in network traffic by clustering data points based on density, effectively

distinguishing between legitimate low-rate traffic and malicious DoS activity. Utilizing features like packet arrival rate and packet size distribution, the algorithm identifies sparse clusters indicative of potential attacks. Experimental results demonstrate SADBSCAN's high accuracy and low false positive rates, proving its robustness in detecting low-rate DoS attacks. However, this method is quite complex and requires signification efforts especially in complex networks.

Liao **[32],** study focuses on differentiating user behaviors by analyzing web logs, proposing features that represent user characteristics and transforming these logs into a 14-dimensional feature space. The goal is to achieve better representations of user behavior and investigate the differences and similarities between DDoS attackers and normal users. Features like – Source address, request times, total length in bytes of a user request, session duration, sequence Of Request Frequency & Interval were considered during analysis to effectively distinguish between legitimate users and attackers at the application layer.

Abushwereb **[33],** delve into the critical issue of Denial of Service (DoS) flooding attacks, which have inflicted substantial economic losses in recent years. To counter these attacks, the authors investigate the efficacy of machine learning classification algorithms that utilize features extracted from an SNMP-MIB dataset obtained from a real-world testbed. Multiple classifiers enhance the effectiveness of DoS attack detection by leveraging diverse perspectives, ensemble techniques, and adaptability. If one classifier misses an attack, others may catch it. Ensemble techniques, such as voting, stacking, or bagging, mitigate individual classifier biases and errors, resulting in more accurate and robust decisions. Additionally, multiple classifiers adapt to changing attack patterns, recognizing new threats early and improving overall detection performance. The paper provides valuable insights for enhancing network security against evolving threats.

The authors **[34]** propose two novel feature selection techniques: RF-FSR (RandomForest-Forward Selection Ranking) and RF-BER (RandomForest-Backward Elimination Ranking). These methods optimize feature selection by ranking features based on relevance and gradually eliminating less relevant features, respectively. Additionally, the authors introduce a hybrid attack detection model called SABADT (Signature- and Anomaly-Based Attack Detection Technique), which combines signature-based and anomaly-based approaches to enhance intrusion detection accuracy and robustness. Their work contributes valuable insights to combat evolving cyber threats.

Estimating statistical properties **[35]** of network traffic, specifically entropy variations in packet header fields, is a common approach for detecting DDoS attacks. These methods gained popularity in the mid-2000s due to their effectiveness in identifying irregularities during volumetric DDoS attacks. Feinstein as cited in Patil **[35]** developed a detection method based on source IP address entropy and Chi-square distribution and their results showed that entropy changes were smaller during normal peak hour traffic bottlenecks compared to DDoS attacks. However, entropy is not always a reliable measure due to high false positives or negatives. Choosing an appropriate threshold is another challenging for these approaches.

3.3 Common Datasets Vs Synthetic Datasets

Cordero **[36]**, discussed the challenges in obtaining dependable datasets for assessing Network Intrusion Detection Systems (NIDSs). Commonly accessible datasets are typically outdated, deficient in labelled attacks, and may contain unnoticed flaws. Moreover, these datasets are often not publicly accessible or are challenging to acquire. Most datasets are provided in the form of Packet Capture (PCAP) files, which essentially consist of network packets originating from either benign or malicious sources.

The author in **[36]**, further elaborates on the requirements of suitable datasets for the evaluation of IDS. These requirements can be categorized as functional requirements or non-functional requirements. Functional requirements include payload availability, datasets with accurately labelled attacks, flexibility in context of ability to test different scenarios and continuously renewable. The non-Functional requirements consist of the possibility to replicate datasets or make them publicly available; interoperability – using a common format like PCAP file format & a less biased datasets.

Synthetic datasets can be created by brute force attacks or by injecting synthetic attacks into background traffic. Rather than merely merging background and attack traffic, synthetic attacks attempt to replicate the features of background traffic inside synthetic attack traffic. The extent to which these properties are replicated can be adjusted by the user to align with their specific requirements **[36]**.

3.4 Research Gap

The common approach researchers opt while improving the IDS, is using the public datasets like – KDD-Cup1999, CIC-IDS2017 and more. These datasets are generated on test environments which are not accessible by researchers. Also, the focus in on finding solutions for Detecting and Preventing attack. These solutions can be based on ML, Statistical or Algorithm models. Further, the research works on polymorphic DDoS and DoS attacks are based on studying and analysing some of the traffic features which can lead to better performance of IDS. However, there is less focus on understanding the data generated by various traffic flow features generated by these attacks and what set of features can be impacted by them.

Through my research project I aim at finding the correlation between the parameters of attack tool with the features generated from that traffic flow which in the lab test bed using an improved feature extraction tool – Lycostand.

4. Project Plan & Methodology

4.1 Project Objective

This project focuse on creating a realistic polymorphic attack in the test lab and work on finding the relationship between polymorphic attack features whenever the attack tool parameters are changed. To help in establishing and analysing the relationship a DoS attack like Slowloris is generated in the lab set up. The Lycostand tool then analyses the set of features generated by attack.

An initial test is run with basic arguments and considered as base values. Then, changing some of those parameters one at a time multiple iterations are run in the lab to collect the data. These changes do not impact the **attack features** hence keeping it a **valid attack** and not evading it from being detected as an attack. Hence, this will help to create a list or set of attack features that is generated by traffic flow, indicating which input parameters of Slowloris need to be adjusted to modify a feature

4.2 Project Implementation

In this project we achieve our objective of improving IDS anomaly detection from polymorphic attacks by –

- First, we create traffic using an attack tool like Slowloris or httptest,
- Traffic is captured using Wireshark and saved as PCAP files,
- Then we feed these files into Lycostand,
- The Lycostand tool extracts the set of features generated from the attack.

Lycostand is an alternative tool to CICflowmeter, which can help in improving the Feature extraction of the traffic data. Below is the list of 82 features of from Lycostand, for this project I will be looking at some of the features.

✓ List of features from Lycost https://github.com/17204612710214/lycos-ids2017/blob/main/LycoSTand_features.pdf

4.2.1 Set up of Virtual Environment

Oracle VM VirtualBox Manager

Oracle VM VirtualBox is utilized for installing a Linux Server, aiming to establish two distinct environments within VirtualBox while implementing DoS attack protection on the server. The objective is to create two servers to mimic an attack scenario and generate feature file when execute a DoS attack on an Apache web server. Through VirtualBox's capability to run multiple operating systems on a single host, this simulated attack operates within an emulated environment. **[11]**

Steps to Install Oracle VM VirtualBox Manager:

Download VirtualBox:

- ✓ Visit the official Oracle VM VirtualBox website (https://www.virtualbox.org/) and navigate to the Downloads section.
- ✓ Download the appropriate installer for your operating system (Windows, macOS, Linux, or Solaris).

Install VirtualBox:

- ✓ Run the downloaded installer and follow the on-screen instructions to install VirtualBox on your system.

