Journal of *Cybersecurity and Privacy*

MDPI

Article

Data-Driven Network Anomaly Detection with Cyber Attack and Defense Visualization

Eric Muhati * and Danda Rawat

Data Science and Cybersecurity Center (DSC^2), Department of Electrical Engineering and Computer Science, Howard University, Washington, DC 20059, USA
* Correspondence: eric.muhati@howard.edu

Abstract: The exponential growth in data volumes, combined with the inherent complexity of network algorithms, has drastically affected network security. Data activities are producing voluminous network logs that often mask critical vulnerabilities. Although there are efforts to address these hidden vulnerabilities, the solutions often come at high costs or increased complexities. In contrast, the potential of *open-source* tools, recognized for their security analysis capabilities, remains under-researched. These tools have the potential for detailed extraction of essential network components, and they strengthen network security. Addressing this gap, our paper proposes a data analytics-driven network anomaly detection model, which is uniquely complemented with a visualization layer, making the dynamics of cyberattacks and their subsequent defenses distinctive in near real-time. Our novel approach, based on network scanning tools and *network discovery services*, allows us to visualize the network based on how many IP-based networking devices are *live*, then we implement a data analytics-based intrusion detection system that scrutinizes all network connections. We then initiate mitigation measures, visually distinguishing malicious from benign connections using red and blue hues, respectively. Our experimental evaluation shows an F_1 score of 97.9% and a minimal false positive rate of 0.3% in our model, demonstrating a marked improvement over existing research in this domain.

Keywords: anomaly detection; data analytics; TCP connections; cyber visualization

Citation: Muhati, E.; Rawat, D. Data-Driven Network Anomaly Detection with Cyber Attack and Defense Visualization. *J. Cybersecur. Priv.* **2024**, *4*, 241–263. https://doi.org/10.3390/jcp4020012

Academic Editors: Feng Wang and Yongning Tang

Received: 20 January 2024
Revised: 5 April 2024
Accepted: 8 April 2024
Published: 9 April 2024

1. Introduction

Despite significant investments in cybersecurity infrastructures, the occurrence of successful cyberattacks remains significant. The evolving landscape of cyber threats consistently leans in favor of attackers, as they exploit the intricate nature of traditional network protocols and adapt rapidly to emerging security mechanisms [1]. This prevailing imbalance is exacerbated by defenders often grappling with complex and overly specialized cybersecurity tools, which, while sophisticated, can introduce overheads that result in slower response times and fragmented investigative processes. Moreover, the sheer intricacy of such tools can overwhelm defenders, inadvertently providing a window of opportunity for cyber adversaries [2]. In response to these escalating threats, our research is driven by research questions on how can we enhance real-time anomaly detection through improved, low-cost data extraction and visualization techniques? In addition, what role can effective visualization play in identifying and responding to prevalent threats? These questions underscore the urgent need for cybersecurity tools that are not only advanced in their capabilities but are also accessible and efficient in practice.

Network intrusion detection systems (NIDSs) play a crucial role in detecting and mitigating network compromises. Data analytics-based NIDSs utilize algorithms and large datasets to identify malicious activities, offering a promising approach to cybersecurity. However, the dynamic nature of cyber threats presents challenges. Many existing NIDSs struggle with timely updates and adaptation to novel malware patterns, leading to potential vulnerabilities [3]. Moreover, the responses of some systems can inadvertently

reveal defensive mechanisms to attackers, providing them with information that could be exploited in future attacks [4]. Additionally, the vast amount of data generated by modern networked systems can strain the processing capabilities of NIDSs, affecting their real-time detection performance [5].

In response to these challenges, there is a growing inclination in the cybersecurity domain towards the integration of visualization techniques with traditional NIDSs [6–8]. Despite this interest, many current visualization approaches fall short in effectively representing the technical ramifications of cyberattacks [6–8]. There remains a significant need for a cyber visualization tool that seamlessly integrates with NIDS alerts and provides real-time comprehensive insights into the threat landscape.

In the context of these cybersecurity challenges, a synergy between streamlined abstraction and the visualization of security events may offer a robust solution for network anomaly detection [9]. The adversarial tactics, techniques, and procedures detailed in [10] span seven distinct phases. Within this schema, adversaries first undertake reconnaissance, subsequently weaponize malware, devise a delivery mechanism, exploit identified vulnerabilities, and eventually escalate and sustain the attack while maintaining persistent network connectivity. Notably, the execution of the latter five stages hinges on a successful breach, emphasizing the significance of network connectivity and the absence of effective NIDS intervention. Given that adversarial actions invariably leave detectable footprints within network traffic [11], an incisive focus on specific traffic patterns and behaviors can prevent sub-optimal network traffic analysis. "Footprints" are signs that indicate potential security threats within network traffic, including unusual patterns like spikes in traffic or unexpected request types, which can suggest malicious activities. For instance, if a server typically receives 100 connection requests per time interval but suddenly receives $10,000$ requests/probes as an outlier considering all other network constants, that can be a clear footprint of a potential distributed denial of service (DDoS) attack.

In light of the aforementioned context, our research introduces a novel model characterized by the following model objectives (`MO`):

- `MO-1`: We present a low-cost and straightforward technique for extracting network data using *open-source* scanning tools designed to dynamically capture network connections, facilitating the efficient identification of potentially malicious hosts.
- `MO-2`: We automate the network data extraction mechanism to ensure its replicability for subsequent research endeavors.
- `MO-3`: We overlay our anomaly detection framework with a cyber visualization layer, capturing benign or malicious connections. This visualization granularity is derived from computed attack and defense metrics, as illustrated in Figure 1.
- `MO-4`: Finally, we incorporate proactive mitigation strategies within our visual module, where link hues transition from red to blue, signifying the successful counteraction of cyber threats, and conversely from blue to red upon detecting malicious activities.

Our proposed model case study can be described as an analysis of temporal and behavioral patterns' network traffic, distinguishing between normal operations and potential threats. An anomaly is detected when a surge in failed connection attempts is observed, deviating from a baseline of established connections typically seen from benign hosts. Unlike traditional systems that might flag this as a mere spike in traffic, our proposed model assesses the failure rate in conjunction with established connections to flag attack attempts.This is also summarized in Figure 1 to depict the novelty of the model's ability to accurately identify threats by analyzing nuanced patterns of network interactions, thereby giving an improved and effective NIDS.

The essence of our paper revolves around addressing specific and prevalent threats related to DoS and malware probing. It is pertinent to emphasize that while the cybersecurity domain is vast and multifaceted, zooming in on specific threat vectors offers the advantage of depth, precision, and optimized solutions. While there exists a multitude of cyber threats like malware command and control, command/SQL injection, and drive-by download attacks, our targeted approach is towards DoS and malware probing threats. This focus is substantiated by evidence of their disruptive capabilities on cyber–physical systems, as highlighted in [12]. DoS attacks, for example, are shown to significantly impair communications between physical and computing systems, thereby demonstrating their potential to disrupt operations. Focusing our analytical techniques on these specific threat vectors allows us to deliver a highly refined and effective detection mechanism [12].

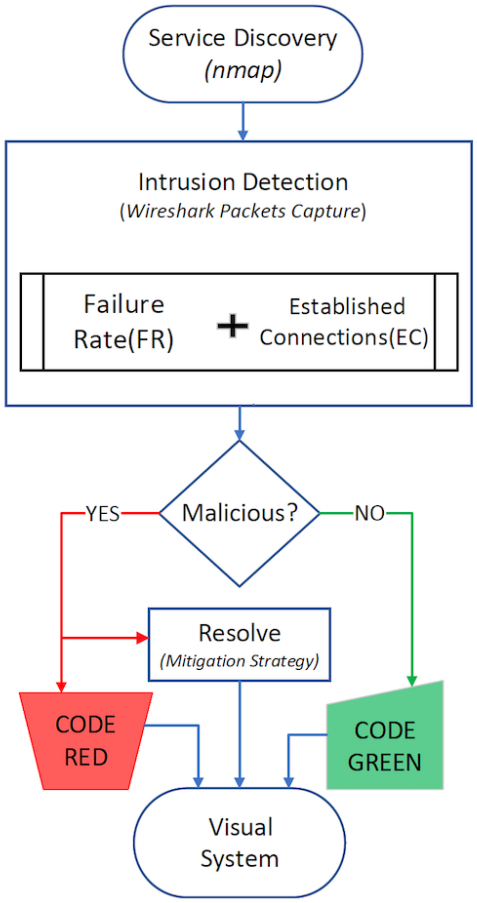

Figure 1. Model flow diagram.

