electronics

MDPI

Article

Automated Network Incident Identification through Genetic Algorithm-Driven Feature Selection

Ahmet Aksoy *, Luis Valle and Gorkem Kar

Department of Computer Science and Cybersecurity, University of Central Missouri, Warrensburg, MO 64093, USA; lev33910@ucmo.edu (L.V.); kar@ucmo.edu (G.K.)
* Correspondence: aksoy@ucmo.edu

Abstract: The cybersecurity landscape presents daunting challenges, particularly in the face of Denial of Service (DoS) attacks such as DoS Http Unbearable Load King (HULK) attacks and DoS GoldenEye attacks. These malicious tactics are designed to disrupt critical services by overwhelming web servers with malicious requests. In contrast to DoS attacks, there exists nefarious Operating System (OS) scanning, which exploits vulnerabilities in target systems. To provide further context, it is essential to clarify that NMAP, a widely utilized tool for identifying host OSes and vulnerabilities, is not inherently malicious but a dual-use tool with legitimate applications, such as asset inventory services in company networks. Additionally, Domain Name System (DNS) botnets can be incredibly damaging as they harness numerous compromised devices to inundate a target with malicious DNS traffic. This can disrupt online services, leading to downtime, financial losses, and reputational damage. Furthermore, DNS botnets can be used for other malicious activities like data exfiltration, spreading malware, or launching other cyberattacks, making them a versatile tool for cybercriminals. As attackers continually adapt and modify specific attributes to evade detection, our paper introduces an automated detection method that requires no expert input. This innovative approach identifies the distinct characteristics of DNS botnet attacks, DoS HULK attacks, DoS GoldenEye attacks, and OS-Scanning, explicitly using the NMAP tool, even when attackers alter their tactics. By harnessing a representative dataset, our proposed method ensures robust detection of such attacks against varying attack parameters or behavioral shifts. This heightened resilience significantly raises the bar for attackers attempting to conceal their malicious activities. Significantly, our approach delivered outstanding outcomes, with a mid 95% accuracy in categorizing NMAP OS scanning and DNS botnet attacks, and 100% for DoS HULK attacks and DoS GoldenEye attacks, proficiently discerning between malevolent and harmless network packets. Our code and the dataset are made publicly available.

Keywords: network traffic analysis; incident classification; automated incident detection; network security; traffic pattern recognition; cybersecurity; machine learning in networking; protocol analysis; network forensics

Citation: Aksoy, A.; Valle, L.; Kar, G. Automated Network Incident Identification through Genetic Algorithm-Driven Feature Selection. *Electronics* **2024**, *13*, 293. https://doi.org/10.3390/electronics13020293

Academic Editor: Wajeb Gharibi

Received: 19 November 2023
Revised: 30 December 2023
Accepted: 6 January 2024
Published: 9 January 2024

1. Introduction

Over the past few years, Denial of Service (DoS), Operating System (OS) fingerprinting, and Domain Name System (DNS) botnet techniques have undergone significant advancements, becoming increasingly complex and challenging to identify [1–8]. While these activities may not always constitute an attack in the traditional sense, they encompass a spectrum of activities, including reconnaissance and denial of services. The distinction lies in the intent and context of such actions.

Among all the types of cyber attacks, DoS attacks are considered the most harmful, as they can entirely disconnect an organization from the Internet or severely impede network links, resulting in a significant disruption of packet delivery [8–10]. Similarly, OS fingerprinting attacks are crucial for cybercriminals to identify the OS of a target system, which can be exploited to launch further attacks or gain unauthorized access to the system.

Unlike DoS attacks, which aim to disrupt the target system, OS fingerprinting is a crucial component of reconnaissance rather than a standalone attack. In this stage, attackers gather crucial data about the target system regarding any potential security weaknesses that could be exploited, such as its network layout, interdependencies of services, and OSes in use [11,12]. While OS fingerprinting is not typically considered an attack, it plays a significant role in the broader context of cyber activities. It may be part of an attack described within the cyber kill chain framework. A widespread tool most commonly used for reconnaissance is NMAP. The output of NMAP provides information such as device type, OS family and generation, Common Platform Enumeration (CPE) representation, OS details, uptime guess, network distance, details of the operating system, estimation of uptime, network distance, prediction of Transmission Control Protocol (TCP) sequences, and generation of Internet Protocol (IP) Identifier (ID) sequences [13]. To perform fingerprinting, NMAP sends out a series of probes, analyzes the responses received, and compares them against a database of known OS characteristics. By carefully analyzing network traffic, NMAP is widely used for attempting to determine the OS running on the target device [14]. However, it is worth noting that the accuracy of NMAP's OS identification in real-world networks and on the Internet remains a topic of debate, and there is limited research available on this subject to serve as a reference [15].

Denial of Service (DoS) attacks have been increasingly disruptive and pose a significant challenge to the stability and security of networked systems. These attacks have been utilized for political purposes, encompassing cyberwarfare, hacktivist actions, and acts of terrorism [16]. Over time, the landscape of cyber threats has evolved, and the prevalence and impact of DoS and Distributed DoS (DDoS) attacks have continued to escalate. As such, these attacks now stand as formidable adversaries that security systems must confront [8,17,18]. The DoS GoldenEye and Http Unbearable Load King (HULK) attacks are malicious techniques aimed at overwhelming a target web server's resources, rendering it inaccessible to legitimate users. In the case of GoldenEye, it operates by bombarding the target server with an extensive barrage of HyperText Transfer Protocol (HTTP) GET or POST requests, exploiting the HTTP protocol's statelessness. This incessant flood of requests forces the server to allocate resources to each incoming request, leading to resource exhaustion and rendering the server unable to handle legitimate traffic. Similarly, the HULK attack generates numerous HTTP requests rapidly, aiming to exhaust the web server's resources, especially its network bandwidth and computing capacity. Both attacks share the goal of causing service unavailability, but they employ different methods to achieve this disruptive outcome, making them potent tools in the arsenal of cyber attackers [8,19]. Attackers are adopting more intricate strategies, utilizing botnets composed of compromised devices to launch DDoS attacks. These botnets can execute coordinated, large-scale attacks, making detection and mitigation more challenging. Attack vectors and techniques are also diversifying, including reflection and amplification attacks that exploit vulnerable servers to magnify the assault's impact. Moreover, attackers increasingly leverage encryption to obfuscate their activities and make attribution more complex. In response to these advancements, defenders are developing more resilient mitigation measures and leveraging machine learning, Artificial Intelligence (AI), and advanced traffic analysis to detect and thwart attacks in real-time. As the arms race between attackers and defenders continues, the evolution of DoS attacks remains critical in cybersecurity [20–22].

Amidst this ever-evolving landscape, a particular threat has gained prominence as well. As an insidious strain of malware, DNS botnets have significantly intensified their presence on the digital landscape and caused several negative economic impacts [7]. Their operational methods are remarkably disruptive, utilizing an extensive network of compromised devices to coordinate and execute synchronized attacks. The primary impact of DNS botnets is their potential to overwhelm and flood their target with malicious DNS traffic, rendering the targeted system virtually inaccessible. This assault can paralyze essential online services and disrupt business operations, thereby incurring significant economic losses. These economic disruptions can affect organizations ranging from corporations

and financial institutions to e-commerce platforms, healthcare, and governmental agencies. As a result, DNS botnets have emerged as a formidable threat in the cybersecurity arena. The harm inflicted by DNS botnets goes beyond mere financial losses. The consequences of these attacks extend to an erosion of trust and reputation for the targeted entities. Downtime resulting from a DNS botnet assault can leave a lasting impact on the user experience, potentially driving customers away and causing long-term damage to an organization's brand [6,7,23,24]. Additionally, service disruptions can result in significant data loss, posing a severe threat to the confidentiality and integrity of sensitive information. The consequences become dangerous when these attacks are directed towards vital infrastructure. They can potentially disrupt essential services such as healthcare, transportation, and emergency response systems, as noted in various studies [25,26]. It is worth highlighting that the increasing integration of Internet of Things (IoT) devices into critical infrastructure renders them particularly susceptible to exploitation. These connected devices often lack robust security measures, making them attractive targets for malicious actors seeking to amplify the impact of their attacks [27,28].

This paper aims to develop an automated methodology for detecting and distinguishing attacks from benign traffic in network data. We propose a distinctive approach to attack detection that leverages genetic algorithms (GA) and machine learning to extract unique features from the data and establish correlations with attack behaviors. We analyze packet headers and extract unique features associated with each TCP/IP stack and application layer, resulting in nearly 100% accuracy. As mentioned in our prior work [29–31], employing a GA for selectively identifying the most distinctive features in datasets offers numerous benefits.

With the help of GA and machine learning, our methodology eliminates noise and uncovers subtle attack signatures that could easily go unnoticed by other methods. Unlike conventional detection techniques, our approach does not hand-select features, but rather analyzes all features in the data and dynamically selects only the most relevant features to maximize accuracy. This level of flexibility ensures that our approach can quickly adapt to changes in attack behavior, provided that a suitable dataset is available.

By fundamentally selecting a subset of the most contributing features at the packet level, our approach enables faster detection and, therefore, prevents attacks before they can inflict significant damage. Our approach also is independent of the specific protocol(s) for performing classification. It can analyze a variety of packet headers, not limited to TCP/IP. This research has significant implications for developing more advanced and effective attack-prevention strategies. Some of the prominent passive fingerprinting tools, such as p0f, ettercap, and siphon, perform a perfect match in their signatures when classifying hosts [32–34]. In some cases, partial matching can still provide a reliable classification, which allows further flexibility. Our approach allows for the classification of packets utilizing a portion of the fingerprints generated by the machine learning model when a suitable model, such as a Decision Trees, is selected. Many approaches utilize existing fingerprinting databases when performing fingerprinting. Our approach does not rely on any existing tools, nor utilizes their fingerprinting databases, but instead generates its own fingerprints. However, our approach heavily depends on how representative the dataset is in extracting behavior-specific information. The performance and selection of features are proportional to the dataset. We employed various machine-learning algorithms for the classification of malicious and benign packets. Some of the algorithms include Decision Trees (DTs), Random Forest (RF), Gaussian Naive Bayes (GNB), and Multi-layer Perceptron (MLP). In this study, we employed these algorithms to identify and classify attacks accurately. By analyzing the classification accuracy of each algorithm, we identified the one that performed the best in each scenario. Additionally, we compared the results obtained with and without the use of GA. We made our dataset and tool publicly available at https://netml.github.io (accessed on 5 January 2024).

The rest of the paper is organized as follows: Section 2 summarizes the related work. Section 3 explains our methodology. Section 4 presents the identification performance of our approach, and Section 5 concludes the paper.

2. Related Work

Numerous research papers have emphasized the usage of semi-supervised incident classification and feature mapping to identify attacks. These studies demonstrate the application of different machine learning algorithms in identifying and classifying various incidents based on their severity and impact on systems. The papers utilize diverse datasets and techniques such as feature selection, pre-processing, and model optimization to achieve a high classification accuracy. They also highlight the need for further research to improve incident classification by incorporating more data sources and addressing the challenge of imbalanced datasets. Table 1 lists recent papers that specifically explore feature selection or enhancement methods such as Recursive Feature Elimination (RFE), Autoencoders, Genetic Algorithms (GA), and more. These papers are elaborated upon in the upcoming sections.

Table 1. Feature selection/enhancement in network attack classification.