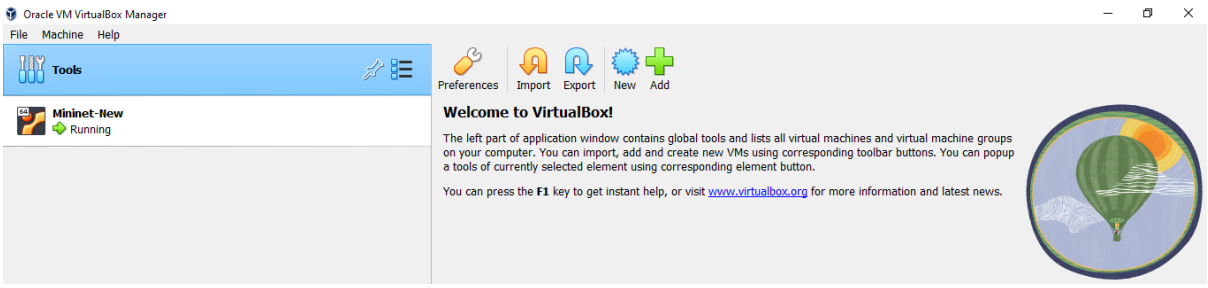


Figure 4: Oracle VM Virtualbox Manager

Kali Linux

It is a specialized Linux distribution designed for penetration testing, ethical hacking, and security assessments. It comes with a wide range of security tools and utilities pre-installed, making it a popular choice among cybersecurity professionals and enthusiasts.

Figure 5: Kali Linux Desktop Version

To install Kali Linux on Oracle VirtualBox, follow these steps:

1. **Download Kali Linux ISO Image**:
    - Go to the official Kali Linux downloads page and download the Kali Linux ISO image.
    - Choose the appropriate version (64-bit) based on the system architecture.
2. **Create a New Virtual Machine in VirtualBox**:
    - Open VirtualBox and click on "New" to create a new virtual machine.
    - Provide a name for your virtual machine (CRP_KaliVM).
    - Choose "Linux" as the type and "Debian (64-bit)" as the version.
    - Allocate memory (RAM) to the virtual machine (at least 2 GB is recommended).
    - Create a new virtual hard disk (VDI or VHD) with sufficient storage space (around 20 GB or more).
3. **Configure Virtual Machine Settings**:
    - Select your newly created virtual machine and click on "Settings."
    - In the "Storage" section, add the Kali Linux ISO image as a bootable CD/DVD.
    - Adjust other settings as needed (e.g., network adapter, display, etc.).
4. **Install Kali Linux**:
    - Start the virtual machine.
    - When prompted, select the "Graphical Install" option.
    - Follow the installation wizard:
        - Choose your language, location, and keyboard layout.
        - Set the hostname and domain name.
        - Create a user account and set a password.
        - Partition the disk (use the guided option or manual partitioning).
        - Confirm the installation and wait for it to complete.
5. **Post-Installation Configuration**:
    - After installation, reboot the virtual machine.
    - Log in using the credentials set during installation.
    - Update Kali Linux using the following commands:
        - `sudo apt update`
        - `sudo apt upgrade`
6. **Optional: Install VirtualBox Guest Additions**:
    - To improve integration between the host and guest OS, installed VirtualBox Guest Additions.

Kali Linux is now installed on Oracle VirtualBox.

Figure 6: Kali VM installed on VirtualBox

Figure 7: Kali VM Log-in page on VirtualBox

Wireshark
Is a network protocol analysis tool developed by Gerald Combs in 1998 and renamed Wireshark in 2006, became famous owing to its excellent packet analysis features.

Important Features:

- Packet Capture: Wireshark can capture live network traffic from various interfaces, including Ethernet, Wi-Fi, and loopback.

- Protocol Support: It supports a wide range of protocols, including Ethernet, IP, TCP, UDP, HTTP, DNS, SSL, and many others, making it versatile for analyzing diverse network environments.
- Packet Analysis: Wireshark provides detailed packet inspection capabilities, allowing users to analyze packet headers and payloads, as well as decode protocols to understand network communication.
- Filtering and Search: Users can apply complex filters and search criteria to isolate specific packets or types of traffic, making it easier to focus on relevant data during analysis.
- Statistics: Wireshark offers various statistics features, including packet summary, endpoint conversations, protocol hierarchy, and traffic graphs, enabling users to gain insights into network performance and behavior.
- Export and Save: Users can export captured packets in various formats, including pcap, CSV, JSON, and plaintext, for further analysis or sharing with others.
- Customization: Wireshark is highly customizable, with options to customize packet decoding, display preferences, and coloring rules to suit individual preferences or specific analysis requirements.

Wireshark is a powerful and flexible tool for network troubleshooting, protocol analysis, security auditing, and educational purposes, making it indispensable for network professionals and researchers alike. In this project, Wireshark will help capture traffic features during an attack.

Snort Tool

A widely used open-source network intrusion detection system (IDS) and intrusion prevention system (IPS) developed by Sourcefire. It can perform real-time traffic analysis and packet logging on IP networks. Snort is known for its robust detection capabilities, flexibility, and extensive rule-based language, making it a popular choice for network security monitoring and threat detection.

Main Features of Snort as an IDS:

- Packet Sniffing: Snort can capture and analyze network traffic in real-time, allowing it to detect suspicious or malicious activities on the network.
- Rule-Based Detection: Snort uses a rule-based language to define detection criteria for identifying various types of network attacks, such as port scans, denial-of-service (DoS) attacks, and malware activity.
- Protocol Analysis: Snort can perform deep packet inspection to analyze network protocols at the application layer, enabling it to detect anomalies and suspicious behavior.
- Flexible Alerting: Snort can generate alerts in various formats, including syslog, email, and text logs, to notify administrators of detected security events.
- Logging and Reporting: Snort can log detected events to disk for further analysis and reporting, helping administrators track and investigate security incidents.

- Community Rules: Snort benefits from a large community of users who contribute to the development of detection rules, providing access to a comprehensive set of rules for detecting known threats.
- Customization: Snort can be customized and extended through the use of custom rules, pre-processors, and plug-ins to adapt to specific network environments and security requirements.