TCP connectivity-based analysis, in particular, has shown to be adept at uncovering the subtle nuances and patterns related to these threats [13]. Integrating machine learning (ML) techniques or rule-based detection mechanisms can provide an enriched detection landscape [1]. However, introducing them without a well-defined scope might lead to a dilution of precision, increased false positives and potential inefficiencies [3,11]. Our current approach sets a strong foundation upon which subsequent research can build, expanding to other threats and incorporating diverse detection mechanisms. TCP connectivity-based analysis, in particular, has shown to be adept at uncovering the subtle nuances and patterns related to these threats.

The remainder of this paper is organized as follows: Section 2 delineates the existing research gap; Section 3 provides the foundational background; Section 4 details our model's implementation. Our performance evaluation and its empirical results are presented in Section 5, while Section 6 describes the design of our visual prototype. We conclude the paper in Section 7.

2. Related Work

Network complexities and cost implications are pivotal in network security models. Due to technical challenges, including cost and network complexities, many existing models are grouped on techniques, algorithms, system types, and available datasets [14]. Our spectrum of scope in this section considers approaches based on network data types and, more so, transmission control protocol (TCP) connections that form the crux of several NIDS approaches. To distinguish between these, we categorize them based on their treatment of TCP data in terms of feature extraction and classification. Approaches such as [15–19] emphasize extracting features from TCP connections and labeling data. These processed data are then utilized to classify network traffic as malicious or benign.

- Anomaly Detection with AdaBoost: The AdaBoost-based solution [15] deploys an online AdaBoost algorithm, leveraging both extracted network data and global detectors.
- Boosted Decision Trees: Both Ruan et al. [16] and the Hierarchical SOM method [17] use a similar foundational model but enrich it by integrating boosted decision trees, leading to multiple decision classifiers.
- Support Vector Machines (SVM): refs. [18,19] exploit the SVM algorithm post extensive training from TCP feature extracted data. Particularly, ref. [19] melds kernel principal component analysis and the genetic algorithm, while the multi-level hybrid SVM and extreme learning machine (ELM) introduces a modified *k*-means algorithm.

Each method offers its unique strengths, but a common lacuna is the absence of an affordable, nonproprietary, *on-the-fly* data probing and collection mechanism that operates sans extensive training. Our proposal fills this gap using *open-source service discovery tools* for network extraction and intrusion detection. Additionally, our model does not require extensive training and is differentiated with a visualization module offering enhanced portrayal of NIDS outputs.

Addressing SYN-ACK-based attacks is a topic of much debate. The brute-force method, as described in [20], aims to inflate the data structure for TCP connections awaiting acknowledgment, making it cumbersome for average attacker requests to sustain bandwidth constraints. While effective, this method is not without its pitfalls. The survey [21] flags issues like sluggish response times during connection look-ups, owing to the sheer volume of TCP data structures housed in vast spans of protected kernel memory. Moreover, unwarranted memory allocation might inadvertently modify TCP signaling behaviors. Our proposal counters these challenges by offering an economical and simplified approach. By monitoring TCP connections via open-source tools, we boost intrusion detection efficiency, merging the dual strengths of combined failure rates and pre-established TCP connections.

Visualization remains an under-explored territory in intrusion detection. Our method stands out by integrating a visual module that augments information processing, an emphasis not universally applied. For instance, Bhardwaj and Singh [22] leverage supervised data mining for network data extraction but offer limited details on live packet capture parameters, performance metrics, or validation data. In contrast, our approach ensures transparency by utilizing *service discovery tools* for automated network data mining.

Other notable endeavors include Ohnof et al.'s [23] methodology, which focuses on visualizing IP addresses related to cyberattacks by analyzing attack timing, source, and type with network sensors yet primarily targets Internet-worm attacks with a narrow visual scope. Ulmer et al. [24] develop a web-based prototype analyzing Geo-IP data changes to identify unusual behavior, focusing on historical threat source visualization rather than networked assets' current state. Our approach surpasses these models by not just visualizing security events for informed cyber decisions but also a visual representation of the extent to which cyber assets are affected, showing the total effects of cyberattacks (TCE).

3. Background

3.1. Reliable Network Reconstruction

To improve the reliability of network data reconstructions, advanced filtering algorithms can be employed to reduce noise, while anomaly detection techniques can help in identifying erroneous data patterns. However, challenges arise when statistical analyses, like packet sniffing in [22], yield data with compromised timeliness, making them unsuitable for many research applications [25]. True reconstruction of mutable network entities remains a demanding task [26]. Efforts like [27] grappled with these issues but depended heavily on extensive training data, which might not always be available. Thus, in our pursuit of `MO-1` and `MO-2`, we utilize *nmap* [28] as a more efficient alternative that bypasses the need for vast training datasets. Nmap is an open-source tool widely used for high-speed network scans using raw IP packets to determine running service applications, operating system (OS) versions, and type of packet filter/firewall network device.

3.2. Connectivity

An actual network process is difficult to simulate because of enormous infrastructures and immeasurable nonlinear, nonstationary variables, whereas an inaccurate representation of network statistical characteristics could induce errors in our proposed model performance, as stated in [29]. Simulating a precise network process is inherently challenging due to the sheer size of the infrastructures involved and the unpredictable nature of nonlinear, non-stationary variables. Moreover, an erroneous portrayal of network statistical attributes can detrimentally affect our model's performance, as mentioned in [29]. We navigate these intricacies by operating under the presumption that every TCP connection undergoes a three-way handshake for benign data transmission between a client and target host. Notably, specific connection parameters are essential precursors to the actual data exchange. Typically, certain connection parameters must be established before the exchange of application data.

The client host randomly picks a sequence number a and sends a synchronize (SYN) packet with supplementary TCP flags and options. The target host increments a by one, chooses its random sequence number b, appends distinct flags and options, and replies with a SYN Acknowledgement (SYN-ACK). The client host wraps up the handshake by adding one to both a and b and dispatching the final ACK packet. Upon a benign handshake, i.e., when $ACKFlag == 1$, both the client and target host are primed for data exchange, although either could be malicious. This insight equips our methodology to evade network performance mishaps affecting `MO-1` and `MO-2`, ensuring connectivity recognition after the last ACK is sent.

3.3. Threat Description

Although encryption algorithms have been relied on to protect sensitive data, the statistical patterns of underlying ciphers that successfully reveal encryption keys have become common [30], meaning that billions of terabytes of data transmitted daily over computer networks face many internet-based threats. In addition to the difficulty in ensuring a secure network environment, multi-agent systems are vulnerable to complex distributed attacks from multiple adversaries.

The traditional information security model characteristics, i.e., confidentiality, integrity, and availability, inspire our threat classification scope of understanding threats that vary from passive hidden communication lines that attack confidentiality or active adversaries that affect real-time data exchange. Both attacks aim at confidentiality, integrity, or availability categories. Our threat model considers both active and passive attackers who obtain unauthorized network connectivity. Our primary focus is on the availability of a computing resource flagged as unavailable when the device is under attack. Notably, although availability is the primary focus of our defense system, an attack connection could translate to the loss of confidentiality and integrity as well. Our focus is mainly on exposing the consequences of cyberattacks on the technical space, i.e., on computing resources [8].

We describe the following malware mechanisms that have severe high impacts.

3.3.1. Worms

These can cause greater damage to networks than viruses because, in a small burst of time, a worm attack can automatically infect millions of hosts [31]. Worms autonomously exploit vulnerabilities in the target system, as well as penetrate and replicate them subject to the conditions stated in Section 3.2. A worm propagation model involves the target acquisition, elevation of privilege, and infection phase [32]. Sophisticated worm attacks on both applications and protocols are quickly evolving and resulting in greater vulnerability exploits from minimized low-cost attack efforts [33,34].

3.3.2. TCP-SYN Flood Attack

This is a denial-of-service (DoS) attack to exhaust resources and render a targeted host unresponsive during the standard TCP three-way handshake. The attack involves repeatedly sending SYN packets to all target host ports, causing the target host to try to respond with an SYN-ACK packet from each open port. The result is that legitimate clients are denied service because of an overflow in the host's connection tables.