Paper	Dataset	Approach	Performance
Ustebay et al. [35]	CICIDS2017 [36]	Recursive Feature Elimination (RFE) via Random Forest	91.0%
Pran et al. [21]	CICIDS2017 [36]	Hybrid Genetic Algorithm and Particle Swarm Optimization Technique (GPSO)	98.6%
Aksoy et al. [31]	Internet of Things (IoT) identification Dataset [37]	Genetic Algorithms (GA)	95.0%
Thalji et al. [38]	SQL Injection Dataset [39]	Neural Network Autoencoder	92.0%
Awad et al. [40]	UNSW-NB15 [41]	Recursive Feature Elimination (RFE) using Decision Tree (DT) models	95.0%
Liu et al. [42]	Bot-IoT [43]	Genetic Algorithms (GA)	94.4%

2.1. Network Scanning Incident Classification

In [44], the authors present a passive Operating System (OS) fingerprinting approach to detect unauthorized OSes in an enterprise network. They use Transmission Control Protocol (TCP) fields such as Time to live (TTL), total length, and window size to detect the OSes generating the packets. Their methodology involves analyzing network traffic and comparing it to a database of known OS fingerprints. The results show a high accuracy in identifying and detecting unauthorized OSes. Another paper aims to do something similar, but instead looks at the popular network scanning tool NMAP https://netml.github.io (accessed on 5 January 2024) [11,12]. Their paper discusses the importance of detecting NMAP scanning behavior to protect hosts from malicious attacks. They discuss that traditional defense methods like firewalls are less effective at detecting NMAP scanning. However, Intrusion Detection Systems (IDS) can monitor network security events and alert when abnormalities appear. The authors propose a Comprehensive NMAP Detection Rules (CNDR) set based on the Suricata system and consider IDS evasion. The CNDR achieves a detection rate of 100% for regular NMAP scanning and 91.7% for the detection accuracy of NMAP with IDS evasion on the authors' designed dataset.

2.2. DDoS Attack Incident Classification

Considerable efforts have been dedicated to classifying and detecting Denial of Service (DoS) attacks. Many papers employ similar approaches, which aim to detect and classify DoS attacks [45–49]. The Distributed Reflection Denial-of-Service attack (DRDoS) is covered extensively in [45]. Their study examines and identifies the differences between TCP-based and User Datagram Protocol (UDP)-based DRDoS attacks. They examine the DRDoS attack, and find that it exploits vulnerabilities in the UDP protocol to flood a target with traffic; they explain that the UDP protocol is vulnerable because it allows the amplification of responses and does not verify the source Internet Protocol (IP) addresses. A proposed solution is IEWA, which combines increased expenses and weak authentication to protect the Network Time Protocol (NTP). Another paper in which DRDoS is examined is [46]. The authors evaluate the susceptibility of popular UDP-based protocols to DRDoS attacks, finding 14 protocols vulnerable, with traffic multiplied up to a factor of 4670. They further identify millions of potential amplifiers for six vulnerable protocols, and evaluate countermeasures against DRDoS attacks, showing that poorly designed rate-limiting solutions are evaded by some attacks, and packet-based filtering techniques are also evaded. They propose a threat model that analyzes P2P botnets that use amplification attacks to understand the potential severity of such attacks. The key focus is that an attacker aims to consume all available bandwidth of a victim by using systems that reflect the attack traffic to the victim. Another work allows sharing attack data and anomaly profiles with other parties without disclosing data [47]. Recent works have explored FL methods on the CIC-DDoS2019 [50] dataset, including LwResnet, FLDDoS, and FIDS. However, some solutions rely on the vanilla FEDAVG algorithm, which can increase the convergence time on unbalanced non-i.i.d. attack data and may jeopardize clients' privacy with data-sharing mechanisms [47]. In response to these challenges, the authors suggest an adaptive Federated Learning (FL) approach named FLAD. FLAD facilitates the collaborative training of deep learning models using distributed profiles of cyber threats while preserving the confidentiality of training data. The proposed solution manages the FL process by dynamically allocating additional computation resources to members with more intricate attack profiles, all without the need to share test data. The study showcases that FLAD surpasses the performance of the original FL algorithm in terms of convergence time and accuracy across various unbalanced datasets featuring heterogeneous Distributed Denial of Service (DDoS) attacks. Meanwhile, other approaches aim to more closely fingerprint at the packet level [48]. The authors use an approach to classify traffic patterns based on their statistical properties, including packet length, packet inter-arrival time, and time-to-live. The proposed system generates application fingerprints based on transport layer packet-level and flow-level features. These fingerprints identify Distributed Denial of Service (DDoS) attacks by analyzing statistical information collected at the flow level. Paper [49] discusses how easily DoS attacks can be launched, but detection and response are often manual and slow. Present methods relying on packet headers are also vulnerable to spoofing. The framework relies on header content, transient ramp-up behavior, and spectral analysis, making it more challenging to spoof. By evaluating a regional Internet Service Provider (ISP)'s access links, they could detect 80 live attacks. The framework has several applications, including aiding in the rapid response to attacks, developing realistic models of DoS traffic, and estimating the level of DoS activity on the Internet.

2.3. Automated Attack Classification and Intrusion Detection

Several studies have been conducted on classifying and detecting network attacks using machine learning algorithms. The following papers [51–57] all have similarities in their methodology, but differ in which dataset and problem they tackle. Bayu Adhi et al. discuss using deep neural networks (DNNs) for classifying attacks in the transportation layer of Internet of Things (IoT) networks in their paper [51]. They discuss how anomaly detection is considered one of the most demanding tasks in intrusion detection systems (IDSs), and the authors propose a robust DNN classifier model that can intelligently detect different kinds

of attacks. The proposed method is evaluated on three benchmark datasets (UNSW-NB15, CIDDS-001, and GPRS) in wired and wireless network environments. The authors use a grid search strategy to obtain the best parameter settings for each dataset, and their experimental results show that the DNN approach is practical regarding accuracy, precision, recall, and false alarm rate. Another recent paper looked into preventing and detecting cyber attacks on IoT devices [52]. They evaluate various machine learning techniques, including k-nearest neighbor, support vector machine, decision tree, naive Bayes, random forest, artificial neural network, and logistic regression, for both binary and multi-class classification. The authors use the Bot-IoT dataset to compare and evaluate the algorithms based on accuracy, precision, recall, accuracy, and log loss metrics. Their results demonstrate that random forest outperforms other algorithms compared to binary classification, while k-nearest neighbor (kNN) performs best in multi-class classification. The paper also provides an overview of the increasing number of IoT devices, the associated risks of cyber-attacks, and the limitations of traditional intrusion detection systems in detecting these attacks. The authors of [53] provide a method of identifying DDoS attacks using a semi-supervised machine learning approach. The approach involves obtaining clusters of network traffic data using unsupervised methods and then labeling them through a voting method to mark normal, DDoS, and suspicious traffic. The dataset used consists of three features extracted using Principal Component Analysis (PCA), and three machine learning algorithms are applied—kNN, Support Vector Machine (SVM), and Random Forest (RF)—to classify the labeled traffic data. The emergence of Software-Defined Networking (SDN) has given rise to a new form of networking, bringing about new types of attacks [54]. Ahuja et al. propose a machine learning-based solution to classify benign traffic from DDoS attack traffic by using novel features for DDoS attack detection. They create a dataset of SDN traffic logs and use a hybrid machine learning model of Support Vector Classifier with Random Forest (SVC-RF) to classify traffic. The authors highlight the security issues of SDN and explain how DDoS attacks can occur at different architectural planes. To achieve this, the authors create a dataset of SDN traffic logs with novel features for DDoS detection, using a hybrid machine learning model to classify traffic and evaluate the model's accuracy. The authors used a hybrid model of Support Vector Classifier with Random Forest (SVC-RF) to classify the SDN traffic logs. Regarding threat classification, two papers provide an excellent framework [56,57]. The two papers use a similar approach of proposing a new method to detect anomalies. The [56] paper proposes an algorithm that involves three steps: identifying related packets/flow records, deriving metrics related to the anomaly, and classifying the anomaly using a signature-based approach. In contrast, ref. [57] proposes developing machine-learning models to classify HyperText Transfer Protocol (HTTP) requests as normal or malicious to detect web application attacks. They both validate their proposed methods on datasets while highlighting the importance of automated classification.

2.4. DNS-Based Botnet Detection

There have also been many studies in the field of Domain Name System (DNS) botnet detection. Singh et al. discuss the limitations of existing surveys, which either need more in-depth comparisons or cover the full spectrum of DNS-based botnet detection techniques. This work aims to tackle this problem; the research focuses on botnet detection methods, specifically those using the DNS Protocol. The contributions of the study include categorizing DNS-based botnet detection techniques, providing an analysis of each technique within these categories, and proposing essential attributes for an innovative DNS-based botnet detection system [6,7]. Similarly, in a related context, the following work discusses a proposed system called "Notos" for dynamic DNS reputation scoring, which has shown promise in identifying malicious domains with high accuracy and low false positives in an extensive ISP's network [23]. Another more active approach was introduced by Ma et al., who proposed an active probing approach by looking at DNS query characteristics, such as leveraging the Time to live (TTL)-based caching mechanism of R-DNS

servers. By probing the cache of these servers, the monitoring system can observe cache behavior and, in turn, estimate DNS query activities. This approach significantly reduces management costs and privacy concerns [24]. Other research also leverages DNS query data to detect malicious DNS traffic [6,58–60]. Another study has adopted a comparable methodology in a published paper, employing Genetic Algorithms (GA) to enhance the feature selection capabilities of an Intrusion Detection System (IDS). This approach aims to optimize the system's performance within resource-constrained environments such as the Internet of Things (IoT) [42]. This study showcases an 85% reduction in required features while upholding a commendable 94% accuracy rate. In contrast to our methodology, this research does not delve into the importance of the selected features or endeavor to establish correlations with other studies to substantiate the significance of Genetic Algorithm (GA) elected features. This omission allows for future work. The collective body of research in DNS botnet detection highlights the diverse strategies employed to address this critical cybersecurity concern, from passive monitoring to active probing, all with the common aim of enhancing network security and safeguarding against malicious DNS traffic.

2.5. Feature Selection Approaches: Recursive Feature Elimination (RFE) and Autoencoder Analysis

While our approach only covers using Genetic Algorithms (GA) to find the most prominent features in packet header data, there are other approaches which aim to do the same thing. Such approaches are Recursive Feature Selection (RFE) and Autoencoders. There have been studies which focus on using Recursive Feature Selection (RFE) to perform intrusion detection [35,40,61]. The Recursive Feature Elimination (RFE) algorithm systematically removes features, initially evaluating classifier performance with the entire feature set and progressively generating subsets by eliminating features. This iterative process determines the most effective subset [35]. Awad et al. combine cross-validation with feature elimination, further refining the feature count and enhancing model performance [40]. Similarly other methodologies such as Autoencoders are also viable for feature selection. The authors from [62] introduce a novel approach leveraging Autoencoder (AE) technology to discern behavioral patterns in IoT attacks. Their method innovatively constructs features by autonomously learning semantic similarities between command-derived data, providing improved clustering and a deeper understanding of attack behavioral patterns compared to traditional approaches. Another approach looks specifically at SQL injection attacks. Thalji et al. proposed leveraging an Autoencoder network (AE-Net) to automatically engineer features for detecting SQL injection attacks. By extracting deep features from SQL textual data, the AE-Net facilitates the creation of a more efficient data representation. Their method, integrated with the extreme gradient boosting classifier, achieved a k-fold accuracy score of 0.99, surpassing existing approaches. Employing techniques like hyperparameter tuning and validation via k-fold cross-validation ensured robust performance evaluation [38].