Steps to Install Snort on Kali VM

1. Install or Update Package Repository: sudo apt update or sudo apt install snort

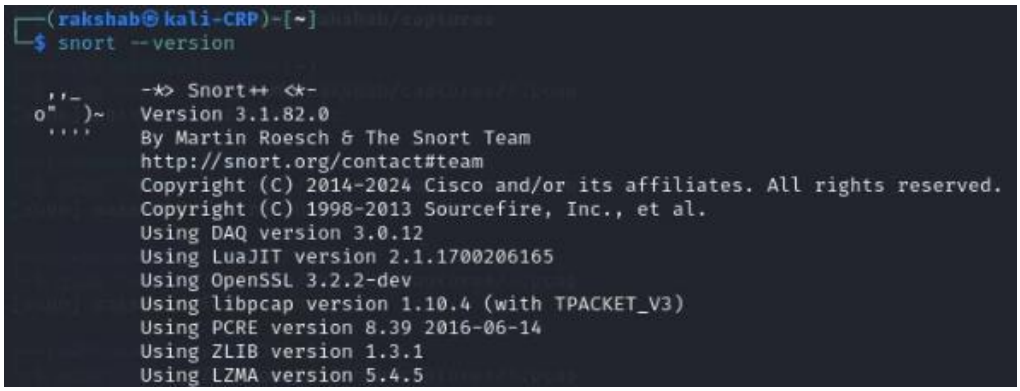


Figure 8: Snort Version

2. Configure Snort:

- Snort configuration files are in `/etc/snort/`. We can edit these files to customize Snort's behavior according to our network environment and security policies.
- The main configuration file is ` /etc/snort/snort.lua`, where we can specify network interfaces, rule sets, logging options, and alerting mechanisms.

3. Start Snort: “sudo snort -c /usr/local/etc/snort/snort.lua -T

This command specifies the configuration file (-c) and runs Snort in test mode (-T) to check for any configuration issues without actually running the IDS.

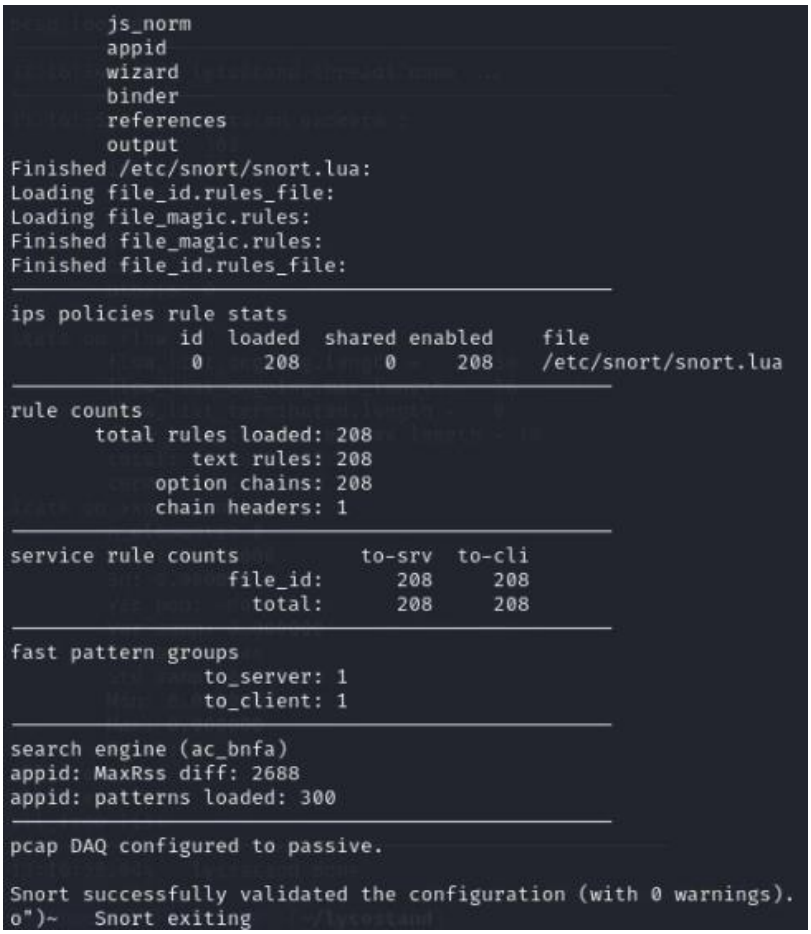


Figure 9:Snort alerts.

4. Verify Snort Installation: sudo snort -c /etc/snort/snort.lua -i eth0

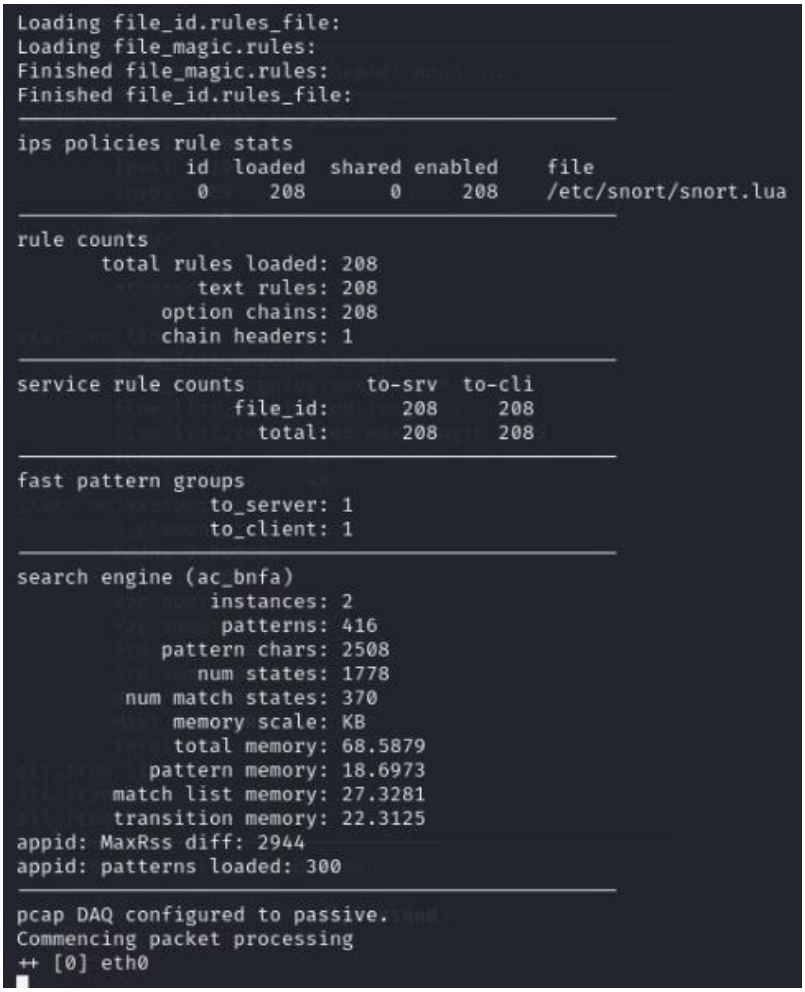


Figure 10:Snort Validation

5. Verify that Snort is running by checking its process status: "sudo systemctl status snort"

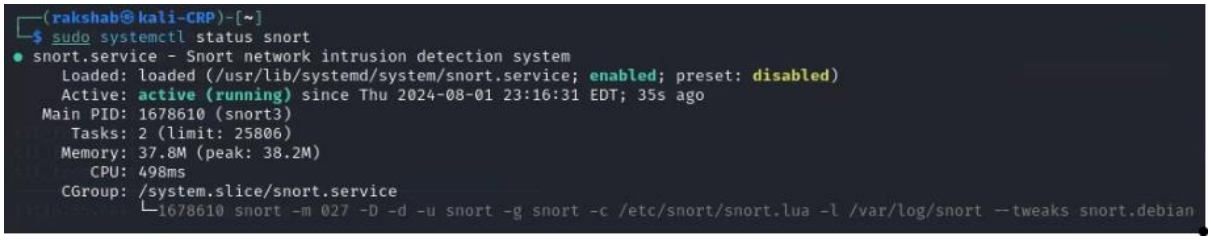


Figure 11: Snort status

Generate an Attack:

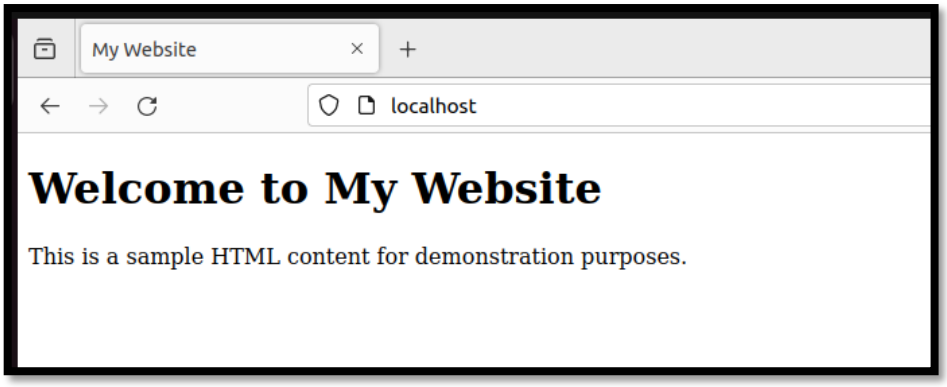


Figure 12: Localhost

Generated a sample Slowloris attack for testing purpose using the below command – "**slowloris <localhost> -s 800"**

It slows down the response of the server.

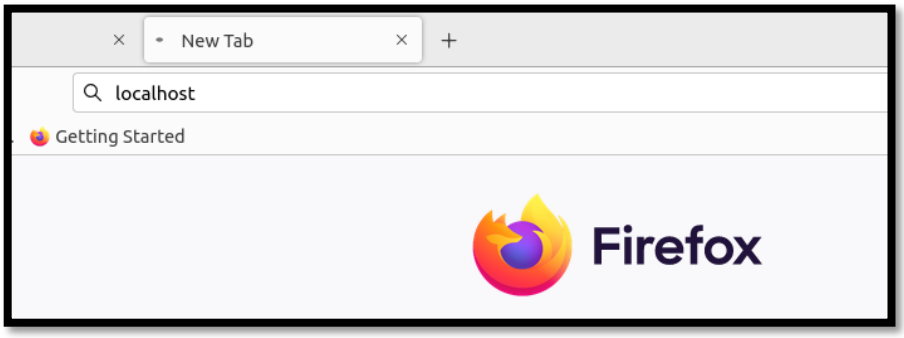


Figure 13: Localhost slow response.

5. Feature Selection

Lycostand Feature Extraction Tool
In the research paper **[37]** introduces LycoSTand as a new flow-based feature extraction tool based on C program. It relies on libpcap library to parse Ethernet packets and requires gcc compiler. The program takes the PCAP files, analyzes the packets and calculates 82 flow-based features.

It creates one CSV file containing all the features for each PCAP file. Here are the traffic features typically extracted by Lycostand and their meanings:

- **flow_duration**: The total time duration of a flow.
- **fwd_pkt_cnt**: The total number of packets sent in the forward direction.
- **bwd_pkt_cnt**: The total number of packets sent in the backward direction.
- **fwd_pkt_len_tot**: The sum of all lengths of packets sent in the forward direction.
- **bwd_pkt_len_tot**: The sum of all lengths of packets sent in the backward direction.

- **fwd_pkt_len_max**: The maximum length of a packet sent in the forward direction.
- **fwd_pkt_len_min**: The minimum length of a packet sent in the forward direction.
- **fwd_pkt_len_mean**: The mean length of packets sent in the forward direction.
- **fwd_pkt_len_std**: The standard deviation of packet lengths in the forward direction.

- **bwd_pkt_len_max**: The maximum length of a packet sent in the backward direction.
- **bwd_pkt_len_min**: The minimum length of a packet sent in the backward direction.
- **bwd_pkt_len_mean**: The mean length of packets sent in the backward direction.
- **bwd_pkt_len_std**: The standard deviation of packet lengths in the backward direction.

- **bytes_per_s**: The number of bytes per second for the flow.
- **pkt_per_s**: The number of packets per second for the flow.

- **iat_mean**: The mean inter-arrival time between packets in the flow.
- **iat_std**: The standard deviation of the inter-arrival time between packets in the flow.
- **iat_max**: The maximum inter-arrival time between packets in the flow.
- **iat_min**: The minimum inter-arrival time between packets in the flow.

- **fwd_iat_mean**: The mean inter-arrival time between packets in the forward direction.
- **fwd_iat_std**: The standard deviation of the inter-arrival time between packets in the forward direction.
- **fwd_iat_max**: The maximum inter-arrival time between packets in the forward direction.
- **fwd_iat_min**: The minimum inter-arrival time between packets in the forward direction.

- **bwd_iat_mean**: The mean inter-arrival time between packets in the backward direction.
- **bwd_iat_std**: The standard deviation of the inter-arrival time between packets in the backward direction.

- **bwd_iat_max**: The maximum inter-arrival time between packets in the backward direction.
- **bwd_iat_min**: The minimum inter-arrival time between packets in the backward direction.

- **fwd_flag_psh**: The number of times the PSH flag was set in packets traveling in the forward direction.
- **bwd_flag_psh**: The number of times the PSH flag was set in packets traveling in the backward direction.
- **fwd_flag_urg**: The number of times the URG flag was set in packets traveling in the forward direction.
- **bwd_flag_urg**: The number of times the URG flag was set in packets traveling in the backward direction.

- **fwd_pkt_hdr_len_tot**: The total bytes used for headers in the forward direction.
- **bwd_pkt_hdr_len_tot**: The total bytes used for headers in the backward direction.

These features help in characterizing network traffic and are used in network intrusion detection systems to identify suspicious or malicious activities.

Challenges with CICflowmeter addressed in Lycostand tool
The developers of Lycostand tool [37] list the challenges with CICflowmeter and their new tool can help in the improvement of feature extraction from the traffic flow captured by PCAP files

- CICflowmeter has 3 duplicated features in the feature extraction list

First Duplicate value - **Pkt Len Mean = Pkt Size Avg**; the second duplicate value is **Fwd Pkt Len Mean = Fwd Seg Size Avg** & the third duplicate value - **Bwd Pkt Len Mean = Bwd Seg Size Avg.**

- CICflowmeter has 21 (excludes above duplicates) mistakes in feature calculations

11 Features – 6 related to bulks, 4 related to subflows and 1 with down link/up link ratio are affected due to a common issue with calculation “type”, since CICflowmeter uses integer division instead of floating-point division.