4. Methodology and Implementation

4.1. Automated Service Discovery

Information about the actual state of devices in a networked system can be obtained through network scanning. In this section, we explain `MO-2`, i.e., how our model actively probes and gathers network information *on the fly* without requiring extensive training data. We automate service discovery by compiling the following commands in a *bash script*. First, we dynamically discover the network IP class; secondly, we scan for all active IPs in the network, followed by a fingerprint scan for the OS version. This *bash script* enables us to extract a network topology *on the fly* based on active IPs as input to our model. Hence, our experiment can be easily replicated elsewhere as per `MO-2`, as follows:

```
 IPs = $(ip addr show | grep 'inet' | awk '{print $2}' | cut -d/ -f1 | head
-n 1)
```

Then, obtain a list of all active IPs

```
nmap -n -sn $IPs -oG - | awk '/Up$/print $2'
```

Loop through each IP and obtain the OS version'

```
nmap -O -T2 $Each_ip | grep "Device" | awk 'print $3' » scan.txt
```

Additionally, through *Wireshark* [35], a commonly available *open-source* network protocol analyzer, we execute remote packet capture to extract network traffic, applying $TCP.flags()$ SYN, ACK, and FIN to filter all established TCP connections per device. The Wireshark output becomes an input network model to our visualization system set to update in real time, depending on network activities. We selected Wireshark capture on the basis of the knowledge that the identification of connectivity from such a network-level view, same as stated in [36], would provide a realistic network model for Algorithm 1, instead of ineptly considering all relevant network aspects (e.g., user behavior and size).

Algorithm 1 Visualization of TCE (S, t)

Require: All hosts S, time-intervals t
Ensure: Total Cyberattack Effect (TCE%)

```
Start Timer
for i ← 1, t do
    for j ← 1, S do                          ▷ hosts ordered by IP address
        if Timer > t_i then                  ▷ Check in intervals
            break;
        end if
        if Solve (14)_j then
            Solve (15)_j
            ∀j_(TCP) ← Solve (21) Color Red;
            ∀j_(TCP) ← LineWidthSize Double
        else
            ∀j_(TCP) ← Solve (21) Color Blue;
            ∀j_(TCP) ← LineWidthSize Single;
        end if
        Solve (16)
        TCE[i] += Solve(17)
    end for
end for
Stop Timer
return (TCE)
```

4.2. Malicious Hosts

To simulate the described attacks in our network model, we need to arbitrarily select random hosts as malicious hosts, attempting both worm and TCP-SYN flood attacks. Randomness is challenging to characterize mathematically because of process unpredictability, but it is a critical feature of nature and crucial for many scientific applications [37]. Inaccuracies or failures in the theoretical modeling of such processes, e.g., from adversarial attacks, restrict random number generators' reliability through weaknesses that are difficult to detect and control. We use a randomness generation model verified by Bell inequality violation from [38] in Equation (1) to obtain designated attack hosts for our network model.

$$Z = \sum_{i,j} (-1)^{ij} [P(x = y|ij) - P(x \neq y|ij)] \tag{1}$$

where $P(x = y|ij)$ represents the probability of $x = y$ if sets (x, y) are selected and $P(x \neq y|ij)$ is analogously defined. Therefore, Z becomes the summation outcome that quantifies the contrast in probabilities over certain network behavior conditions, and we use i and j as summation indices when iterating over the network model. x and y denote variables within the probability functions P, where these could represent specific network states, behaviors, or outcomes being analyzed.

4.3. Attack Dataset

The underlying choice of the dataset in NIDS research is not purely about modernity, but rather the comprehensiveness, applicability, and the depth of provided features. While our selected dataset KDDCUP'99 [39] might be considered historically grounded, its continued significance in recent research endeavors is evident [40–42]. As delineated in Sections 1 and 2, our study focuses on particular malicious activities and behaviors within network settings. Many of these behaviors echo the characteristics found in the KDD-99 dataset, indicating that foundational threat vectors remain deeply embedded in contemporary attack patterns. Furthermore, the mechanisms employed by computer worms are still pivotal in elucidating the propagation dynamics of infections in current computational models [43–46].

For experiment replicability and reliability purposes, it is essential to select a well-known but vast attack dataset publicly available to other researchers. The KDDCUP'99 [39] dataset developed by the MIT Lincoln Labs is widely used to validate most proposed NIDSs [47]. This dataset of approximately 2,000,000 network records has approximately 4,900,000 single connection threat vectors, with each threat vector having 41 normal or attack labeled features. Based on our selected threat description, we choose attacks that fall into the DoS category, and probing attacks as shown in Table 1. The malicious hosts selected in Section 4.2 are simulated as inimical source nodes performing an attack. The attack dataset we use has a wide range of DoS and probing attack vectors, as detailed in Table 1, which allows our model a comprehensive array of attack scenarios, thereby enabling a rigorous validation of our detection capabilities against both volumetric (DoS) and reconnaissance (probing) activities.

Table 1. Selected KDDCUP'99 dataset [39].

Attack Label	Attack Types	Instance Count	Total Instances
DoS	Neptune	101,201	386,203
	Back	2203	
	teardrop	979	
	Apache2	737	
	Mailbomb	293	
	Smurf	280,790	
	IPsweep	1247	
Probe	Nmap	231	3514
	Mscan	996	
	Portsweep	1040	
		Grand Total	389,717

4.4. Network Model

To effectively design and evaluate our model's essential constituents, we need to evaluate the network prerequisites in terms of handling communication latency. This is crucial for Algorithm 2, where a *resolve module* is triggered to hosts with anomalous connections as part of `MO-4`. In this section, we abstractly describe our consideration of a dynamic *step model* [48] network simulation approach using the network state we derived through the process in Sections 3.2 and 4.1. We place fixed latency estimation values used to consider routing decision time as routing delay (R_d), router passing cycle-time as switch delay (S_d), link passing cycle-time as link propagation delay (L_d), and link bandwidth (L_b).

All packets sent at time interval Δt use a modeled average latency L or fixed network parameters. For different network parameters, the implementer would need to recompute L and cover expanded areas, such as routing algorithms or different policies. Applying the queue and regression models from [49], we quantify source–destination packet behavior in our model network and describe the *resolve module* latency to a selected destination host as follows:

$$L_i = f(P) + \xi(D) \tag{2}$$

where $f(P)$ represents the minimal source–destination path physical latency accumulated by a particular packet. We assume $f(P)$ is given only by network constraints, e.g., the distance between hosts, meaning that there is no resource contention among packets. $\xi(D)$ represents the dynamic contention interaction of packets given network constraints, e.g., link bandwidth.

Algorithm 2 Visualization of defense effects (S, t)

Require: All hosts S, time-intervals t
Ensure: Total Cyberattack Effect (TCE%)

```
Start Timer
for i ← 1, t do
    for j ← 1, S do                          ▷ hosts ordered by IP address
        if Timer > t_i then                  ▷ Check in intervals
            break;
        end if
        if Solve (14)_j then
            DefenseModule (Solve (2)_j)      ▷ Section 4.8
            ∀j_(TCP) ← Solve (21) Color Blue;
            ∀j_(TCP) ← LineWidthSize Single;
            Solve (16)
            TCE[i] += Solve (17)
        end if
    end for
end for
Stop Timer
return (TCE)
```

4.4.1. Physical Delays ($f(P)$)

In our calculation of the source–destination path P, we define the time taken to cross a link by the first packet header byte as burst time(B_t) and time taken for one packet to cross a switch as switching time (H_t) over a path containing W switches. Switching techniques are essential considerations, as they determine how internal switches connect the input to output and message transfer time. We adopt Virtual Cut-Through [50], which is highlighted by Duato et al. [51] as the most used switching technique. Hence, our calculations use the following:

$$f(P) = B_t + \frac{Pk_s}{Lk_b} + W\left(B_t + (H_t + \frac{Pk_h}{Lk_b}\right) \tag{3}$$

where Pk_s denotes the packet full size in bits, Pk_h denotes the packet header size in bits, and Lk_b denotes the link bandwidth in bit/sec.

4.4.2. Contention Delays ($\xi(D)$)

We assume that network synchronization occurs at fixed time intervals, as this augments simulation speed with minimal accuracy loss, as described in [52]. Consequently, packets are sent to the network asynchronously until the next synchronization interval Δt. Our model calculates $\xi(D)$ by evaluating simulated destination hosts using a uniform distribution of arrival times, which represents a queue over the time interval Δt. This can introduce additional delays if consecutive packets overlap upon arrival. For each packet at a time interval, we identify the simulated destination host and compute distributed arrivals including new packets, and then, as per the destination queue and network synchronization, we compute ξlatency. To avoid recursive dependencies, we define an initial latency, ξ_0, which represents the system latency at the start of the simulation or the latency of the first packet when the queue is empty. The $\xi(D)$ of packet y per destination n as follows:

$$\xi(D)_y = \begin{cases} \xi[y-1] + f(P)_{(y-1)} + \frac{T_i}{T_{Pk_s}}, & \text{if } \xi[y-1] > 0 \\ f(P)_{(y-1)} + \frac{T_i}{T_{Pk_s}}, & \text{if } \xi[y-1] \leq 0 \end{cases} \tag{4}$$

where ξ_0 is the latency before any packets are processed, $f(P)_{(y-1)}$ denotes the physical delay of the previous packet in the queue, i.e., $y-1$, $\xi[y-1]$ denotes the contention delay of the previous packet, T_i denotes the synchronization time interval, and T_{Pk_s} denotes total

packets processed in T_i, all with simulated destination n. The condition $\xi[y-1] \leq 0$ could occur if a packet is processed without any queuing delay, in which case its latency is only due to processing and transmission times. The initial condition ξ_0 is a predetermined latency that is assigned before the arrival of the first packet. This latency could be due to various factors such as initial processing delay, propagation delay, or any inherent delays present in the network. The condition $\xi[y-1] \leq 0$ would imply that the preceding packet encountered no queuing delay and was processed within its synchronization interval, which could occur under light network load conditions or when a packet arrives exactly at the synchronization boundary. These conditions ensure that $\xi(D)_y$ represents the total latency experienced by packet y, considering both queuing and processing times. These approaches allow us to test network behavior under different realistic inputs, providing insightful results for our proposed solution. We are also able to model the accumulation of delays due to contention in the network and test behavior under different realistic inputs, providing great results for our proposed solution.