3. Automated Attack Identification from Network Traffic

We employ a systematic methodology leveraging network packet headers to address the challenge of automated attack identification. Our strategy involves utilizing Genetic Algorithms (GA) and machine learning to discern behaviors indicative of infected hosts. Evaluating each packet's accuracy, measured through feature extraction across Network, Transport, and Application Layers, is a critical indicator. This analysis not only aids in accurate incident classification, but also assesses the layer-specific contribution of features. We propose a generalized approach to enhance our strategy's applicability beyond this specific dataset. Initially utilizing GA for feature selection and subsequently integrating these features into classification models, our methodology aims for broader adaptability across various datasets. Details regarding the specific dataset and experiments are provided in the experiment section for comprehensive understanding and validation.

3.1. Data Preparation

For the NMAP Operating System (OS) scan dataset, we analyzed the Kitsune Network Attack dataset [63] to determine the protocols associated with malicious and benign packets. The OS Scanning dataset had Internet Protocol (IP) and Transmission Control Protocol (TCP) protocols comprising 65,700 malicious and 1,022,501 benign packets. We created our dataset to analyze the Denial of Service (DoS) GoldenEye and Http Unbearable Load King (HULK) attacks, and published it [64]. Similar to the NMAP OS Scan, malicious and benign Packet Captures (PCAPs) were captured from them, and we extracted the HyperText Transfer Protocol (HTTP), TCP, and IP protocols. For the malicious PCAP, we have 179,906 for GoldenEye and 241,991 for HULK. For the benign traffic, we captured a total of 1604 packets. For the analysis of the Domain Name System (DNS) botnet, we utilized the dataset from [65]. This dataset comprises two sets of traffic captures: one encompasses malicious DNS data related to nine different botnets, and the other encompasses benign DNS data associated with 19 widely recognized software applications. The protocols we extracted include IP, User Datagram Protocol (UDP), and DNS. Regarding the capture sizes, the malicious dataset comprises 182,720 packets, whereas the benign dataset encompasses 495,522 packets.

The initial attack from the Kitsune dataset we analyzed was the OS fingerprinting reconnaissance attack [63]. This attack employs the NMAP tool to scan the network using probes to determine the Operating System (OS) employed by the scanned targets based on the characteristics of the return traffic [13]. While OS scanning reconnaissance can potentially provide attackers with information to identify vulnerabilities [11], it is essential to note that scanning itself may not always be inherently dangerous. Exploiting system weaknesses can be attempted without prior scanning, and in some cases, such scans are conducted to minimize the 'noise' associated with blind exploitation attempts. Passive reconnaissance services like Shodan.io can also gather information similar to internet-facing systems. However, the context can change when considering internal hosts, especially if they are standard workstations. In such cases, unauthorized OS scanning can pose a security risk and may warrant attention. Regardless of the scenario, it is essential to emphasize the importance of strong security practices, including regular system patching and network monitoring, to detect and respond to any potential threats effectively.

The second type of attacks we analyzed were the DoS attacks, specifically the DoS GoldenEye and DoS HULK attacks [64]. These attacks are designed to overwhelm target systems, such as web servers, with a flood of requests, ultimately causing service disruption or degradation. Both attacks employ various obfuscation techniques to make each request appear unique and circumvent security measures. They both randomize user agents, introduce variability in request parameters, and utilize obfuscation methods to disrupt server responses and bypass security measures. GoldenEye appends a suffix to Uniform Resource Locator (URLs) to bypass Content Delivery Networks (CDNs) and employs real-time parameter randomization. At the same time, the HULK attack uses reference forgery, Keep-Alive with variable time windows for stickiness, and "no-cache" requests to ensure unique server responses. However, HULK attacks demand significant client-side resources, potentially requiring multiple nodes for high attack rates. This can overwhelm servers and leave legitimate requests unanswered when concurrent connection limits are reached. These tactics underscore the challenges in defending against such disruptive attacks [8,66,67]. Detecting these attacks allows organizations to protect their online services, maintain server availability, and ensure a consistent user experience. Timely detection and response are vital in thwarting malicious actors who employ such tactics to compromise the stability and functionality of web-based systems, ultimately safeguarding the integrity and availability of digital resources [2,4]. Just like the NMAP OS Scan, these tools can be employed for load testing or vulnerability testing on web servers without the intent of launching a malicious attack.

The final attack we analyzed was the DNS Botnet dataset. The botnet dataset comprises complete DNS packets from nine exploit kits gathered within a virtual setting. Each bot was installed within a Windows XP virtual machine, running for several consecutive days.

This virtual environment underwent comprehensive monitoring from the DNS server to the router [68]. The selection of this dataset was based on the diversity and utility present in their PCAP captures. Protecting against DNS-based botnets is imperative, due to their growing prevalence and sophisticated nature in cybercrimes. Botnets, such as DNS, have become the weapon of choice for cyber attackers, offering them remote control over vast networks of infected hosts. Initially designed for legitimate purposes, the DNS has been exploited as a communication channel for botnet command and control, making it a crucial component of malicious activities. The use of DNS by botnets not only enables them to hide their identities, but also poses a severe challenge to detection. Researchers have recognized this escalating threat and developed detection frameworks, machine learning classifiers, and proactive techniques for analyzing DNS query patterns. As the number of infected machines continues to rise, addressing DNS-based botnets becomes paramount in protecting organizations and individuals from economic damage and security threats in today's evolving cyberspace [6,7,24,68].

The pre-processing applied to the respective datasets is described in Algorithm 1. After determining the protocols used in each dataset, we categorized the packets based on their topmost protocol layers. For instance, we collected all IP and UDP headers when examining a dataset employing the UDP protocol for the attack. This categorization was achieved using the tshark/pyshark libraries, which provided parsed packet dictionaries, enabling us to extract packet headers from the specified protocol. These libraries, on average, furnish hundreds of key-value pairs for each examined header, with some reaching 60 or more for HTTP and 300 or more for DNS [69]. These categorized packets were denoted as P_i (line 1). We split the packets of each protocol i into two categories: malicious and benign, as P_i^m and P_i^b, respectively (lines 5–10). We then split malicious and benign packets, P_i^m and P_i^b, into three equivalent-sized files, resulting in a total of six files for each protocol i. Finally, for each protocol i, we created three batches of packets $P_i^{batch_1}$, $P_i^{batch_2}$, and $P_i^{batch_3}$, where each of them contains one-third of malicious and benign packets (lines 11–17). The reason for splitting is to ensure a k-fold cross-validation, k being 3 in this case. K-fold cross-validation helps reduce bias in model evaluation, allows more reliable performance estimation, and generalizes the machine learning model generated for its applicability to newer data. We will discuss the reason behind this splitting in depth in Section 3.2. We will also describe the use of GA for feature selection and processing of the three batch files during this phase.

Algorithm 1 Data Pre-processing Procedure

1: Initialize a set P containing n subsets: $P = \{P_1, P_2, \ldots, P_n\}$
2: **for** each subset P_i in P **do**
3: Initialize P_i with k elements: $P_i = \{p_1, p_2, \ldots, p_k\}$
4: **end for**
5: **for** each subset P_i in P **do**
6: Define P_i^m with l elements: $P_i^m = \{p_1^m, p_2^m, \ldots, p_l^m\}$
7: Split P_i^m into three equal parts
8: Define P_i^b with r elements: $P_i^b = \{p_1^b, p_2^b, \ldots, p_r^b\}$
9: Split P_i^b into three equal parts
10: **end for**
11: **for** each subset P_i in P **do**
12: **for** $j = 1$ to 3 **do**
13: Initialize batch j of P_i: $P_i^{batch_j}$
14: Include the j-th third of P_i^m in $P_i^{batch_j}$
15: Include the j-th third of P_i^b in $P_i^{batch_j}$
16: **end for**
17: **end for**

3.2. Genetic Algorithms for Feature Subset Selection

We utilize a Genetic Algorithm (GA) to automate the process of identifying a subset of features that play a significant role in classifying network packets. GA allows us to detect the most contributing features without any expert input. It also helps increase classification accuracy by eliminating features causing noise in the dataset. In our previous work [70], we compared the classification accuracy of various feature selection algorithms such as filter methods, wrapper methods, hybrid methods, and search optimization techniques such as GA, Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO) for IoT fingerprinting utilizing TCP/IP packets. We observed that search optimization algorithms yield higher classification accuracy. Therefore, we utilized the GA to benefit the most in the accuracy of our classification.

As seen in Algorithm 2, we initially populate randomly generated binary-encoded solutions called chromosomes (line 1). Each solution in the population represents a potential solution, indicating which features to select and which to ignore. Unless the algorithm converges to the optimal solution, or the current generation is less than the maximum generation allowed, the GA continues exploring potential solutions (line 5). The *maxGenerations* in our case was set to 100. We wanted to ensure the GA does not enter a potential loop while allowing enough iterations for a better solution. The fitness of each solution is evaluated using a fitness function, which calculates the contribution of the potential solution to the classification of the dataset (line 6). The fitness function is explained in depth in Section 3.3. In (line 7), parents are selected for reproduction based on their fitness scores. In our implementation, we employed a roulette wheel selection method with elitism. In other words, we ensured that a portion of the best solutions obtained from previous iterations is retained in the subsequent generation. We also performed a uniform crossover operation where offspring are created by combining the genetic information of two parent solutions. The crossover rate determines the probability of each gene being swapped. We utilized the value of 0.5 for the *crossoverRate*. We then applied a mutation to the offspring with a predefined mutation rate of 0.015. GA creates a new population of solutions in each generation through selection, crossover, and mutation processes.

Algorithm 2 Genetic Algorithm (GA) Setup

1: $population \leftarrow$ INITIALIZEPOPULATION($populationSize, solutionSize$)
2: $bestSolution \leftarrow$ None
3: $bestFitness \leftarrow -\infty$
4: $generation \leftarrow 0$
5: **while** $generation < maxGenerations$ and not converged **do**
6: $\quad fitnessScores \leftarrow$ EVALUATEFITNESS($population$)
7: $\quad parents \leftarrow$ SELECTPARENTS($population, fitnessScores$)
8: $\quad newPopulation \leftarrow$ empty list
9: $\quad$ **while** size of $newPopulation < populationSize$ **do**
10: $\quad\quad$ Select $parent1, parent2$ from $parents$
11: $\quad\quad child \leftarrow$ UNIFORMCROSSOVER($parent1, parent2, crossoverRate$)
12: $\quad\quad child \leftarrow$ MUTATE($child, mutationRate$)
13: $\quad\quad$ Append $child$ to $newPopulation$
14: $\quad$ **end while**
15: $\quad population \leftarrow newPopulation$
16: $\quad$ Update $bestSolution$ and $bestFitness$ if a better solution is found
17: $\quad generation \leftarrow generation + 1$
18: **end while**
19: **return** $bestSolution, bestFitness$

3.3. Fitness Function Implementation

For each potential solution generated by the GA, which is called a chromosome, it is essential to determine the quality of the solution. The quality of solutions is used

to determine the most information-gaining solutions in each iteration. The quality of a given solution is determined by the fitness function, which is problem-specific. In our case, we implemented a fitness function that attempts to increase the accuracy of the machine-learning classification of incidents while reducing the number of features used. This approach enables not only a higher accuracy, but also more efficiency.