6 bulk-related features – irrespective of the packet direction, backward bulk features are updated with each new packet but the forward bulk features are never updated. Hence this results errors in values of all the six features.

3 Features of packet length - the mean, standard deviation and total are effected when first packet of each flow is used to update them twice.

The subflow count incorrectly increases with each packet due to a faulty timestamp test. TCP-related features have errors, with inverted flag counts, PSH and URG counts are not

updated, and the backward TCP window size showing the last value instead of the initial one.

6. Test Analysis

Below four distinct test modes from the Slowhttp tool are used for experimentation,

-H slow headers a.k.a. Slowloris (default): Starts slowhttptest in SlowLoris mode, sending unfinished HTTP requests.

-B slow body a.k.a R-U-Dead-Yet: Starts slowhttptest in Slow POST mode, sending unfinished HTTP message bodies.

-R range attack a.k.a Apache killer: Starts slowhttptest in Range Header mode, sending malicious Range Request header data.

-X slow read a.k.a Slow Read: Starts slowhttptest in Slow Read mode, reading HTTP responses slowly **[39]**

Start a slowloris test of http://localhost with 1000 connections, with base values which are named as itr0 (base iterations) and then change one parameters value for each type of profile to compare the results with base values.

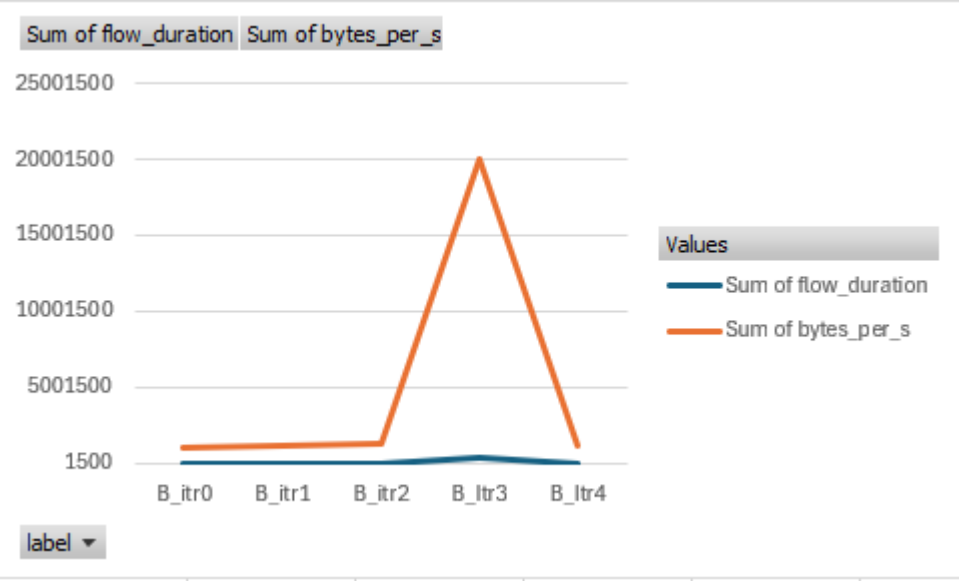
verb = "GET" # Default for Slow headers and response
verb_post = "POST" # Default for Slow body

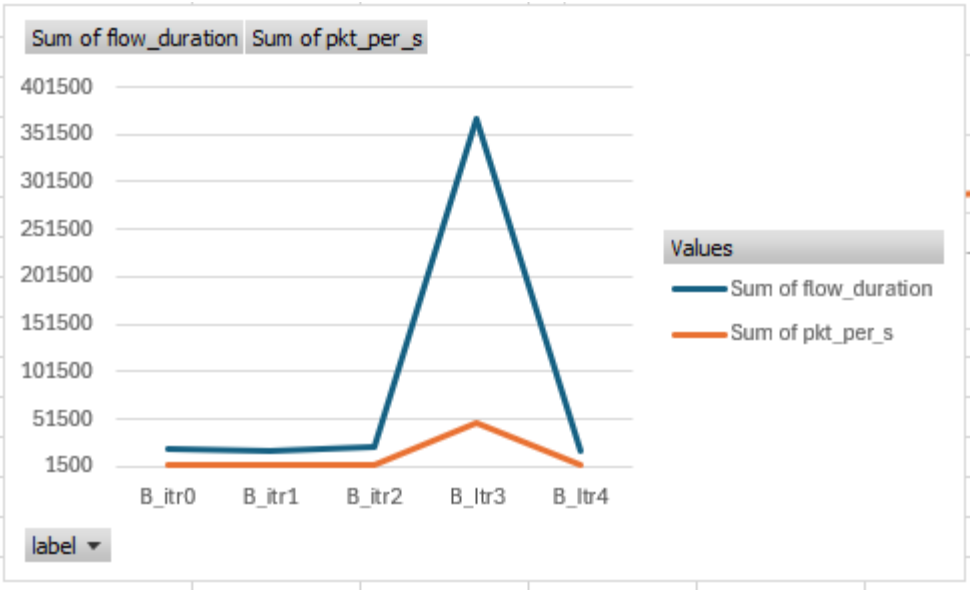
General options: connections = 1000					
Test Parameters	Itr0	Itr1	Itr2	Itr3	Itr4
Interval (-i) vlaues	5	50	100	5	5
test_length	240	240	240	2400	240
Rate	50	50	50	50	200
Content_length	4096	4096	4096	4096	4096
Max_length	2	2	2	2	2
Options specific to Range attack					
range_start	50	500	1000	2000	2500
range_limit	2000	2000	2000	2000	2000
Options specific to Slow read					
repeat_request (Max 10)	5	50(failed)	10	1	5
read_interval	1	1	1	10	1
window_start	1	1	1	1	1

window_end	512	512	512	512	512
slow_read_bytes	5	5	5	50	1

Slowloris Body attack results

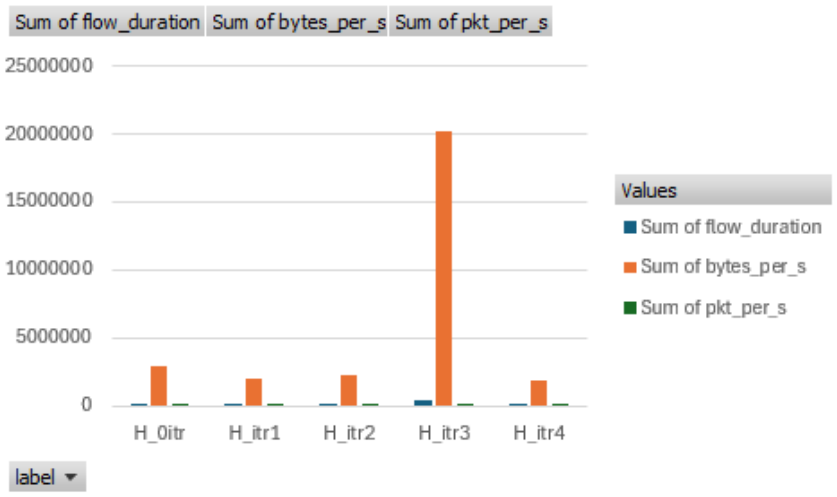
Iterations	Sum of flow_duration	Sum of bytes_per_s	Sum of pkt_per_s
B_itr0	19463	1073148.015	2573.496439
B_itr1	18489	1134198.002	2719.899284
B_itr2	22225	1356395.835	3252.747807
B_ltr3	368726	19959184.92	47863.75281
B_ltr4	18524	1127474.539	2703.775873