4.5. Intrusion Detection

There are two widely used intrusion detection techniques: anomaly and signature-based detection. An anomaly-based detection approach is based on the hypothesis that any intrusive activity will be distinctive from normal activities. Meanwhile, signature-based detection assumes that each attack has an associated pattern that can be referenced for identification. Before visualizing the effects of a cyberattack, a model has to be adept in various attack detection including DoS [53], which is the limitation of signature-based detection techniques. The collection of intrusion signatures for comparison is not only *reactive* but also onerous with the current growing number of threats.

We innovate our network anomaly detection by analyzing the temporal behavior of network traffic, focusing on the dynamic nature of network interactions, allowing a real-time adaptation to what is considered normal behavior while enabling the detection of anomalies not just based on static patterns but on evolving network conditions. By extracting sufficient TCP connections and network packets from our Wireshark capture, we enhance the traditional anomaly detection model to include a novel data analytics-based technique. The novelty of our model is highlighted in the ability to dynamically interpret network behavior and giving granular analysis of network traffic patterns to improve conventional anomaly detection through open-source tools.

Therefore, as part of `MO-2`, we select the anomaly-based detection technique for our intrusion detection based on the available dataset and how it significantly influences data analytics-based techniques. We adopt this consideration from successfully improved performance reviews in [54], using temporal data to handle anomaly detection. After extracting sufficient TCP connections and network packets from our Wireshark capture, we transition our model to include a novel data analytics-based anomaly detection technique. To differentiate success rates between intruders and benign hosts based on targeted services running, we combine part a) failure rate (FR) and part b) established connections (ECs) to develop a failure rate and established connections-based network intrusion detection system named FREC-NIDS. Within the scope of our proposed model, the failure rate and established connections-based network intrusion detection system (FREC-NIDS) employs a unique blend of network data analytics to identify potential threats. By definition, FREC-NIDS analyzes network traffic through two primary lenses as described: the failure rate (FR) of connection attempts and the volume of successfully established connections (ECs). This dual analysis allows for enhanced detection of sophisticated intrusion attempts.

We consider that malicious hosts have a higher number of failed connections attempting many different host connections within a short period than non-malicious hosts. Malicious connection requests are mostly rejected because not all hosts would be running the targeted service, resulting in a low connection success rate. By contrast, non-malicious hosts attempt connection requests with near certainty for positive response, resulting in a high success rate.

4.6. Failure Rate (FR)

The FR comparison allows us to check "apples to apples" on client nodes requesting data exchange to a particular active host, instead of comparing client nodes to all active hosts. This is crucial because not all active hosts are identical in operations, although data exchange from a particular host might be similar to different client nodes. For example, in a real-life scenario, when a computer user cannot connect to a mail server, they will confirm with their colleagues if access is impossible on their computers too. Thus, the FR of all computers trying to connect to the mail server in that subnet will be similar, whereas it would be odd if other computers can connect except one computer.

Malicious hosts try maximizing their exploits by attempting sundry connections within a short period. Thus, we stratify normal and anomalous network connections considering the ratio of failed connections F_r to overall connections O_v as the number of connections made within a particular time window t_i. If we depict the network with k number of client nodes as $C = \{C_1, C_2, C_3, \ldots, C_k\}$ and m number of active hosts as $H = \{H_1, H_2, H_3, \ldots, H_m\}$, a successful TCP connection is confirmed when an ACK from an active host H_m is received for a particular C_k SYN request:

$$BenignConnection_i = BC_i = SYN\ sent\ by\ C_i\ and\ ACK\ received\ from\ H_m \tag{5}$$

Similarly, the failure of a client node requesting a TCP connection is confirmed when there is a *timeout* response from an active host H_i to a particular SYN request:

$$Failure = F_{C_i} = \begin{cases} 1 & \text{if } SYN \text{ request by } C_i \text{ times out,} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Overall connections initiated to a particular active host H_m are obtained through:

$$OverallConnections = O_{v_{C_i}} = BC_i + F_{C_i} \tag{7}$$

Now, we define the FR $C_{f_{r_1}}$ of client C_1 in the network as the ratio of the number of failures in C from each active host H_i which is

$$C_{f_{r_1}} = \frac{F_{C_1}}{O_{v_{C_1}}}, \quad C_{f_{r_2}} = \frac{F_{C_2}}{O_{v_{C_2}}}, \quad \ldots, \quad C_{f_{r_k}} = \frac{F_{C_k}}{O_{v_{C_k}}} \tag{8}$$

where $i = 1, 2, \ldots, k$. The average FR denoted as A_{f_r} of all client nodes to H_i in the network is defined as a threshold τ, which is the ratio of the sum of client FRs to all clients present in the network at a constant time interval t_i.

$$A_{f_r}(t_i) = \frac{1}{k} \sum_{j=0}^{k} C_{f_{r_j}}(t_i) \tag{9}$$

Threshold τ is dynamically determined to effectively distinguish between benign and malicious network connection through A_{f_r} of all network connections within a predefined timeframe. In our experiment description, Section 5, we will give a sample predefined timeframe. By analyzing historical network traffic data, τ is calibrated to reflect the network's typical behavior, ensuring that only significant deviations, which could indicate malicious intent, are flagged. Our adaptive thresholding approach allows FREC-NIDS to maintain high sensitivity to the attack dataset while minimizing false positives. Based on τ, we compare A_{f_r} with individual client nodes FRs. A client node C_i is classified as malicious when $C_{f_{r_i}}$ is greater than our threshold τ as follows:

$$C_{f_{r_1}} > \tau, \quad C_{f_{r_2}} > \tau, \quad \ldots, \quad C_{f_{r_k}} > \tau \tag{10}$$

However, this is insufficient, as $C_{f_{r_k}} > \tau$ cannot be examined in isolation in a networked environment. It is important to examine FRs from all clients in the network to positively identify a malicious client node. We use the comparison in Equation (10) to process respective FR events distances in an empirical distribution function. We assigned the FR of each client as p independent and identically distributed arbitrary variable set $X_i \in \{X_1 \leftarrow C_{f_{r_1}}, X_2 \leftarrow C_{f_{r_2}}, X_3 \leftarrow C_{f_{r_3}}, \ldots\ldots, X_p \leftarrow C_{f_{r_p}}\}$. The FR distribution function is as follows:

$$G_p(x) = \frac{1}{p} \sum_{i=1}^{k} I_{[-\infty,x]}(X_i) \tag{11}$$

where $I_{[-\infty,x]}$ denotes the indicator function that differentiates between malicious clients whose F_r is greater than the threshold τ and a benign client whose F_r is less than the threshold τ. Thus, the indicator function $I_{[-\infty,x]}$ is 1 when the given random variable x is greater than X_i, ranging from $[-\infty, x]$; otherwise, it is 0.

$$I_{[-\infty,x]} = \begin{cases} 1, & \text{if } x \geq X_i \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

To quantify the distance among client nodes' FRs, we apply the Kolmogorov–Smirnov (KS) two-sided test statistic [55] based on the difference between the empirical distribution function and cumulative distribution function.

$$D_k = \sup_x |G_k(x) - G(x)| \tag{13}$$

where $\sup_x$ of a subset X_S in a partially ordered set X_i is the least element in X_i, which is greater than or equal to all elements of $X_S \subseteq X_i$, and $G(x)$ represents the cumulative distribution function. Finally, FR is classified as malicious if

$$FR_C = \begin{cases} malicious, & \text{if } D_k \geq \tau \\ benign, & \text{otherwise} \end{cases} \tag{14}$$

To better understand the context of these equations, Equation (11) calculates the empirical distribution of failure rates across network client nodes to help in visualizing and understanding the distribution of connection failures, which is critical for identifying patterns that may indicate malicious activity. By assessing how these failure rates spread across the network, the model can better distinguish between normal and suspicious behaviors. Additionally, we use the indicator function Equation (12) to classify network connections based on their failure rates, comparing them against a predefined threshold. This aids in filtering out benign activities from potentially malicious ones. If a connection's failure rate exceeds the threshold, it is flagged for further inspection, ensuring that the model focuses on connections that more likely represent genuine threats. In KS test statistic, Equation (13) measures the maximum deviation between the empirical distribution of observed network behaviors and the expected distribution under normal conditions to assess the significance of observed anomalies and confirm whether observed patterns of network behavior deviate from what is typically expected in a way that suggests malicious activity. Finally, Equation (14) applies the KS test results to make a decision on whether a network behavior is considered normal or anomalous by providing a concrete criterion that flags nodes as potentially malicious based on statistical evidence. This enables our mode to take action on connections that most strongly deviate from normal behavior patterns.