The dataset was divided into $batch_1$, $batch_2$, and $batch_3$ for training, validation, and testing, using a random splitting approach as discussed in Section 3.1. We analyzed to assess our machine learning model's robustness and generalizability. This analysis included examining the number of packets per label in each split and investigating whether there were malicious packets in the testing set that had feature vectors identical to those in the training set on a one-to-one basis. Identifying many malicious packets in the testing set with identical feature vectors in the training set could suggest overfitting, unless such features and values are inherently specific to the attacks being analyzed and cannot be generalized. After GA comes across a potential solution, we iterate through the bits in the binary string that we use to represent the potential solution to determine which features are to be removed and which features are to be retained. The value 0 indicates that the corresponding feature in the dataset is to be removed, and the value 1 indicates that the corresponding feature is to be retained. After classifying the features, we remove them from $batch_1$ and $batch_2$ files and generate a reduced dataset. We use a machine learning algorithm to train $batch_1$, test it on $batch_2$, and record the accuracy. In order to avoid any potential bias in our results, we train our model using $batch_2$ and test it on $batch_1$ as well. We then take the average of both accuracy. Now that we know how well the current solution performs, we also determine its contribution to reducing the number of features used. Then, we calculate the fitness value that embodies both the accuracy and the rate of the number of features used to determine the quality of a potential solution.

Let F be the fitness function, s be the potential solution of length n, where $s_i = 1$ if the i feature is to be selected, and $s_i = 0$ if the i feature is to be removed, k be the number of features selected, $F_a(s)$ be the average machine learning classification accuracy on the reduced dataset, and $F_f(s)$ be the measure of the number of features selected in the potential solution s, such that:

$$F_f(s) = 1 - (\frac{k-1}{n-1}) \tag{1}$$

Then, the fitness function $F(s)$ can be defined as a weighted sum of $F_a(s)$ and $F_f(s)$:

$$F(s) = 0.7 * F_a(s) + 0.3 * F_f(s) \tag{2}$$

After testing several combinations of weights for the fitness function, we observed that the values presented in Equation (2), as demonstrated in Table 2, were the best-performing set of values for the classification. If we increase the weight for the term in Equation (1), GA ends up sacrificing the accuracy in favor of reducing the number of features. Conversely, if we reduce the weight, GA tends to get stuck with a solution that contains more features than necessary.

Table 2. Parameter optimization.

Weight	Num. of Features	Accuracy
1.0, 0.0	25	94.5%
0.9, 0.1	7	93.7%
0.8, 0.2	5	94.7%
0.7, 0.3	4	95.5%
0.6, 0.4	3	94.1%

Each iteration of the GA involves using a population of individuals, where each individual is responsible for evaluating a potential solution. The population size, denoted

as *population_size*, is set to 50 in our experiments. Additionally, the GA requires specifying the number of generations (or iterations) to determine when to terminate. Our previous work [29] observed that using $n\_generations = 10$ yields high classification accuracy. It is important to note that *n_generations* represents the number of GA iterations, not to be confused with n used earlier to represent the length of the solution vector. We stopped the GA when we observed ten consecutive iterations with the same solution and accuracy to ensure termination criteria. Key parameters, such as crossover and mutation probabilities, were set to 0.5 and 0.015, respectively.

3.4. Machine Learning Classification

As discussed in Section 3.2, the GA runs the fitness function, which performs machine learning classification on the network packet headers with the subset of features determined in the chromosome of the solution at hand to determine the contribution of each potential solution. We used several machine learning algorithms and compared their results. In our previous work [29–31,71], we observed that rule-based algorithms generate a high classification accuracy. Therefore, we used four rule-based algorithms: Decision Trees (DT), Random Forest (RF), Gaussian Naive Bayes (GNB), and Multi-layer Perceptron (MLP), and compared our results. To set up the machine learning algorithms, we utilized the default settings of Python's scikit-learn library. For DT, the *criterion* parameter was set to '*gini*', *min_samples_leaf* was set to 1, *min_samples_split* was set to 2, and *n_estimators* was set to 100. For MLP, *hidden_layer_sizes* was set to $(100, 50)$, and *max_iter* was set to 2000. In order to avoid any potential bias, we obtained results for all three combinations of batches and recorded their average, see Figure 1. The results presented in Section 4 are based on these average results.

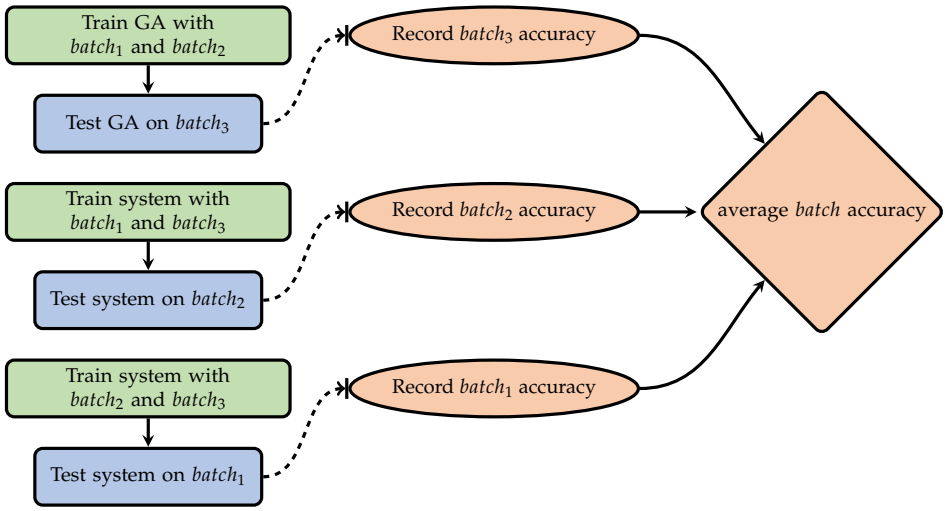

Figure 1. Training and testing process.

4. Experimental Results

This section showcases the peak performances attained for each protocol within the datasets. In this examination, we focus on the use of protocol headers in the classification of incidents. We then subject the features selected by the Genetic Algorithm (GA) to assessment using four distinct machine learning classifiers, namely Decision Tree (DT), Random Forest (RF), Gaussian Naive Bayes (GNB), and Multi-Layer Perceptron (MLP). The aim was to gauge their effectiveness in classifying data by applying a GA feature selection methodology. The key metrics of interest are the maximum accuracy achieved with selected features using GA and the maximum accuracy achieved using all features without GA. For the computational need of our research, we used the Chameleon cloud servers [72]. In all the following instances where accuracy is addressed, it pertains to the F1 score. Table 3 illustrates how our research compares to other studies employing a form

of feature selection/reduction, specifically focusing on papers that have tested on similar attack types.

Table 3. Comparison between other studies utilizing feature engineering.

Paper	Method	Classifier	Attack	Performance
Ustebay et al. [35]	Recursive Feature Elimination (RFE) via Random Forest	Deep Multilayer Perceptron (DMLP)	Distributed Denial of Service (DDoS)	91.0%
Pran et al. [21]	Hybrid Genetic Algorithm and Particle Swarm Optimization Technique (GPSO)	Adaptive Artificial Neural Network (AANN)	Denial of Service (DoS) Golden Eye	98.7%
Awad et al. [40]	Recursive Feature Elimination (RFE)	Decision Tree (DT)	UNSW-NB15 [41]	95.0%
Liu et al. [42]	Genetic Algorithms (GA)	Random Forest (RF)	Internet of Things (IoT) Botnet	94.4%
Proposed Work	Genetic Algorithms (GA)	Decision Tree (DT)	Denial of Service (DoS) Golden Eye	100%
		Random Forest (RF)	Domain Name System (DNS) Botnet	94.8%

4.1. OS Scanning Identification

The Operating System (OS) Scanning dataset was generated by utilizing the widely-used NMAP tool to conduct network scans, enabling the identification of hosts and their respective OS. To mitigate any potential bias stemming from Internet Protocol (IP) addresses, port numbers, IP Identifiers, and checksums, we began by filtering out such features from the dataset. This elimination process enables the algorithms to more effectively identify distinct characteristics relevant to the attacks rather than the hosts from which the data were collected. We aimed to ascertain which combination of packet headers would yield the highest accuracy. Hence, we categorized our findings based on IP, Transmission Control Protocol (TCP), and IP and TCP protocols.

Figure 2 showcases the accuracy achieved by employing packet headers of various protocols. The results depict the accuracy of each algorithm in our analysis. For IP-only feature selection, Decision Trees (DT) attained an average accuracy of 47% with GA-selected features and 67% when using all features. The Random Forest (RF) classifier achieved an average accuracy of 41% with the assistance of GA while obtaining the same level of accuracy without GA. Gaussian Naive Bayes (GNB) achieved 47% accuracy with GA-selected features and 71% with all features. Multi-Layer Perceptron (MLP) reached an average accuracy of 41% with GA and 39% without GA. Although classification accuracy using all features surpasses the GA-selected features, the results among the batches are inconsistent, indicating the non-reliability of IP protocol for the classification of NMAP. This outcome is attributed to the possibility that there is insufficient attack or software fingerprinting information within this header to provide a significant advantage over the other algorithms. In the case of MLP, we observe similar accuracy with and without GA, which can be linked to GA's capability to reduce noise and enhance the model's accuracy, as discussed in prior studies [29,73,74].

When focusing on TCP features, we observed higher accuracy across all algorithms. For the Decision Trees (DT) classifier, we observed an accuracy of 78% with GA-selected features and 89% when using all features. Similarly, in the case of the Random Forest (RF) classifier, the accuracy increased to 67% with GA-selected features and 91% when utilizing all features without GA. Gaussian Naive Bayes (GNB) exhibited enhanced accuracy using GA with 94% and 87% using all features, showcasing the flexibility of GA in feature selection. Lastly, for the MLP classifier, the utilization of GA-selected features led to an

accuracy enhancement of 86% when compared to 80% using all features. These results indicate that the GNB is the most suitable classifier for the TCP protocol. It is also worth noting that GA employed only up to 7 of the 275 available features. In contrast to feature selection, the results for IP were similar, where GA utilized at most 2 out of 138 features. These findings suggest that the uniqueness of the attack lies predominantly within the TCP features, as achieving close to 90% accuracy is attainable solely through TCP analysis for most algorithms. Consistent with our findings, the TCP header is the primary focus for conducting OS scanning, as indicated by previous studies [11,12]. This emphasis on the TCP header explains why higher accuracy is achieved when analyzing this specific component.

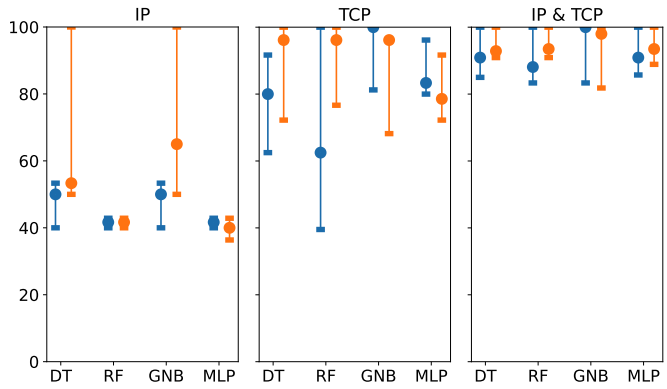

Figure 2. OS scanning identification.