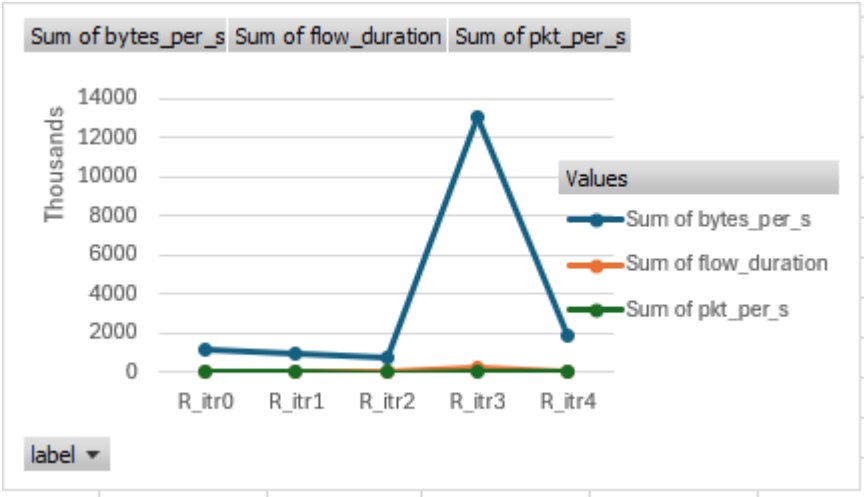
Slowhttptest Header attack results

Iterations	Sum of flow_duration	Sum of bytes_per_s	Sum of pkt_per_s
H_0itr	49274	2894110.211	6940.312256
H_itr1	34714	1955514.321	4689.482785
H_itr2	38028	2214227.811	5309.898828
H_itr3	371954	20148992.23	48318.92621
H_itr4	37245	1877772.951	4503.05264



Slowhttptest Range attack results

Iterations	Sum of bytes_per_s	Sum of flow_duration	Sum of pkt_per_s
R_itr0	1154003.831	18111	2767.395278
R_itr1	901410.1963	14848	2161.655147
R_itr2	694149.0806	10826	1664.626093
R_itr3	13064456.33	234146	31329.63148
R_itr4	1836534.672	37810	4404.159884



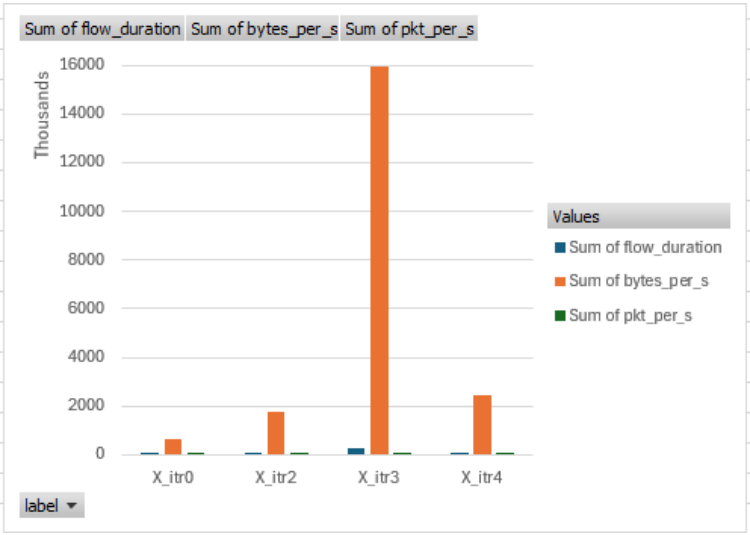
Slowhttptest Slow Read attack results

for itr1 when range was given beyond 10 it failed and resulted in below mentioned error message,

X_itr1: Error message if -k (# of times to repeat the same request in the connection) >10

```
Starting Wireshark:  sudo tshark -i eth0 -w /home/rakshab/captures/X_capture_1.pcap > tshark_1.log 2>&1 &
Running: slowhttptest -X -c 1000 -i 50 -l 240 -r 50 -s 4096 -t GET -u http://localhost/ -x 2 -k 50 -n 1 -w 1 -y 512 -z 5
Error: invalid - value 50, max: 10
Try 'slowhttptest -h' for more information
Starting Wireshark:  sudo tshark -i eth0 -w /home/rakshab/captures/X_capture_2.pcap > tshark_2.log 2>&1 &
Running: slowhttptest -X -c 1000 -i 50 -l 240 -r 50 -s 4096 -t GET -u http://localhost/ -x 2 -k 50 -n 1 -w 1 -y 512 -z 5
Error: invalid - value 50, max: 10
Try 'slowhttptest -h' for more information
Starting Wireshark:  sudo tshark -i eth0 -w /home/rakshab/captures/X_capture_3.pcap > tshark_3.log 2>&1 &
Running: slowhttptest -X -c 1000 -i 50 -l 240 -r 50 -s 4096 -t GET -u http://localhost/ -x 2 -k 50 -n 1 -w 1 -y 512 -z 5
Error: invalid - value 50, max: 10
```

Iterations	Sum of flow_duration	Sum of bytes_per_s	Sum of pkt_per_s
X_itr0	12524	631015.6171	1513.226899
X_itr1	Failed	Failed	Failed
X_itr2	30398	1769001.388	4242.209563
X_itr3	267661	15962007.47	38278.19538