4.7. Established Connections (ECs)

Because TCP defines ECs to be maintained until data exchange or transfer between a client and a host is complete, it is essential to obtain all existing connections in the network following an FR alert for a malicious host. To ensure that the malware is not replicated to other hosts, an aggregate of hosts in the network connected to the malicious host is

obtained. As mentioned before, we induce random generated network malicious hosts into the network running the KDDCUP'99 attack data. Once we receive an alert from the FR part of FREC-NIDS, we need to check all hosts connected to the malicious client and flag them as unavailable. We assume that having infiltrated a network host, connections to the malicious host would be under the attacker's control and malware is replicated. To present a lucid picture of actualized threats in the realization of `MO-2` and `MO-3`, our visual display system will output all affected connections as described in Section 4.9.

We model the FREC-NIDS parameters in our simulated environment to visualize cyberattack effects through Algorithm 1 and Equation (21), as will be explained in Section 4.9. The relationship between our visualization parameters defines $L = C \cup H$ as the set for the total number of network hosts, including all client nodes and active hosts, and as per Equation (5), then, the total number of hosts is given by $S = m + k$ at a set of time intervals $t = \{t_1, t_2, \ldots, t_n\}$. For each t_n, let $G(t_n)$ be an undirected graph with vertex set $V = \{v_1, v_2, \ldots, v_m\}$ as connections to other hosts, with edge set $E(t_n)$. The degree $\delta_{(v)}$ of a vertex v is the number of edges incident to it. Malicious connections M_c in the EC part of FREC-NIDS are given by

$$\gamma_{M_{c_{(t_n)}}} = \sum_{i=0}^{S} \delta_{(v_i)}, \qquad \text{if } FR_c \text{ is malicious} \tag{15}$$

where S denotes the number of nodes at time interval t_n; with this, we calculate the network capacity as:

$$\beta_{Y_{(t_n)}} = \sum_{i=1}^{n} \left(S^i \cdot O^i_{v_{C_k}} \right) \tag{16}$$

where $O_{v_{C_k}}$ denotes the total number of connections at time interval t_n. Then, the network capacity helps obtain average network unavailability, the calculated comparison of all established malicious connections, to the network capacity at t_n

This gives the security analyst all cyber-assets affected as the TCE%.

$$TCE_{t_n} = \frac{\gamma_{M_c}}{\beta_{Y_{(t_n)}}} \tag{17}$$

4.8. Malicious Connection Defense

A simple mitigation approach for dealing with malicious connections would include the termination of malicious connections, but this can be insolent, particularly for false positives. Additionally, the termination only gives an attacker feedback to try other options. One of the best alternatives we found and adopted to achieve high resilience to compromised hosts without relying on static or localized network information is [56], which proposes a polynomial-based compromise-resilient en-route filtering scheme against false data. Therefore, to achieve `MO-4`, i.e., mitigation of attacks in the *resolve-module*, we place a compromised hosts report $R_{FR_{C_i}}$ from Equation (14) on the main network controller verifiable by intermediate hosts through: *Condition 1*: Message authentication polynomial (MAP) on the compromised host FR_{C_i}, *Condition 2*: Timestamp T on network controller. $R_{FR_{C_i}}$ is obtained by:

$$R_{FR_{C_i}} = auth_{C_i} | T \tag{18}$$

where $auth_{C_i}$ is the MAP stored in host i. Using the proposed approach of handling malicious connections in [57] that takes over malicious connections as a transparent proxy by migrating TCP states to a user-level TCP stack, we design our malicious connection resolve module to attract packets from compromised hosts by fallaciously claiming fresh shortest packet routes but not forwarding them onward. This is achieved through a neighboring host having an EC to the compromised host, as described in Section 4.7,

always responding to a malicious host FR_{C_i} route request (RREQ), with the best alternative route reply (RREP).

Despite the process just described, to stop routing packets through a compromised connection, any RREP of a fresh route from any malicious host is dropped. We assume that this process continues until the security analyst takes the compromised host off the network for subsequent malware check. To prevent resource lock-up in case of a delayed malware check response from a security analyst, we purge malicious connections through timeout guidelines proposed in [58]. We use a timeout configuration that accommodates RFC 2988 [59] dynamics as $1.0 \leq \tau \leq 11.0$, where τ is the timeout value in seconds. The value for τ is adaptively set based on different settings checked at time t, with the total number of entries E_t given by:

$$E_t = N_t + \lambda\tau \tag{19}$$

where N_t denotes the benign connection and λ denotes the malicious connection rate. Effectively, FR_{C_i} cannot process the threats described in Section 3.3 as the packets are absorbed and dropped without forwarding. As the attack is mitigated, Algorithm 2 explains this module together with our stated parameters to show how available resources are visually recognized at given time intervals. We propose this defense strategy to handle all malicious connections and then, in our display system, transform the connection color from red to blue, and size from double to single, once malicious connection activities are nulled with dropped packets.

The implemented code snippet after our FREC alerts is:

```
FREC alerts ← Solve(14)
GenerateReport(R) ← Solve(18)
//check for a hostcompromised host
while FR_{C_i} == malicious {
        VerifyReport(R)
        if SRH_{C_i} == RREQ() {
            //get existing host connection
            //send false fresh route
            q ← sendFalseRREP(in(Solve(15)_{C_i}),True)
            drop(q.packets)
        }
        if SRH_{C_i} == RREP() {
            drop(SRH_{C_i}.packets)
        }
}//SRH is Segment Request Header
```

4.9. Visual Display

Our visual display module involves real-time visualization of network activities reported from FREC with two state declaration flags, normal and malicious connections, to disambiguate our structured representations and facilitate easy user comprehension. As much as NIDS gives intrusion alerts, it is crucial to visualize the network system state holistically rather than via single intrusion alerts. The best technique to determine the true severity of a cyberattack on technical assets is to critically monitor all malicious hosts in the network compared with all benign hosts. To achieve simplicity in deducing the severity of a cyberattack and cyber-defense effects through our visualization module, we consider a cyberattack parameter β and a cyber-defense parameter δ.

To express the status of host i in multiple dimensions within a single function as the cyberattack and cyber-defense are in progress, we consider the vector function of $\vec{r}\,'(t) = \langle\beta'(t),\rangle\delta'(t)$, where $\beta'(t)$ and $\delta'(t)$ denote the coordinate functions of time t. Notably, $\vec{r}\,'(t)$ has a limit given by the vector L when t approaches t_0, denoted as $\lim_{t\to t_0}\vec{r}\,'(t) = L$ iff for every $\epsilon > 0$ there exists $\delta > 0$ such that $0 < |t - t_0| < \delta \Rightarrow$

$|r'(t) - L| < \epsilon$. We use $\dot{S}(t)$ for the time derivative of $\vec{r}\,'(t)$ with continuous-time $S(t)$, and because we expect abrupt changes in values, we consider the $\vec{r}\,'(t)$ of discrete-time S^m with the superscript indicating time step of S. We apply two discrete-time models to obtain the granularity of attack and defense effects reflected in the visual representation. We derive the first discrete-time model from a continuous-time model:

$$\dot{S}_t = \beta_i(1 - S_i) \sum_{j=1}^{n} E_{ij} S_j - \delta_i x_i \tag{20}$$

where x_i denotes the attack level in host i, β_i denotes the attack rate, δ_i denotes the defense rate, and E_{ij} denotes the edge weights between hosts. Afterward, we apply Euler's method [60] to Equation (20) to derive the second discrete-time model with time index m and a random sample $Z > 0$ as:

$$S_i^{m+1} = S_i^m + Z\left(\beta_i(1 - S_i^m) \sum_{j=1}^{n} E_{ij} S_j^m - \delta_i S_i^m\right) \tag{21}$$

To achieve `MO-3` and `MO-4`, we use the granularity obtained from Equation (21) to display blue lines with single-width sizes for normal connections and red with double-width sizes for anomalous connections. The granularity depth represents the severity of the attack, i.e., the ratio of compromised host connections to total network connections, or the effectiveness of defense measures, i.e., the ratio of benign host connections to total network connections. This choice of colors leverages common associations of red with danger and blue with safety, thereby utilizing established psychological principles to enhance user comprehension and response efficiency. The width is an additional layered property to help users who might be color blind, primarily when visualizing an extensive network with many connections.