Combining both protocols yields the highest accuracy. The Decision Tree (DT) classifier, including GA-selected features, resulted in a maximum accuracy of approximately 91%, while opting for all features without GA yielded a slightly higher maximum accuracy of about 94%. Similarly, the Random Forest (RF) classifier exhibited a maximum accuracy of approximately 90% with GA-selected features, which is slightly less than when all features were used, where it reached about 94%. In contrast, the Gaussian Naive Bayes (GNB) classifier demonstrated enhanced accuracy, with a maximum accuracy of roughly 94% with GA-selected features, compared to approximately 93% without GA. Surprisingly, the MLP classifier showcased consistent maximum accuracy of approximately 92% for both GA-selected features and 94% for all features. These findings emphasize the diverse impact of GA-based feature selection on classifier performance, highlighting the need for a tailored approach based on the specific classifier and dataset, which we achieve by automating such a process.

Table 4 illustrates the features the genetic algorithm (GA) selects across all protocols. We observe the prominent selection of the *tcp.analysis* parameter. Additionally, certain features, like the *tcp.window_size_value*, suggest its importance as a distinguishing factor in NMAP's OS fingerprinting methodology, as supported by previous studies [12,44,75]. Unlike features such as the data payload, which attackers can readily modify, the *tcp.window_size* is less susceptible to alteration unless modifications are made to NMAP's source code [12]. When considering the *tcp.analysis* feature, it is crucial to emphasize that this is a feature uniquely given by Wireshark that is used to see and follow TCP flags. The features of $tcp.flags.ack$ and $tcp.flags.fin$ combined with other features are intrinsic features of NMAP [44,76]. Similarly, the *tcp.hdr_len* fields can be utilized to identify unauthorized OS scanning activities within a network [12]; however, the *tcp.hdr_len* will change when the payload or header size changes, as previously indicated in the studies [12,77]. Lastly, the $ip.flags.df$ is the only prominent IP-specific field picked by the two algorithms. Other work shows this feature to be used by NMAP and any other tool to better avoid detection [78,79]. These findings emphasize the potential utility of specific features in identifying NMAP OS scanning activities. Simultaneously, our approach can automatically detect features from a given dataset that align with the accumulated scientific knowledge base.

Table 4. OS scan selected features.

Features	DT	RF	GNB	MLP
tcp.analysis	✓	✓	✓	✓
tcp.window_size_value	✓	✓	✓	✓
tcp.flags	✓	✓	✓	✓
tcp.flags.ack	✓	✓		✓
tcp.flags.fin	✓	✓		✓
tcp.hdr_len	✓	✓	✓	
ip.flags.df	✓			✓
Σ (features)	7	6	4	6

4.2. DoS GoldenEye

In Figure 3, we demonstrate the accuracy achieved using our approach in classifying the DoS GoldenEye attack. When utilizing the IP protocol, the Decision Tree (DT) classifier achieved a maximum accuracy of 52% with and without GA-selected features. The Random Forest (RF) classifier also attained a maximum accuracy of 52% with both GA-selected features and all features. Gaussian Naive Bayes (GNB) displayed the most significant improvement, achieving a maximum accuracy of 72%, notably higher than the accuracy of DT and RF. Multi-Layer Perceptron (MLP) achieved a maximum accuracy of 44% with both GA-selected features and all features, yielding the lowest accuracy among all algorithms.

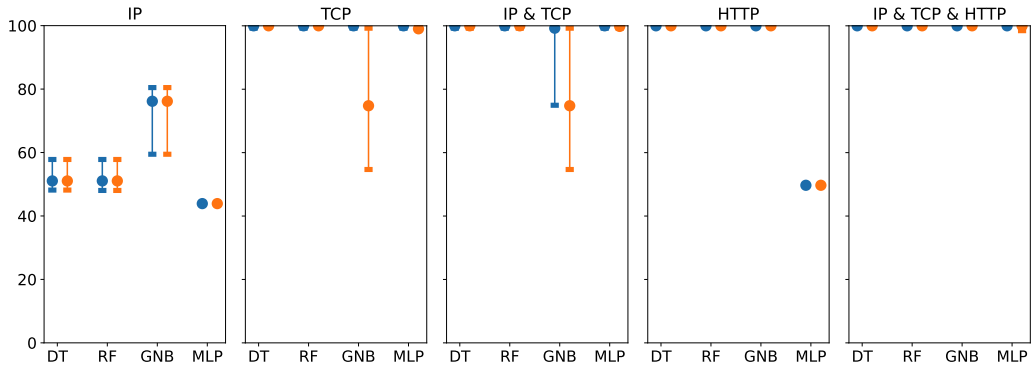

Figure 3. DoS GoldenEye identification.

When considering the TCP protocol headers, in the case of the Decision Tree (DT) and Random Forest (RF) classifiers, the maximum accuracy of 99% with selected features closely approached the maximum accuracy of 100% obtained with all features. Similarly, the Multi-Layer Perceptron (MLP) classifier yielded a maximum accuracy of 99% with selected features and all features. Conversely, the Gaussian Naive Bayes (GNB) classifier exhibited a high accuracy of 99% with GA-selected features, while the accuracy when using all features was notably lower at 76%.

For the HyperText Transfer Protocol (HTTP)-only header classification, we observed that the maximum accuracy of 100% with selected features was identical to the maximum accuracy of 100% obtained with all features using all the algorithms except for MLP. As for MLP, we have the lowest out of all other algorithms with an accuracy of 49% for both the use of GA and without. The reason we achieve 100% is because the benign packets have the *http.cache_control* field set to null, whereas the malicious ones do not. Although this behavior could indicate an overfitting, we observed another case using the MLP classifier where the *http.accept* field was selected to perform classification at high accuracy as well. Such results indicate that GA can find uniqueness based on the dataset provided. Even if the newer data to be trained with includes non-null values for *http.cache_control* field, GA will automatically look for any other combination of fields that yield as high an accuracy as possible, as long as the labels in the dataset are distinguishable. Similar research endeavors have identified these specific markers as distinct fingerprints for these attacks. Further

elaboration on this aspect can be found in Section 4. There are, however, certain cases where it is impossible to perform classification, regardless of which feature combination is used, due to highly similar, if not identical, behavior. For example, we observed in our previous work that the classification accuracy between Microsoft Windows 7 and 8 OSes is 50% because they use the same network libraries [30].

Notably, DT and RF did not experience substantial improvements in accuracy with the application of GA feature selection, as their performance remained consistent when using all features. In contrast, GNB exhibited significant benefits, showcasing a remarkable accuracy boost when GA-selected features were employed for TCP-only classification. In the context of IP-specific classification, although it is not featured in Table 5 due to its relatively low accuracy, it is worth noting that *ip.len* emerged as the sole consistently selected feature across all algorithms. Previous research has explored the potential use of this feature for device fingerprinting purposes, as indicated by references [13,80]. However, it is crucial to highlight that while *ip.len* may serve as a valuable component for device fingerprinting, its standalone utilization needs to be improved for classifying our attack data. Moving to TCP, specific features took center stage across the classifiers in the feature selection process. Notably, *tcp.window_size*, *tcp.flags*, *tcp.flags.ack*, *tcp.flags.syn*, and *tcp.flags.push* were the most frequently chosen features, indicating their crucial roles in shaping the classification outcomes. The *tcp.window_size* was picked predominantly more than other features for TCP header classification. Similar to *ip.len*, this feature is a prominent feature for fingerprinting devices [46,47,50]. TCP flags, such as *tcp.flags.ack*, *tcp.flags.syn*, and *tcp.flags.push*, are reliable features used to identify the DoS GoldenEye attack [8,81,82]. Specifically, *tcp.flags.push* and *tcp.flags.ack* are used for rapid resource depletion [8]. In the context of HTTP analysis, the features *http.cache_control*, *http.accept_encoding*, and *http.connection* were the most preferred by GA. Looking first at *http.cache_control* and *http.connection*, which are specific application layer features used in the DoS GoldenEye attack, these headers are used to persist connections until all the available sockets on the target server are exhausted, causing a denial of service to the web server [22,83,84]. An interesting finding is how GA could pick on the *http.accept_encoding* feature because the specific program used to generate the attack data [64] has a function that randomizes between five specific encoding schemes. While comprehensive studies specifically addressing *http.content_length_header* and *http.request_version* remain limited, it can be inferred that the selection of *http.content_length_header* is influenced by a combination of factors, including the presence of *http.cache_control*, *http.connection*, and other header-related features. This collective influence highlights *http.content_length_header* as a distinctive and noteworthy feature in the classification process, which our approach was able to catch. For *http.request_version*, some work shows this feature's usefulness in performing fingerprinting [85].

Table 5. DoS GoldenEye selected features.

Features	DT	RF	GNB	MLP
http.cache_control	✓	✓	✓	✓
tcp.window_size	✓	✓	✓	✓
http.accept_encoding	✓	✓		
http.connection			✓	✓
http.request.version			✓	✓
tcp.flags.push				✓
http.content_length_header				✓
tcp.options.sack_perm			✓	
tcp.flags.syn			✓	
tcp.analysis.duplicate_ack_num		✓		
Σ (features)	3	4	6	6

4.3. DoS HULK

Figure 4 showcases the accuracy achieved using our approach in classifying the DoS HULK attack. Starting with the IP-only headers, we observe certain resemblances between the two HTTP denial-of-service attacks and some minor distinctions. For the DT classifier, the selected features and all features resulted in the same maximum accuracy of 46%. Similarly, the RF classifier achieved a maximum accuracy of 49% for both selected and all features. In the case of the GNB classifier, both selected features and all features led to a maximum accuracy of 75%. On the other hand, the MLP classifier obtained a maximum accuracy of 34% for both selected and all features, as seen in Table 6.

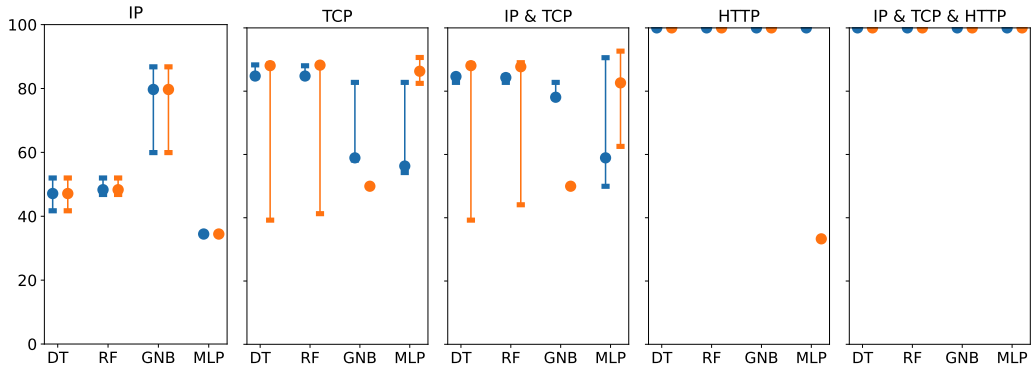

Figure 4. DoS HULK identification.

Table 6. DoS HULK selected features.