Key Findings from test results

Profile	Arguments	Features with change in values	Features with constant values	Features with equal values
Body(-B) & Header(-H)	Interval values (-i)	flow_duration, bytes_per_s, pkt_per_s, fwd_pkt_per_s,	pkt_len_max. pkt_len_min, pkt_len_mean, pkt_len_var, pkt_len_std, bwd & fwd_pkt_len_tot, _max, _min, _mean, bwd & fwd_subflow_bytes_mean	iat_max = iat_min = iat_mean = flow_duration; bwd_pkt_per_s = fwd_pkt_per_s,
	Test length(2000)	flow_duration, **bytes_per_s, (major changes)** pkt_per_s, fwd_pkt_per_s, bwd_pkt_per_s,	pkt_len_max. pkt_len_min, pkt_len_mean, pkt_len_var, pkt_len_std, bwd & fwd_pkt_len_tot, _max, _min, _mean, bwd & fwd_subflow_bytes_mean	iat_max = iat_min = iat_mean = flow_duration; bwd_pkt_per_s = fwd_pkt_per_s,
Range(-R)	Range start values	flow_duration, bytes_per_s, pkt_per_s, fwd_pkt_per_s, bwd_pkt_per_s,	pkt_len_max. pkt_len_min, pkt_len_mean, pkt_len_var, pkt_len_std, bwd & fwd_pkt_len_tot, _max, _min, _mean, bwd & fwd_subflow_bytes_mean	iat_max = iat_min = iat_mean = flow_duration; bwd_pkt_per_s = fwd_pkt_per_s,
Slow Read(-X)	Repeat Request (Max Value 10)	flow_duration, bytes_per_s, pkt_per_s,	pkt_len_max. pkt_len_min, pkt_len_mean, pkt_len_var,	iat_max = iat_min = iat_mean = flow_duration;

		fwd_pkt_per_s, bwd_pkt_per_s,	pkt_len_std, bwd & fwd_pkt_len_tot, _max, _min, _mean, bwd & fwd_subflow_bytes_mean	bwd_pkt_per_s = fwd_pkt_per_s,
	slow_read_bytes(50)	flow_duration, **bytes_per_s,(major changes)** pkt_per_s, fwd_pkt_per_s, bwd_pkt_per_s,	pkt_len_max. pkt_len_min, pkt_len_mean, pkt_len_var, pkt_len_std, bwd & fwd_pkt_len_tot, _max, _min, _mean, bwd & fwd_subflow_bytes_mean	iat_max = iat_min = iat_mean = flow_duration; bwd_pkt_per_s = fwd_pkt_per_s,

7. **Conclusion & Future works**

Based on the tests performed the results show changes in certain traffic flow features. The network selected to perform this test is “Host-only” network hence there is an absence of normal traffic flow and it helped us to focus on the attack flow features. The method to run the attack iterations can be further improved by automating the process of converting pcap files into feature files (.csv). Further enhancement can be made on the open source, feature extraction tool LycoStand, for example converting PCAPNG files and time feature conversion. Further, the data collected can be used for training ML models. These tests can be performed using various test scenarios with more complex settings and help in detecting hidden attack features using IDS.

8. **References**

[1] Sabeel, U., Heydari, S. S., El-Khatib, K., & Elgazzar, K. (2023). Analyzing the Quality of Synthetic Adversarial Cyberattacks. *2023 19th International Conference on Network and Service Management (CNSM)*, 1–5. https://doi.org/10.23919/CNSM59352.2023.10327854

[2] M. M. Rasheed, A. K. Faieq, and A. A. Hashim, "Development of a new system to detect denial of service attack using machine learning classification," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 2, p. 1068, Aug. 2021, https://doi.org/10.11591/ijeecs.v23.i2.pp1068-1072

[3] Kizza, J.M. (2020). System Intrusion Detection and Prevention. *In: Guide to Computer Network Security. Texts in Computer Science. Springer, Cham*. https://doi-org.uproxy.library.dc-uoit.ca/10.1007/978-3-030-38141-7_13

[4] Rao, U.H., Nayak, U. (2014). Intrusion Detection and Prevention Systems. In: *The InfoSec Handbook.* Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4302-6383-8_11 (Nayak & Rao, 2014)

[5] Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019, July 17). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, *2*(1). https://doi.org/10.1186/s42400-019-0038-7

[6] Ratnaparkhi, S., Bhange, A., & Scholar, M. T. (2012). Anomaly Exposure in Network Traffic and its collision: a Survey. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 1(10)

[7] Zargar, S. T., Joshi, J., & Tipper, D. (2013). A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys and Tutorials*, *15*(4), 2046–2069. https://doi.org/10.1109/SURV.2013.031413.00127]

[8] Sabeel, U., Heydari, S. S., El-Khatib, K., & Elgazzar, K. (2024). Unknown, Atypical and Polymorphic Network Intrusion Detection: A Systematic Survey. IEEE Transactions on Network and Service Management/IEEE eTransactions on Network and Service Management, 21(1), 1190–1212. https://doi.org/10.1109/tnsm.2023.3298533

[9] Tianqi Yang, Weilin Wang, Ying Liu, & Huachun Zhou. (2021). Multi-class DRDoS Attack Detection Method Based on Feature Selection. *Research Briefs on Information & Communication Technology Evolution*, *7*, 173–187. https://doi.org/10.56801/rebicte.v7i.127

[10] Ma, R., Chen, X., & Zhai, R. (2023). A DDoS Attack Detection Method Based on Natural Selection of Features and Models. *Electronics*, 12(4), 1059. https://doi.org/10.3390/electronics12041059

[11] B. H. Ali, N. Sulaiman, S. A. R. Al-Haddad, R. Atan and S. L. M. Hassan, "DDoS Detection Using Active and Idle Features of Revised CICFlowMeter and Statistical Approaches," 2022 4th *International Conference on Advanced Science and Engineering (ICOASE)*, Zakho, Iraq, 2022, pp. 148-153, doi: 10.1109/ICOASE56293.2022.10075591.

[12] Ramzan, M., Shoaib, M., Altaf, A., Arshad, S., Iqbal, F., Castilla, N. K., & Ashraf, I. (2023). Distributed Denial of Service Attack Detection in Network Traffic Using Deep Learning Algorithm. *Sensors*, *23*(20), 8642. https://doi.org/10.3390/s23208642

[13] Raza, M. S., Mohammad Nowsin, A. S., I-Shyan Hwang, & Mohammad Syuhaimi Ab-Rahman. (2024). Feature-Selection-Based DDoS Attack Detection Using AI Algorithms. *Telecom*, *5*(2), 333. https://doi.org/10.3390/telecom5020017

[14] Mustapha, A., Khatoun, R., Zeadally, S., Chbib, F., Fadlallah, A., Fahs, W., & El Attar, A. (2023). Detecting DDoS attacks using adversarial neural network. *Computers & Security*, 127, 103117. https://doi.org/https://doi.org/10.1016/j.cose.2023.103117

[15] Saied, A., Overill, R. E., & Radzik, T. (2016). Detection of known and unknown DDoS attacks using Artificial Neural Networks. *Neurocomputing*, *172*, 385-393. https://doi.org/https://doi.org/10.1016/j.neucom.2015.04.101

[16] Menon, A., Anurag, A., Ojha, R., Bhosale, H., & Oak, S. (2023). *DDOS Intrusion Detection Using Hybrid ML Model*. https://doi.org/10.1109/smartgencon60755.2023.10442411