From the visual system, we assume that initially, when network service discovery is made, everything is normal, and host connections are displayed in blue. Later, when an intrusion occurs, hosts connected to a malicious host are indicated with red color as described in Algorithm 1. Because real attacks might not occur simultaneously in a network, we introduce all induced malicious hosts simultaneously for a finite experiment setup. The granularity of the network connection colors, i.e., blueish or reddish, changes *on the fly* as attacks are detected and as our defense module neutralizes the malicious connections.

5. Results and Performance Evaluation

To evaluate our prototype and mitigation strategy, we implemented our experiment inside our Data Science and Cybersecurity Center (DSC^2) lab, Howard University, Washington, DC 20059, USA, on a MS Windows server 2016 Intel(R) Xeon CPU E5-2676 v3 @2.40GHz with 4GB RAM. We extracted a network topology from our (DSC^2) computer lab as per *service discovery* described in Section 4.4 to obtain 101 computing devices in two subnets, as summarized in Figure 2. The network module in our visual display system contains information about network devices and topology from the computer running the application.

We designate 5% attack hosts launched to the network simultaneously to perform the probing and DoS attack described in Section 4.3. Each attack host simulates the attack data in a 3-low rate (0.1, 1, 10 packets/s) and 2-high rate (100, 1000 packets/s) instance, up to a period where the experiment output is stable and does not vary much, i.e., 5 min. This time interval allowed collection of sufficient data to accurately identify anomalies without overwhelming our analysis process, but any subsequent research can vary the interval appropriately. Each attack packet IP header is distinctively labeled by setting a reserved bit. Our experiment closely mimicked an actual network environment having the performance evaluation executed on a real-time environment with a finite target. We assume the network obtains no new connections once the experiment starts, but we surmise

that the scenario could be different in the real world. The prevalent strategy for identifying anomaly-based cyberattacks is using outlier detection.

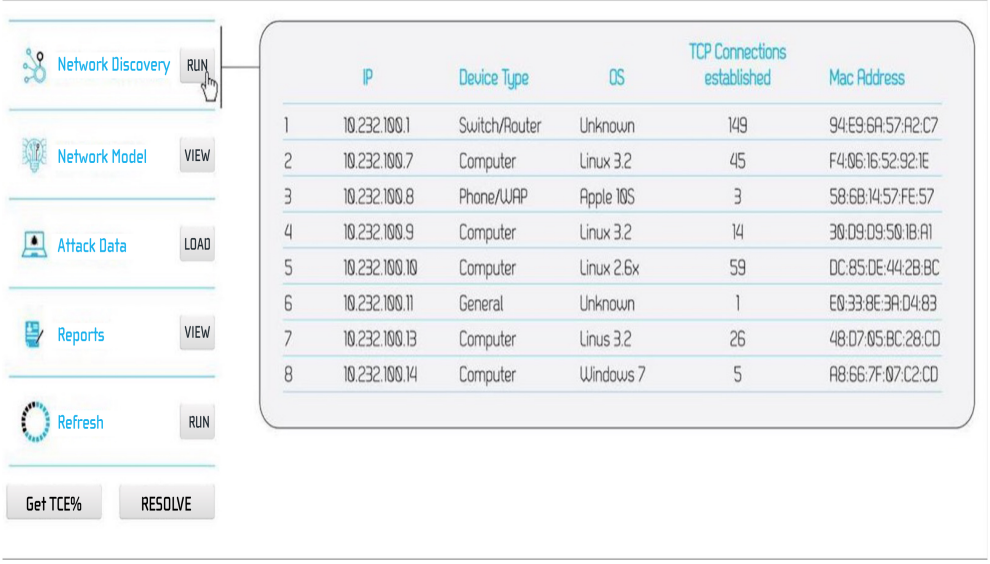

Figure 2. Extracted network model through service discovery

To demonstrate a clear identification of a malicious host having undefined arbitrary connection attempts as with the case of worm attacks, Figure 3 shows a malicious sample host C_1 detected as an outlier to seven other active hosts. The malicious host C_1 tries maximizing the exploit by attempting sundry connections within a short period. However, the connection requests sent from C_1 are mostly rejected, as not all hosts would be running the targeted service, resulting in a low connection success rate; meanwhile, the other non-malicious hosts attempt connection requests with near certainty for a positive response, resulting in a high success rate. Client C_2 and C_3 have slightly higher failure rate than Client C_4 and C_5 because of normal network resource contention, but they are not flagged as outliers.

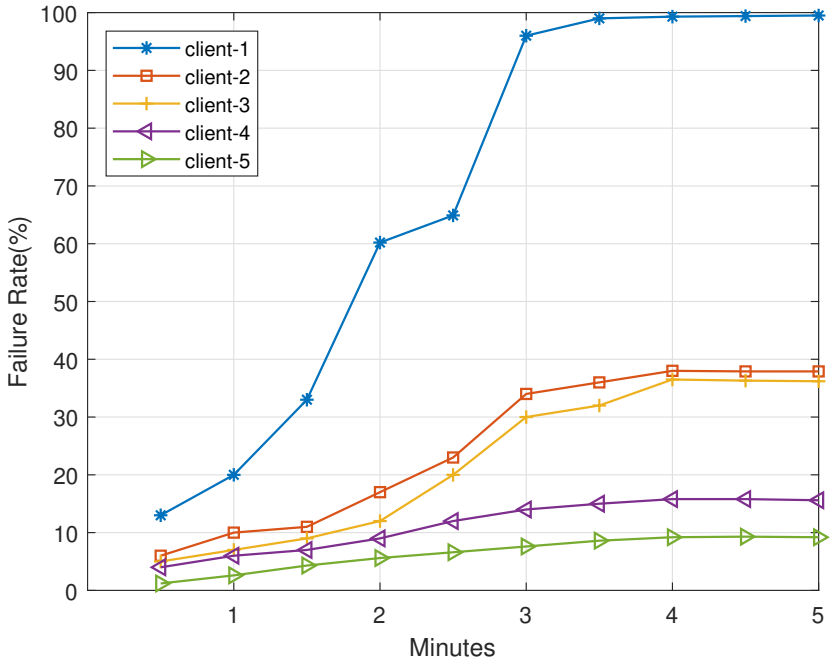

Figure 3. Failure rate for malicious client C_1 and benign clients C_2 to C_5.

Therefore, with a finite target, precision, recall, and F_1 score become crucial to our analysis based on the attack dataset. Accuracy is our first step to understanding what we are obtaining correctly or incorrectly numerically. We extract true and false positives at different time intervals when FREC-NIDS is running and calculate precision and recall as:

$$Precision = \frac{TP}{TP + FP} \quad , \quad Recall = \frac{TP}{TP + FN} \tag{22}$$

where TP, FP, and FN denote true positive, false positive, and false negative, respectively.

Subsequently, precision and recall in Equation (22) help us evaluate if our model identifies most attack data without pulling many false positives, i.e., only what is relevant to the attacks. As shown in Figure 4, our false positive rate (FPR) at the beginning of the experiment is 5.1% but improves to 0.3% at the end of the experiment. TP at the beginning of the experiment was 89.8% and ended at a good ratio of 99.7%. The FPR experimental results summarized in Table 2 show a higher overall precision and recall for FREC-NIDS than the models from [15–19] described in Section 2. We apply the same attack dataset and experiment setup to all algorithms in the selected related works. For any related model that did not specify the percentage of attack dataset used in their implementation, we applied the full attack dataset, and we applied the same to our FREC-NIDS, while following revisions on the dataset given by [61]. By invoking 100% of the dataset, we demonstrated the replicability of implementing FREC-NIDS to real-world attack datasets without selecting which parts of the datasets can apply. Depending on the cost of FN, recall is essential as a consideration of attacks incorrectly predicted as benign. However, the weighted average (or harmonic mean) of precision and recall, calculated as the F_1 score, provides a better overall performance balance. We obtain F_1 score as:

$$F_1 Score = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \tag{23}$$

Table 2. Performance comparison to related works using the KDDCUP'99 dataset.