Features	DT	RF	GNB	MLP
ip.len	✓	✓	✓	✓
tcp.len	✓	✓	✓	✓
tcp.analysis.flags	✓	✓	✓	✓
http.connection	✓	✓	✓	✓
http.accept_encoding	✓	✓	✓	✓
http.accept		✓		✓
http.cache_control	✓			
Σ (features)	6	6	5	6

Looking at the TCP-only headers, the DT classifier reached the maximum accuracy of 85% with selected features, while the accuracy of all features was 71%. The features selected with this classifier were consistent across multiple runs. Similarly, the RF classifier achieved a maximum accuracy of 85% with selected features and 72% when using all features. The accuracy was consistently high. In the case of the GNB classifier, the maximum accuracy with selected features was 66%, whereas it was 49% for all features. On the other hand, the MLP classifier achieved a maximum accuracy of 64% with selected features and 86% with all features. This classifier's accuracy varied notably between the two feature sets. We observe a significant increase in classification accuracy across three algorithms, indicating the contribution of GA in selecting features that increase efficiency and accuracy by eliminating noisy data.

The HTTP-only header classification showed a consistent 100% accuracy for the DT, RF, and GNB classifiers, with the selected and all features indicating consistency across multiple runs. In contrast, the MLP classifier achieved a maximum accuracy of 100% with selected features, but the accuracy dropped to 33% when using all features. The MLP classifier exhibited a significant variation in accuracy between selected and all features. Across all classifiers, the most common theme was the high accuracy, with three classifiers consistently achieving a perfect accuracy of 100%.

A significant overlap exists between the features selected by our algorithms in Table 5 and those chosen for the DoS Http Unbearable Load King (HULK) attack, as depicted

in Table 6. This outcome is expected, since both DoS HULK and DoS GoldenEye have the same objective regarding the attack. In the exclusive IP header-only classification, the sole feature, *ip.len*, was consistently chosen among all features. Although there is limited existing research highlighting a correlation that establishes the importance of this feature in the classification of DoS HULK attacks, relying solely on this feature did not yield significant results. In the TCP header-only classification, we observe improved accuracy, notably for certain algorithms such as DT and RF, utilizing Genetic Algorithms (GA), which contributed to narrowing the gap. In the case of GNB, employing GA led to enhanced accuracy. Another feature picked, such as *tcp.analysis.flags* from Wireshark's website, is a field that analyzes the metrics of the flags being used [69]. Exploring the information associated with this particular feature reveals the presence of the following indication: "Expert Info (Warning/Sequence): Previous segment(s) not captured (common at capture start)". This warning is triggered when Wireshark detects an acknowledgment for a packet it did not capture, potentially indicating packet loss [69]. The manifestation of this feature in the data after a certain duration of the attack suggests it could serve as additional evidence supporting the possibility of packet loss. Much like *ip.len*, there is a lack of dedicated research linking this flag directly to a Denial-of-Service attack. Given that this is a Wireshark-specific feature, it may go unnoticed by other packet-capturing software, such as *tcpdump*. A characteristic, namely *tcp.len*, was identified by GA but was absent in DoS GoldenEye. This feature, recognized unanimously by all algorithms, is considered distinctive. Some studies, such as those by Sanchez et al. [86] and Maliha et al. [87], examine *packet.length* as a relevant feature in the detection of incidents involving DoS HULK, Low Orbit Ion Cannon (LOIC), and GoldenEye. While not identical to *tcp.len*, *packet.length* computes segment size, potentially establishing similarities with other features chosen in GoldenEye, such as *tcp.window_size*. We observe multiple HTTP features overlapping in their selection due to their high accuracy, mirroring the pattern in the DoS GoldenEye HTTP features. The *http.cache_control* feature is implemented to bypass a server's caching engine, as detailed by Mahjabin et al. [18]. Despite limited research attributing *http.connection* and *http.accept_encoding* to DoS HULK, the inclusion of these features is justified by the existence of a similar attack and studies illustrating the correlation between these attributes and DoS GoldenEye [22,83,84].

4.4. DNS Botnet

Starting with IP header-only analysis, utilizing the Decision Tree (DT) classifier, the maximum accuracy of selected features reached 53%, while considering all features yielded a maximum accuracy of 59%. Similarly, the Random Forest (RF) classifier achieved a maximum accuracy of 38% for selected features and 37% for all features. Gaussian Naive Bayes (GNB) demonstrated a maximum accuracy of 31% for selected features and 29% for all features. Multi-layer Perceptron (MLP) showcased a maximum accuracy of 50% for selected features and 34% for all features. Across all classifiers, a consistent pattern emerged with varying features being recurrently selected, reflecting the distinctive preferences of each algorithm.

Looking specifically at the UDP protocol, DT achieved a maximum accuracy of 37% with selected features and 35% with all features. Similarly, the RF classifier obtained a maximum accuracy of 37% with selected features and 34% with all features. GNB demonstrated a maximum accuracy of 39% for both selected and all features. The MLP exhibited a maximum accuracy of 40% for both selected and all features. As seen in Figure 5, IP, UDP, and their union yielded the lowest sets of accuracy results, indicating their lack of contribution to the classification of DNS Botnet attack.

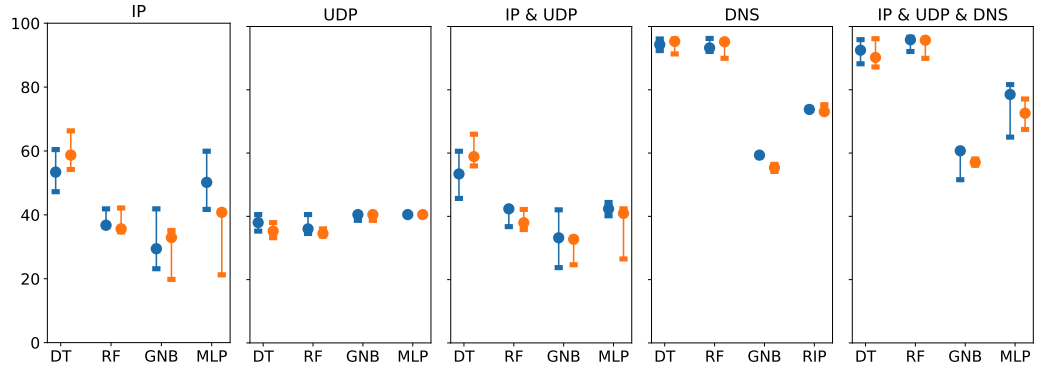

Figure 5. DNS botnet identification.

In DNS classification, however, we observed higher accuracy results. DT achieved a maximum accuracy of 94% for selected features and 94% for all features. The RF classifier demonstrated a maximum accuracy of 93% for selected features and 93% for all features. GNB exhibited a maximum accuracy of 59% for selected features and 55% for all features. The MLP showcased a maximum accuracy of 74% for selected features and 73% for all features. Across all classifiers, the most prominent accuracy rates were achieved by the DT for selected features and RF for all features.

Similar to our analysis of other cyber attacks using our methodology, we observe that *ip.len* is again identified as the predominant feature by all algorithms, as depicted in Table 7. Such coherence arises because the IP length also contains the DNS length, which is the driving force in classifying DNS Botnet attacks. Regarding the significant role of *udp.length* in detecting DNS botnets, existing research, such as the study by Khaing et al. [88], indicates that the UDP packet size can serve as a means to identify DNS botnets within IoT due to its inclusion of DNS header length. Additionally, related studies have utilized it with other features to identify hybrid botnets, as highlighted in [89]. Examining the DNS component of our findings, we observe a pattern similar to other attacks, where the most pronounced impact and the highest achieved accuracy correlate with selecting the most significant number of features. We begin by looking at the most widely researched DNS botnet features: *dns.qry.name* and *dns.resp.primaryname*. Numerous works, including those by Manmeet Singh et al., delve into the intricacies of DNS queries employed by botnets [6]. Kamal Alieyan et al., for instance, emphasize the exclusive use of the DNS feature *dns.qry.name* for anomaly detection [7]. This observation aligns with broader research efforts, such as those led by Manos Antonakakis et al., where an anomaly-based detection system is developed, focusing on query names and leveraging passive DNS query data [23]. Furthermore, the UCSD research by Xiaobo Ma et al. contributes valuable insights into DNS query characteristics through active probing on a large scale [24]. Similarly, Xiaobo Ma et al. actively probe DNS query characteristics by assessing DNS cache activities [24]. In line with the theme of monitoring DNS activities, Xuan Dau Hoang and Quynh Chi Nguyen underscore the importance of monitoring DNS query data to unveil malicious activities [58]. Examining outbound DNS queries, Yong Jin et al. identify potential anomalies [59]. At the same time, the research by Kamal Alieyan et al. proposes an approach based on DNS query and response behaviors for detecting abnormal activities [7]. We have also demonstrated the contribution of DNS protocol in identifying Fast-flux attacks in our previous research [31]. These collective efforts underscore the significance of understanding the behavior within these features to distinguish malicious botnet activity from benign network behavior. The *dns.rrsig.signature* is a component of DNS security and, as of now, there are no studies linking this feature to any form of malicious botnet activity. *dns.soa.serial_number* is a feature that plays a crucial role in DNS zone transfers, as it is responsible for informing the secondary nameserver about the need for an update in its version. In transferring DNS record data from a primary to a secondary nameserver, the start of authority (SOA) record containing the serial number is sent first. This serial number is crucial for the

secondary server to determine if an update is required. T. Frosch et al. reference this feature, noting that the frequency of SOA serial number changes is a distinctive characteristic indicating the rate at which a DNS zone is updated. Their analysis suggests variations in serial number changes between benign and malicious zones, leading them to employ this metric as a distinguishing feature in their research [90]. The presence of the flag *dns.flags.checkdisable* in a query signals that the resolver sending the query finds non-authenticated data acceptable. Despite being selected by most algorithms, except for GNB, there need to be studies demonstrating a correlation between DNS Botnets and the utilization of this feature. A specific feature in the DNS query field *dns.qry.type* is associated with DNS Botnet activity as mentioned in papers [58,60]. Alieyan et al. mention that certain domains exhibit illogical and unnatural structures featuring names that lack a sequential pattern. Distinguishing between normal behavior in proper DNS transactions or queries and abnormal ones becomes evident, especially when many queries are involved [60]. Considering that this feature was selected only by the two algorithms with lower accuracy, it likely does not significantly impact DNS Botnet detection: the two remaining features, namely *dns.resp.name* and *dns.resp.ns*, similar to the previously mentioned features, lack substantial dedicated research demonstrating a clear correlation with DNS Botnets.

Table 7. DNS botnet selected features.