[17] Azam, Z., Islam, Md. M., & Huda, M. N. (2023). Comparative Analysis of Intrusion Detection Systems and Machine Learning Based Model Analysis Through Decision Tree. *IEEE Access*, *11*, 1–1. https://doi.org/10.1109/ACCESS.2023.3296444

[18] S. Potluri, N. F. Henry, and C. Diedrich, "Evaluation of hybrid deep learning techniques for ensuring security in networked control systems," Sep. 2017, doi: 10.1109/etfa.2017.8247662. Available: https://doi.org/10.1109/etfa.2017.8247662

[19] U. J. Carrasquilla, "Benchmarking Algorithms for Detecting Anomalies in Large Datasets," 2010. Available: https://www.semanticscholar.org/paper/Benchmarking-Algorithms-for-Detecting-Anomalies-in-Carrasquilla/60d9e710304b9223c4206c0b29c192f7a0ed6f30

[20] P. Mitra, C. A. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 3, pp. 301–312, Mar. 2002, doi: 10.1109/34.990133. Available: https://doi.org/10.1109/34.990133

[21] J. Novakovic, P. Strbac, and D. Bulatovic, "Toward optimal feature selection using ranking methods and classification algorithms," Yugoslav Journal of Operations Research, vol. 21, no. 1, pp. 119–135, Jan. 2011, doi: 10.2298/yjor1101119n. Available: https://doi.org/10.2298/yjor1101119n

[22] A. Abbas, M. A. Khan, S. Latif, M. Ajaz, A. A. Shah, and J. Ahmad, "A New Ensemble-Based Intrusion Detection System for Internet of Things," *Arabian Journal for Science and Engineering*, vol. 47, no. 2, pp. 1805–1819, Aug. 2021, doi: 10.1007/s13369-021-06086-5.

[23] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm," I.E.E.E. Transactions on Computers/IEEE Transactions on Computers, vol. 65, no. 10, pp. 2986–2998, Oct. 2016, doi: 10.1109/tc.2016.2519914. Available: https://doi.org/10.1109/tc.2016.2519914

[24] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions," Artificial Intelligence Review, vol. 55, no. 1, pp. 453–563, Jul. 2021, doi: 10.1007/s10462-021-10037-9. Available: https://doi.org/10.1007/s10462-021-10037-9

[25] Zhou, L., Zhu, Y., Zong, T., & Xiang, Y. (2022). A feature selection-based method for DDoS attack flow classification. *Future Generation Computer Systems*, *132*, 67–79. https://doi.org/10.1016/j.future.2022.02.006

[26] S. Dongre and M. Chawla, "Analysis of feature selection techniques for denial of service (DoS) attacks," Mar. 2018, doi: 10.1109/rait.2018.8389000. Available: https://doi.org/10.1109/rait.2018.8389000

[27] Hindy, H., Atkinson, R., Tachtatzis, C., Bayne, E., Bures, M., & Bellekens, X. (2021). Utilising Flow Aggregation to Classify Benign Imitating Attacks. *Sensors (Basel, Switzerland)*, *21*(5). https://doi.org/10.3390/s21051761

[28] Umer, M. F., Sher, M., & Bi, Y. (2017). Flow-based intrusion detection: Techniques and challenges. *Computers & Security*, *70*, 238–254. https://doi.org/10.1016/j.cose.2017.05.009

[29] S. Yadav and S. Subramanian, "Detection of Application Layer DDoS attack by feature learning using Stacked AutoEncoder," Mar. 2016, doi: 10.1109/icctict.2016.7514608. Available: https://doi.org/10.1109/icctict.2016.7514608

[30] H. H. Jazi, H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling," Computer Networks, vol. 121, pp. 25–36, Jul. 2017, doi: 10.1016/j.comnet.2017.03.018. Available: https://doi.org/10.1016/j.comnet.2017.03.018

[31] D. Tang, S. Zhang, J. Chen, and X. Wang, "The detection of low-rate DoS attacks using the SADBSCAN algorithm," Information Sciences, vol. 565, pp. 229–247, Jul. 2021, doi: 10.1016/j.ins.2021.02.038. Available: https://www.sciencedirect.com/science/article/abs/pii/S0020025521001808?via%3Dihub

[32] Liao, Q., Li, H., Kang, S., & Liu, C. (2014). Feature extraction and construction of application layer DDoS attack based on user behavior. https://doi.org/10.1109/chicc.2014.6895878

[33] Abushwereb, M., Mustafa, M., Al-kasassbeh, M., & Qasaimeh, M. (2020). Attack based DoS attack detection using multiple classifier. arXiv.Org. https://doi.org/10.48550/arxiv.2001.05707

[34] M. Ozkan-Okay, R. Samet and Ö. Aslan, "A New Feature Selection Approach and Classification Technique for Current Intrusion Detection System," 2021 6th International Conference on Computer Science and Engineering (UBMK), Ankara, Turkey, 2021, pp. 227-232, doi: 10.1109/UBMK52708.2021.9559011

[35] V. T. Patil and S. Deore, "A Study Of Ddos Attack Detection Methods," ResearchGate, Aug. 2023, doi: 10.5281/zenodo.98549840. Available: https://doi.org/10.5281/zenodo.98549840

[36] Cordero, C. G., Vasilomanolakis, E., Wainakh, A., Mühlhäuser, M., & Nadjm-Tehrani, S. (2021). On Generating Network Traffic Datasets with Synthetic Attacks for Intrusion Detection. *ACM Transactions on Privacy and Security*, *24(2)*, 1–39. https://doi.org/10.1145/3424155.

[37] A. Rosay, E. Cheval, F. Carlier, and P. Leroux, "Network Intrusion Detection: A Comprehensive Analysis of CIC-IDS2017," Jan. 2022, doi: 10.5220/0010774000003120. Available: https://doi.org/10.5220/0010774000003120

[38] “Denial-of-service attack,” *Wikipedia*, Jul. 30, 2024. Available: https://en.wikipedia.org/wiki/Denial-of-service_attack

[39] Shekyan, “InstallationAndUsage,” GitHub. Available: https://github.com/shekyan/slowhttptest/wiki/InstallationAndUsage

[40] “Home - Wireshark Wiki.” Available: https://wiki.wireshark.org/

[41] “Snort - Network Intrusion Detection & Prevention System.” Available: https://www.snort.org/

[42] GeeksforGeeks, “Slowloris DDOS Attack Tool in Kali Linux,” GeeksforGeeks, Nov. 25, 2022. Available: https://www.geeksforgeeks.org/slowloris-ddos-attack-tool-in-kali-linux/

[43] Ahlashkari, “CICFlowMeter/ReadMe.txt at master · ahlashkari/CICFlowMeter,” GitHub. Available: https://github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt

[44] “lycos-ids2017/LycoSTand_features.pdf at main · 17204612710214/lycos-ids2017,” GitHub. Available: https://github.com/17204612710214/lycos-ids2017/blob/main/LycoSTand_features.pdf