Model	Packets%		Performance	
	DoS	Probe	Precision	Recall
AdaBoost-based [15]	100	100	0.922	0.956
Ruan et al. [16]	N/A *	N/A *	0.895	0.894
Hierarchical SOM [17]	96.90	N/A *	0.857	0.954
Multi-Level Hybrid SVM and ELM [18]	98.13	99.54	0.981	0.949
Velliangiri [19]	N/A *	N/A *	0.990	0.921
FREC-NIDS	100	100	0.997	0.963

* Not specifically mentioned in that research.

Figure 5 summarizes the F_1 score results, where FREC-NIDS starts with an F_1 of 96.8% and ends with a high value of 97.9%. Multi-level hybrid SVM [18] finished closest to FREC-NIDS with 96.4% but having started with a low value of 88.6%. Actually, most models compared in our experiment started with a low performance score, possibly because of the required extensive training. Our network model, shown in Section 4.4, is a complex network model to subject all related models, as well as FREC-NIDS, to a robust experiment.

Ruan et al. [16] start the experiment with the worst performance, followed by Hierarchical SOM [17] and Velliangiri [19], having performance scores between 89% and 95%. The difference in performance between Bagged C5 and the other related models [15–19] is possibly due to the different classifiers from multiple decision trees. Notably, although we used the full KDDCUP'99 dataset, FREC-NIDS achieved better results.

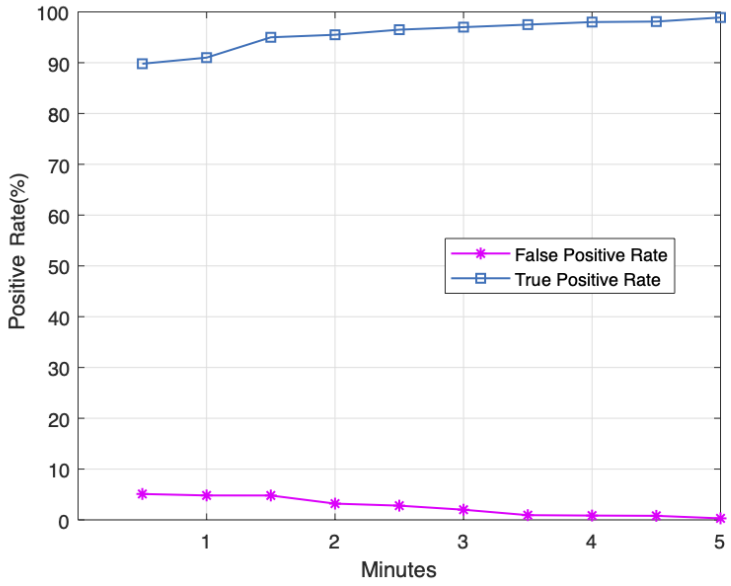

Figure 4. True and false positive rates comparison.

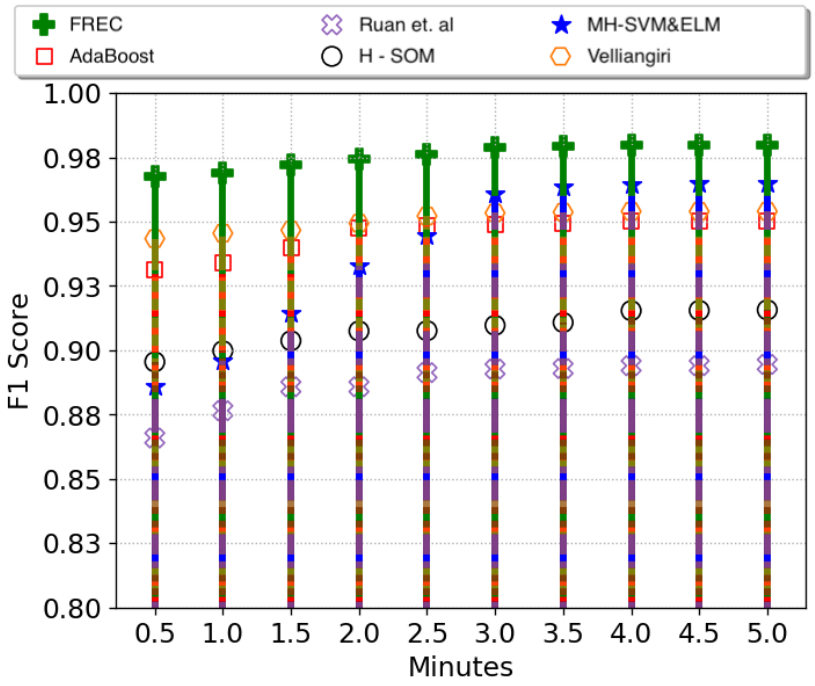

Figure 5. F_1 score compared to other models [15–19].

The core objective of our research was articulated through MO-1 in Section 1, aiming at presenting a low-cost and straightforward technique for dynamic network data extraction then significantly enhancing network anomaly detection. Our findings in Section 5, demonstrating an F_1 score of 97.9% and a minimal false positive rate of 0.3%, not only corroborate the achievement of this objective but also represent a critical step in network security. These outcomes directly address the prevailing challenges highlighted by Ferrag et al. [1] and Chowdhury & Gkioulos [2], where traditional NIDS often lag in adapting to the fast-evolving cyber threats. Our approach for MO-1 further elaborated in Section 4 utilizes open-source tools for network data extraction, coupled with a data analytics-based intrusion detection system, which enables us to achieve MO-2 by ensuring replicabil-

ity and operational efficiency without necessitating extensive training data or bespoke closed-source solutions.

6. Prototype

This paper includes an exploratory proof-of-concept demonstration developed as a web service that queries required data from a MySQL database and sends them to a visual display. The front-end is developed through a combination of JavaScript and C# published as a WebGL project from the new Unity software Entity Component System (ECS) [62]. Figure 6 shows our high-fidelity prototype with self-explanatory iconography and simple categorization of system functions. System navigation is placed on the left side with the following:

- *Network Discovery*: This executes an automated *bash script* described in Section 4.1. In case of network discovery performed outside our prototype, the data can be loaded as a CSV file through the *select file* icon on the bottom left side of the page.
- *Network Model*: This displays the scanned network topology.
- *Attack Data*: This allows loading the attack data described in Section 4.3. These data can be 100% completely loaded as used in our experiment but can also be modified for different studies.
- *Reports*: This summarizes exportable data from the experiment.
- *Refresh*: This allows a new experiment to be conducted.
- *GET TCE%*: This calculates the TCE and displays it on the top right side of the page.
- *Resolve*: This runs the defense mitigation strategy described in Section 4.8.

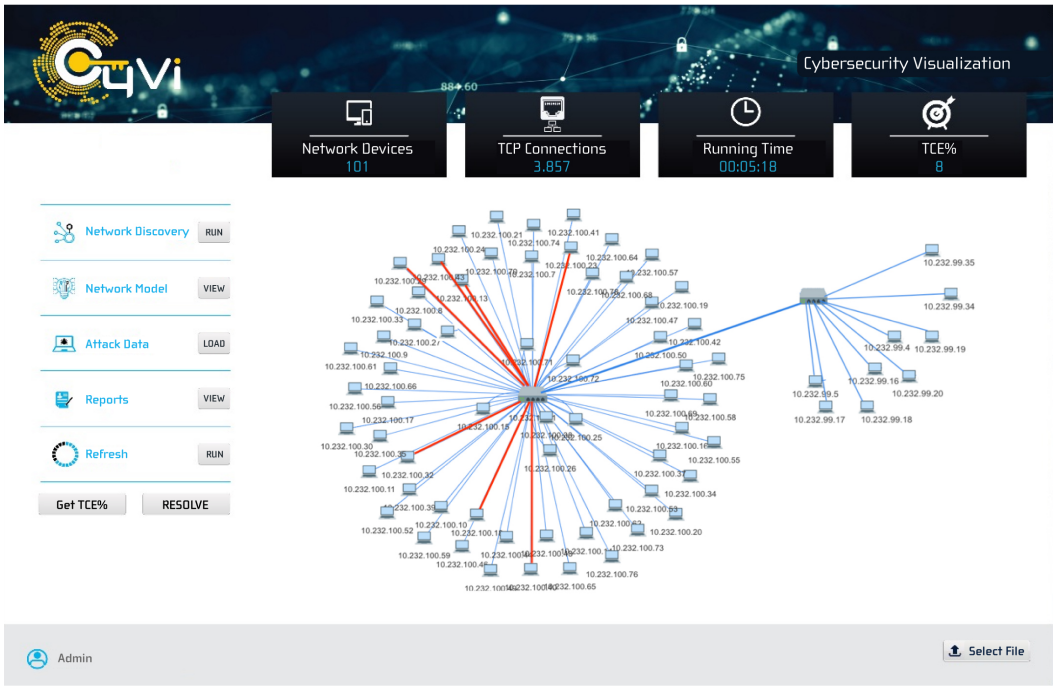

Figure 6. Snapshot of the proposed prototype.