Features	DT	RF	GNB	MLP
ip.len	✓	✓	✓	✓
udp.length	✓	✓	✓	✓
dns.qry.name	✓	✓	✓	✓
dns.rrsig.signature	✓	✓	✓	✓
dns.soa.serial_number	✓	✓		✓
dns.flags.checkdisable	✓	✓		✓
dns.resp.ns	✓		✓	✓
dns.resp.primaryname			✓	✓
dns.resp.name			✓	✓
dns.qry.type			✓	✓
Σ (features)	7	6	8	10

5. Conclusions

This study underscores the effective integration of machine learning algorithms such as Decision Tree (DT), Random Forest (RF), Gaussian Naive Bayes (GNB), and Multi-layer Perceptron (MLP) with genetic algorithms (GA) for discerning distinctive features associated with specific programs or attacks. By gaining insights into these unique characteristics, we enhance our understanding of the nature of attacks, mainly focusing on Operating System (OS) scanning, Domain Name System (DNS) Botnet, Denial of Service (DoS) Http Unbearable Load King (HULK), and DoS GoldenEye. GA plays a crucial role in automatically selecting a subset of features, significantly improving the precision of classifying benign and malicious packets. Feature selection is essential for filtering out irrelevant elements that may introduce noise during data classification. Notably, our observations indicate that, in certain instances, utilizing GA results in higher accuracy than using all features collectively. Furthermore, the elimination of noise facilitates quicker data import, accelerating the classification process and enabling a focus solely on the features relevant to the attack. Based on our discoveries, the features selected by GA for OS scanning in Internet Protocol (IP) and Transmission Control Protocol (TCP) headers achieved a high accuracy of 94%. For DoS GoldenEye, utilizing only the HyperText Transfer Protocol (HTTP) header resulted in a perfect 100% accuracy, as did DoS HULK with the HTTP header alone. Similarly, for DNS Botnet, employing only the DNS header yielded an accuracy of 94%. These outcomes were achieved by employing a minimal subset of features, varying from 3 to 10, out of over 60 potential features for HTTP, over 300 for DNS, and over 300 for combined IP and TCP headers.

The consistency between features selected by GA and their contribution to identifying attacks underscores the effectiveness of this approach. In our future analysis, we aim to delve into the statistical features of each header field of various protocols to enhance our classification accuracy. While packet rate and timing are known to improve the detection of specific attacks, we aim to refine our understanding of these features for better identification and differentiation of attack types. Additionally, a more thorough examination of packet contents could unveil further attack patterns concealed within the packet features. We also want to compare our classification results with other search optimization algorithms to observe any potential increase in the accuracy and performance of our approach.

Author Contributions: Conceptualization, A.A., L.V. and G.K.; Methodology, A.A. and L.V.; Software, A.A.; Validation, A.A., L.V. and G.K.; Formal analysis, A.A.; Investigation, A.A.; Resources, A.A.; Data curation, L.V.; Writing—original draft, A.A. and L.V.; Writing—review and editing, A.A., L.V. and G.K.; Supervision, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data supporting the results reported in this article are openly available on our dedicated Kaggle repository at date https://www.kaggle.com/datasets/exitium505/dos-goldeneye-and-hulk-pcaps (accessed on 5 January 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Huang, H.; Ahmed, N.; Karthik, P. On a New Type of Denial of Service Attack in Wireless Networks: The Distributed Jammer Network. *IEEE Trans. Wirel. Commun.* **2011**, *10*, 2316–2324. [CrossRef]
2. Palmieri, F.; Ricciardi, S.; Fiore, U.; Ficco, M.; Castiglione, A. Energy-oriented denial of service attacks: An emerging menace for large cloud infrastructures. *J. Supercomput.* **2015**, *71*, 1620–1641. [CrossRef]
3. Xu, Y.; Deng, G.; Zhang, T.; Qiu, H.; Bao, Y. Novel denial-of-service attacks against cloud-based multi-robot systems. *Inf. Sci.* **2021**, *576*, 329–344. [CrossRef]
4. Asri, S.; Pranggono, B. Impact of distributed denial-of-service attack on advanced metering infrastructure. *Wirel. Pers. Commun.* **2015**, *83*, 2211–2223. [CrossRef]
5. Hagos, D.H.; Yazidi, A.; Kure, O.; Engelstad, P.E. A Machine-Learning-Based Tool for Passive OS Fingerprinting with TCP Variant as a Novel Feature. *IEEE Internet Things J.* **2021**, *8*, 3534–3553. [CrossRef]
6. Singh, M.; Singh, M.; Kaur, S. Issues and challenges in DNS based botnet detection: A survey. *Comput. Secur.* **2019**, *86*, 28–52. [CrossRef]
7. Alieyan, K.; ALmomani, A.; Manasrah, A.; Kadhum, M.M. A survey of botnet detection based on DNS. *Neural Comput. Appl.* **2017**, *28*, 1541–1558. [CrossRef]
8. Shorey, T.; Subbaiah, D.; Goyal, A.; Sakxena, A.; Mishra, A.K. Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools. In Proceedings of the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 19–22 September 2018; pp. 318–322. [CrossRef]
9. Cameron, C.; Patsios, C.; Taylor, P.C.; Pourmirza, Z. Using Self-Organizing Architectures to Mitigate the Impacts of Denial-of-Service Attacks on Voltage Control Schemes. *IEEE Trans. Smart Grid* **2019**, *10*, 3010–3019. [CrossRef]
10. Chahal, J.K.; Bhandari, A.; Behal, S. Distributed Denial of Service Attacks: A Threat or Challenge. *New Rev. Inf. Netw.* **2019**, *24*, 31–103. [CrossRef]
11. Albanese, M.; Battista, E.; Jajodia, S. A deception based approach for defeating OS and service fingerprinting. In Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS), Florence, Italy, 28–30 September 2015; pp. 317–325. [CrossRef]
12. Liao, S.; Zhou, C.; Zhao, Y.; Zhang, Z.; Zhang, C.; Gao, Y.; Zhong, G. A Comprehensive Detection Approach of Nmap: Principles, Rules and Experiments. In Proceedings of the 2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Chongqing, China, 29–30 October 2020; pp. 64–71. [CrossRef]
13. Li, R.; Sosnowski, M.; Sattler, P. An overview of os fingerprinting tools on the internet. *Network* **2020**, *73*, 73–77.
14. Acosta, J.C.; Basak, A.; Kiekintveld, C.; Kamhoua, C. Lightweight On-Demand Honeypot Deployment for Cyber Deception. In *Digital Forensics and Cyber Crime*; Gladyshev, P., Goel, S., James, J., Markowsky, G., Johnson, D., Eds.; Springer: Cham, Switzerland, 2022; pp. 294–312.

15. Khera, Y.; Kumar, D.; Sujay; Garg, N. Analysis and Impact of Vulnerability Assessment and Penetration Testing. In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 14–16 February 2019; pp. 525–530. [CrossRef]
16. Brooks, R.R.; Yu, L.; Ozcelik, I.; Oakley, J.; Tusing, N. Distributed Denial of Service (DDoS): A History. *IEEE Ann. Hist. Comput.* **2022**, *44*, 44–54. [CrossRef]
17. Khalaf, B.A.; Mostafa, S.A.; Mustapha, A.; Mohammed, M.A.; Abduallah, W.M. Comprehensive Review of Artificial Intelligence and Statistical Approaches in Distributed Denial of Service Attack and Defense Methods. *IEEE Access* **2019**, *7*, 51691–51713. [CrossRef]
18. Mahjabin, S. Implementation of DoS and DDoS attacks on cloud servers. *Period. Eng. Nat. Sci.* **2018**, *6*, 148–158. [CrossRef]
19. Catillo, M.; Pecchia, A.; Villano, U. No more DoS? An empirical study on defense techniques for web server Denial of Service mitigation. *J. Netw. Comput. Appl.* **2022**, *202*, 103363. [CrossRef]
20. Nayyar, S.; Arora, S.; Singh, M. Recurrent Neural Network Based Intrusion Detection System. In Proceedings of the 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 28–30 July 2020; pp. 136–140. [CrossRef]
21. Gokul Pran, S.; Raja, S. An efficient feature selection and classification approach for an intrusion detection system using Optimal Neural Network. *J. Intell. Fuzzy Syst.* **2023**, *44*, 8561–8571. [CrossRef]
22. Kshirsagar, D.; Shaikh, J.M. Intrusion detection using rule-based machine learning algorithms. In Proceedings of the 2019 5th International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 19–21 September 2019; pp. 1–4.
23. Antonakakis, M.; Perdisci, R.; Dagon, D.; Lee, W.; Feamster, N. Building a dynamic reputation system for {DNS}. In Proceedings of the 19th USENIX Security Symposium (USENIX Security 10), Washington, DC, USA, 11–13 August 2010.
24. Ma, X.; Zhang, J.; Li, Z.; Li, J.; Tao, J.; Guan, X.; Lui, J.C.; Towsley, D. Accurate DNS query characteristics estimation via active probing. *J. Netw. Comput. Appl.* **2015**, *47*, 72–84. [CrossRef]
25. Vargas, L.; Blue, L.; Frost, V.; Patton, C.; Scaife, N.; Butler, K.R.; Traynor, P. Digital Healthcare-Associated Infection: A Case Study on the Security of a Major Multi-Campus Hospital System. In Proceedings of the NDSS, San Diego, CA, USA, 24–27 February 2019.
26. Ghafir, I.; Prenosil, V.; Hammoudeh, M.; Baker, T.; Jabbar, S.; Khalid, S.; Jaf, S. BotDet: A System for Real Time Botnet Command and Control Traffic Detection. *IEEE Access* **2018**, *6*, 38947–38958. [CrossRef]
27. Kumar, A.; Sharma, I. Augmenting IoT Healthcare Security and Reliability with Early Detection of IoT Botnet Attacks. In Proceedings of the 2023 4th International Conference for Emerging Technology (INCET), Belgaum, India, 26–28 May 2023; pp. 1–6. [CrossRef]
28. Quamara, M.; Gupta, B.B.; Yamaguchi, S. An End-to-End Security Framework for Smart Healthcare Information Sharing against Botnet-based Cyber-Attacks. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 10–12 January 2021; pp. 1–4. [CrossRef]
29. Rana, S.; Aksoy, A. Automated Fast-flux Detection using Machine Learning and Genetic Algorithms. In Proceedings of the IEEE INFOCOM 2021—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Vancouver, BC, Canada, 10–13 May 2021; pp. 1–6.
30. Aksoy, A.; Louis, S.; Gunes, M.H. Operating system fingerprinting via automated network traffic analysis. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017; pp. 2502–2509.
31. Aksoy, A.; Gunes, M.H. Automated iot device identification using network traffic. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–7.
32. Zalewski, M. *p0f v3*, Version 3.07b. Available online: https://lcamtuf.coredump.cx/p0f3/ (accessed on 5 January 2024).
33. Ornaghi, A.; Valleri, M. Ettercap. 2003. Available online: http://ettercap.sourceforge.net/index.php?s=home (accessed on 5 January 2024).
34. Subterrain Security Group. Siphon Project. 2000. Available online: https://github.com/unmarshal/siphon (accessed on 5 January 2024).
35. Ustebay, S.; Turgut, Z.; Aydin, M.A. Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier. In Proceedings of the 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), Ankara, Turkey, 3–4 December 2018; pp. 71–76. [CrossRef]
36. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the International Conference on Information Systems Security and Privacy, Funchal, Portugal, 22–24 January 2018.
37. Miettinen, M.; Marchal, S.; Hafeez, I.; Asokan, N.; Sadeghi, A.R.; Tarkoma, S. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; pp. 2177–2184. [CrossRef]
38. Thalji, N.; Raza, A.; Islam, M.S.; Samee, N.A.; Jamjoom, M.M. AE-Net: Novel Autoencoder-Based Deep Features for SQL Injection Attack Detection. *IEEE Access* **2023**, *11*, 135507–135516. [CrossRef]
39. Sajid. SQL Injection Dataset. Available online: https://www.kaggle.com/datasets/sajid576/sql-injection-dataset (accessed on 5 January 2024).

40. Awad, M.; Fraihat, S. Recursive Feature Elimination with Cross-Validation with Decision Tree: Feature Selection Method for Machine Learning-Based Intrusion Detection Systems. *J. Sens. Actuator Netw.* **2023**, *12*, 67. [CrossRef]
41. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
42. Liu, X.; Du, Y. Towards Effective Feature Selection for IoT Botnet Attack Detection Using a Genetic Algorithm. *Electronics* **2023**, *12*, 1260. [CrossRef]
43. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset. *arXiv* **2018**, arXiv:1811.00701. Available online: http://arxiv.org/abs/1811.00701 (accessed on 5 January 2024).
44. Tyagi, R.; Paul, T.; Manoj, B.; Thanudas, B. Packet Inspection for Unauthorized OS Detection in Enterprises. *IEEE Secur. Priv.* **2015**, *13*, 60–65. [CrossRef]
45. Nuiaa, R.R.; Manickam, S.; Alsaeedi, A.H. Distributed reflection denial of service attack: A critical review. *Int. J. Electr. Comput. Eng.* **2021**, *11*, 5327. [CrossRef]
46. Rossow, C. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In Proceedings of the NDSS, San Diego, CA, USA, 23–26 February 2014; pp. 1–15.
47. Doriguzzi-Corin, R.; Siracusa, D. FLAD: Adaptive Federated Learning for DDoS Attack Detection. *arXiv* **2022**, arXiv:2205.06661. Available online: http://arxiv.org/abs/2205.06661 (accessed on 5 January 2024).
48. Ahmed, M.E.; Ullah, S.; Kim, H. Statistical application fingerprinting for DDoS attack mitigation. *IEEE Trans. Inf. Forensics Secur.* **2018**, *14*, 1471–1484. [CrossRef]
49. Hussain, A.; Heidemann, J.; Papadopoulos, C. A framework for classifying denial of service attacks. In Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Karlsruhe, Germany, 25–29 August 2003; pp. 99–110.
50. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 1–3 October 2019; pp. 1–8.
51. Tama, B.A.; Rhee, K.H. Attack classification analysis of IoT network via deep learning approach. *Res. Briefs Inf. Commun. Technol. Evol. (ReBICTE)* **2017**, *3*, 1–9.
52. Churcher, A.; Ullah, R.; Ahmad, J.; Ur Rehman, S.; Masood, F.; Gogate, M.; Alqahtani, F.; Nour, B.; Buchanan, W.J. An experimental analysis of attack classification using machine learning in IoT networks. *Sensors* **2021**, *21*, 446. [CrossRef] [PubMed]
53. Aamir, M.; Zaidi, S.M.A. Clustering based semi-supervised machine learning for DDoS attack classification. *J. King Saud-Univ.-Comput. Inf. Sci.* **2021**, *33*, 436–446. [CrossRef]
54. Ahuja, N.; Singal, G.; Mukhopadhyay, D.; Kumar, N. Automated DDOS attack detection in software defined networking. *J. Netw. Comput. Appl.* **2021**, *187*, 103108. [CrossRef]
55. Ayoade, G.; Chandra, S.; Khan, L.; Hamlen, K.; Thuraisingham, B. Automated threat report classification over multi-source data. In Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), Philadelphia, PA, USA, 18–20 October 2018; pp. 236–245.
56. Fernandes, G.; Owezarski, P. Automated classification of network traffic anomalies. In Proceedings of the Security and Privacy in Communication Networks: 5th International ICST Conference, SecureComm 2009, Athens, Greece, 14–18 September 2009; Revised Selected Papers 5; Springer: Berlin/Heidelberg, Germany, 2009; pp. 91–100.
57. Bhagwani, H.; Negi, R.; Dutta, A.K.; Handa, A.; Kumar, N.; Shukla, S.K. Automated classification of web-application attacks for intrusion detection. In Proceedings of the Security, Privacy, and Applied Cryptography Engineering: 9th International Conference, SPACE 2019, Gandhinagar, India, 3–7 December 2019; Proceedings 9; Springer: Berlin/Heidelberg, Germany, 2019; pp. 123–141.
58. Hoang, X.D.; Nguyen, Q.C. Botnet detection based on machine learning techniques using DNS query data. *Future Internet* **2018**, *10*, 43. [CrossRef]
59. Jin, Y.; Ichise, H.; Iida, K. Design of Detecting Botnet Communication by Monitoring Direct Outbound DNS Queries. In Proceedings of the 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing, New York, NY, USA, 3–5 November 2015; pp. 37–41. [CrossRef]
60. Alieyan, K.; Almomani, A.; Anbar, M.; Alauthman, M.; Abdullah, R.; Gupta, B.B. DNS rule-based schema to botnet detection. *Enterp. Inf. Syst.* **2021**, *15*, 545–564. [CrossRef]
61. Kannari, P.R.; Chowdary, N.S.; Laxmikanth Biradar, R. An anomaly-based intrusion detection system using recursive feature elimination technique for improved attack detection. *Theor. Comput. Sci.* **2022**, *931*, 56–64. [CrossRef]
62. Haseeb, J.; Mansoori, M.; Hirose, Y.; Al-Sahaf, H.; Welch, I. Autoencoder-based feature construction for IoT attacks clustering. *Future Gener. Comput. Syst.* **2022**, *127*, 487–502. [CrossRef]
63. Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: An ensemble of autoencoders for online network intrusion detection. *arXiv* **2018**, arXiv:1802.09089.

64. Valle, L. DoS GoldenEye & Hulk Network Attack PCAPs. 2023. Available online: https://www.kaggle.com/datasets/exitium505/dos-goldeneye-and-hulk-pcaps (accessed on 5 January 2024).
65. Traore, I.; Woungang, I.; Awad, A. *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*; Springer: Berlin/Heidelberg, Germany, 2018.
66. Paolini, E.; Valcarenghi, L.; Maggiani, L.; Andriolli, N. Real-time clustering based on deep embeddings for threat detection in 6G networks. *IEEE Access* **2023**, *11*, 115827–115835. [CrossRef]
67. Asad, M.; Asim, M.; Javed, T.; Beg, M.O.; Mujtaba, H.; Abbas, S. Deepdetect: Detection of distributed denial of service attacks using deep learning. *Comput. J.* **2020**, *63*, 983–994. [CrossRef]
68. Alenazi, A.; Traore, I.; Ganame, K.; Woungang, I. Holistic Model for HTTP Botnet Detection Based on DNS Traffic Analysis. In Proceedings of the Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments, Vancouver, BC, Canada, 26–28 October 2017; Traore, I., Woungang, I., Awad, A., Eds.; Springer: Cham, Switzerland, 2017; pp. 1–18.
69. Team, The Wireshark Developers Wireshark User's Guide. Wireshark. Available online: https://www.wireshark.org (accessed on 5 January 2024).
70. Aksoy, A.; Varma, S.; Moorthy, G.; Pan, E.; Kar, G. Comparative Analysis of Feature Selection Algorithms for Automated IoT Device Fingerprinting. In Proceedings of the 10th International Conference on Information Systems Security and Privacy (ICISSP), San Francisco, CA, USA, 26–28 February 2024.
71. Aksoy, A.; Gunes, M.H. Operating system classification performance of tcp/ip protocol headers. In Proceedings of the 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops), Dubai, United Arab Emirates, 7–10 November 2016; pp. 112–120.
72. Keahey, K.; Anderson, J.; Zhen, Z.; Riteau, P.; Ruth, P.; Stanzione, D.; Cevik, M.; Colleran, J.; Gunawi, H.S.; Hammock, C.; et al. Lessons Learned from the Chameleon Testbed. In Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20), Online, 15–17 July 2020; USENIX Association: Berkeley, CA, USA, 2020; pp. 219–233.
73. Zhang, Y.; Li, P.; Wang, X. Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network. *IEEE Access* **2019**, *7*, 31711–31722. [CrossRef]
74. Yang, D.; Yu, Z.; Yuan, H.; Cui, Y. An improved genetic algorithm and its application in neural network adversarial attack. *PLoS ONE* **2022**, *17*, e0267970. [CrossRef]
75. Hwang, J.; Kim, M. Effective Detecting Method of Nmap Idle Scan. *J. Adv. Inf. Technol. Converg.* **2019**, *9*, 1–10. [CrossRef]
76. Hartpence, B.; Kwasinski, A. Combating TCP port scan attacks using sequential neural networks. In Proceedings of the 2020 International Conference on Computing, Networking and Communications (ICNC), Big Island, HI, USA, 17–20 February 2020; pp. 256–260.
77. Naik, N.; Jenkins, P.; Savage, N. Threat-Aware Honeypot for Discovering and Predicting Fingerprinting Attacks Using Principal Components Analysis. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, 18–21 November 2018; pp. 623–630. [CrossRef]
78. Jetty, S.; Rahalkar, S. *Securing Network Infrastructure: Discover Practical Network Security with Nmap and Nessus 7*; Packt Publishing Ltd.: Birmingham, UK, 2019.
79. Naik, N.; Jenkins, P.; Savage, N.; Yang, L. A computational intelligence enabled honeypot for chasing ghosts in the wires. *Complex Intell. Syst.* **2021**, *7*, 477–494. [CrossRef]
80. Skowron, M.; Janicki, A.; Mazurczyk, W. Traffic Fingerprinting Attacks on Internet of Things Using Machine Learning. *IEEE Access* **2020**, *8*, 20386–20400. [CrossRef]
81. Rasheed, M.M.; Faieq, A.K.; Hashim, A.A. Development of a new system to detect denial of service attack using machine learning classification. *Indones. J. Electr. Eng. Comput. Sci.* **2021**, *23*, 1068–1072. [CrossRef]
82. Muraleedharan, N.; Janet, B. A deep learning based HTTP slow DoS classification approach using flow data. *ICT Express* **2021**, *7*, 210–214. [CrossRef]
83. Das, A. A deep transfer learning approach to enhance network intrusion detection capabilities for cyber security. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*. [CrossRef]
84. Soufiane, S.; Magan-Carrion, R.; Medina-Bulo, I.; Bouden, H. Preserving authentication and availability security services through multivariate statistical network monitoring. *J. Inf. Secur. Appl.* **2021**, *58*, 102785. [CrossRef]
85. Kar, A.; Natadze, A.; Branca, E.; Stakhanova, N. *HTTPFuzz: Web Server Fingerprinting with HTTP Request Fuzzing*; SciTePress: Setúbal, Portugal, 2022.
86. Sanchez, O.R.; Repetto, M.; Carrega, A.; Bolla, R.; Pajo, J.F. Feature selection evaluation towards a lightweight deep learning DDoS detector. In Proceedings of the ICC 2021—IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
87. Maliha, M. A supervised learning approach: Detection of cyber attacks. In Proceedings of the 2021 IEEE International Conference on Telecommunications and Photonics (ICTP), Dhaka, Bangladesh, 22–24 December 2021; pp. 1–5.
88. Khaing, M.S.; Thant, Y.M.; Tun, T.; Htwe, C.S.; Thwin, M.M.S. IoT botnet detection mechanism based on UDP protocol. In Proceedings of the 2020 IEEE Conference on Computer Applications (ICCA), Yangon, Myanmar, 27–28 February 2020; pp. 1–7.

89. Almutairi, S.; Mahfoudh, S.; Almutairi, S.; Alowibdi, J.S. Hybrid botnet detection based on host and network analysis. *J. Comput. Netw. Commun.* **2020**, *2020*, 9024726. [CrossRef]
90. Frosch, T.; Kührer, M.; Holz, T. Predentifier: Detecting botnet C&C domains from passive DNS data. *Adv. Early Warn.* **2013**, 78–90.