The top dashboard summarizes our experiment including 3857 TCP connections, a total experiment time of 5 min, and an initial TCE of 8%. However, in a real environment, immediate defense and mitigation should occur on detection, and our visual system is set only to deploy the *resolve module* on Section 4.8 on execution from the user so that cyberattack effects can be viewed first.

These technical details and functional aspects of our proof-of-concept prototype build on our prototype's topology, as summarized in Figure 6, that depicts a real-time interactive network model having an array of network devices and connections. This high-level overview demonstrates the prototype's structural and operational accuracy, reflecting a

complex network. As explained, the technical backbone combines a MySQL database queried by a web service, with a responsive front-end JavaScript and Unity's ECS interface, emphasizing real-time data processing. For a practical demonstration of the system's versatility, the prototype incorporates various simulated attack scenarios. The attack data function facilitates the loading of specific attack types from the KDDCup'99 dataset, such as DoS and probing attacks, to validate the detection and visualization mechanisms.

The prototype's layout, with dedicated controls for network discovery, data loading, and mitigation strategy implementation ("Resolve"), provides real-time observations of threats, thus verifying the prototype's effectiveness and practical contribution within the field of cybersecurity visualization. The introduction of a real-time visualization layer achieves our MO-3 by presenting a novel mechanism for network security monitoring that enables informed decision-making. This feature directly addresses the gap in effective threat visualization noted in existing NIDS solutions and marks a significant step in layering effective threats identification and mitigation. The utilization of color-coded connection statuses simplifies the complexity associated with network monitoring amid massive network events. The prototype's design also achieves our MO-4 through changing network link colors from red to blue, indicating the mitigation of threats in real time. This feature directly reflects our objective to not only detect but also visually signify the neutralization of network threats for enhanced clarity and immediacy of security responses.

7. Conclusions

In light of the challenges posed by intricate NIDS schemes, there has been a surge in network breaches. Multilayered strategies that embed cyber visualization of NIDS alerts have emerged as a countermeasure, albeit accompanied by inherent constraints. Recognizing that most cyberattacks stem from unauthorized network connections, we introduced a novel NIDS, which gleans network information through processes easily replicated from network scanning utilities. Our approach monitors TCP connections, handshake FR, and the count of ECs at specific intervals, generating alerts to identify benign or malicious network connections. In our strategy, a visual-interactive prototype is designed to highlight suspicious connections, with color-coded outcomes based on cyberattack and cyber-defense rates. This enhancement provides cyber-analysts with a more informed decision-making tool, especially in situations where rapid termination of malicious connections prompts attackers to constantly switch targets. Central to our defense approach is the introduction of a verified report on compromised hosts, presented on the primary network controller. This is complemented by the deceptive broadcast of incorrect packet destination routes to compromised entities.

Once a false route is accepted by the malicious hosts and packets forwarded, these packets are subsequently dropped. Our visual tool provides the status of networked hosts' availability at given intervals rather than mere visualization of cyberattack effects on the physical world, i.e., non-technical space. Our experiment shows a significant improvement in F_1 score compared with other models [15–19], which have a lowest performance score of 86.5%. Our model showed only a 0.3% FPR and a high F_1 score of 97.9%. However, we are limited in differentiating high-level network components, such as hubs and network switches, that might not operate the same as data point networked hosts. Nonetheless, main network controllers have in the past been poorly configured and breached as high-priced attack targets, necessitating special attention. Our future work will cluster network components differently, including network computing hosts handled diversely like main network controllers.

Testing our model in expansive networks will yield more detailed data on network events, potentially enhancing our false positive rate metrics. A controlled network environment allowed us to closely monitor the model's performance and fine-tune its detection mechanisms but does not offer the ability to capture important computation complexity metrics. Recognizing the importance of this aspect for real-world applications, our future works will include processing time and memory usage processing as the model analyzes

live network traffic over time. This will set up the stage for future investigations into its computational complexity. This next step is crucial for ensuring FREC-NIDS meets the dynamic demands of modern network security challenges. Our future plans also involve expanding our model to include behavioral analysis for anomaly detection that will enhance the model's ability to analyze patterns of behavior within the network traffic and adapt to new patterns of network behavior over time. Furthermore, as an expansion of our mitigation strategy, future iterations of our model will consider integration with other security systems such as open-source firewalls and intrusion prevention systems. This integrated approach will advance our model's fingerprint of malicious connections in a holistic defense approach.

Author Contributions: Conceptualization, E.M. and D.R.; methodology, E.M. and D.R. formal analysis, E.M. and D.R.; validation, E.M. and D.R.; Evaluation, E.M. and D.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partly supported by the U.S. Air Force Research Lab through Radiance Technologies, the U.S. NSF under grants CNS 1650831, and HRD 1828811, and by the U.S. Department of Homeland Security under grant DHS 2017-ST-062-000003. However, any opinions, findings, and conclusions or recommendations expressed in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the funding agencies.

Data Availability Statement: The data and prototype presented in this study are available on request from the corresponding author.

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

References

1. Ferrag, M.A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *J. Inf. Secur. Appl.* **2020**, *50*, 102419. [CrossRef]
2. Chowdhury, N.; Gkioulos, V. Cyber security training for critical infrastructure protection: A literature review. *Comput. Sci. Rev.* **2021**, *40*, 100361. [CrossRef]
3. Chapaneri, R.; Shah, S. A comprehensive survey of machine learning-based network intrusion detection. In *Smart Intelligent Computing and Applications, Proceedings of the Second International Conference on SCI 2018, Bhubaneswar, India, 21–22 December 2018*; Springer: Singapore, 2018; pp. 345–356.
4. Silva, A.R.; McClain, J.T.; Anderson, B.R.; Nauer, K.S.; Abbott, R.; Forsythe, J.C. *Factors Impacting Performance in Competitive Cyber Exercises*; Technical Report; Sandia National Lab. (SNL-NM): Albuquerque, NM, USA, 2014.
5. Kashyap, R.; Piersson, A.D. Impact of big data on security. In *Handbook of Research on Network Forensics and Analysis Techniques*; IGI Global: Hershey, PA, USA, 2018; pp. 283–299.
6. Zhao, H.; Tang, W.; Zou, X.; Wang, Y.; Zu, Y. Analysis of Visualization Systems for Cyber Security. In *Recent Developments in Intelligent Computing, Communication and Devices*; Springer:Berlin/Heidelberg, Germany, 2019; pp. 1051–1061.
7. Shiravi, H.; Shiravi, A.; Ghorbani, A.A. A survey of visualization systems for network security. *IEEE Trans. Vis. Comput. Graph.* **2011**, *18*, 1313–1329. [CrossRef] [PubMed]
8. Damaševičius, R.; Toldinas, J.; Venčkauskas, A.; Grigaliūnas, Š.; Morkevičius, N.; Jukavičius, V. Visual Analytics for Cyber Security Domain: State-of-the-Art and Challenges. In Proceedings of the International Conference on Information and Software Technologies, Vilnius, Lithuania, 10–12 October 2019; pp. 256–270. Author: We couldn't confirm the location
9. Ware, C. *Information Visualization: Perception for Design*; Morgan Kaufmann: Amsterdam, The Netherlands, 2012.
10. MITRE. Threat-Based Defense. 2024. Available online: https://attack.mitre.org (accessed on 16 March 2024).
11. Kim, S.; Park, K.J.; Lu, C. A survey on network security for cyber–physical systems: From threats to resilient design. *IEEE Commun. Surv. Tutor.* **2022**, *24*, 1534–1573. [CrossRef]
12. Neupane, S.; Ables, J.; Anderson, W.; Mittal, S.; Rahimi, S.; Banicescu, I.; Seale, M. Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. *IEEE Access* **2022**, *10*, 112392–112415. [CrossRef]
13. Kapustin, V.; Paulauskas, N. Analysis of TCP flood attack using NetFlow. *Moksl.-Liet.-Ateitis/Sci.-Future Lith.* **2023**, *15*. [CrossRef] Author: Hello, looks like it is not available.
14. Moustafa, N.; Hu, J.; Slay, J. A holistic review of network anomaly detection systems: A comprehensive survey. *J. Netw. Comput. Appl.* **2019**, *128*, 33–55. [CrossRef]
15. Guo, W.; Luo, Z.; Chen, H.; Hang, F.; Zhang, J.; Al Bayatti, H. AdaBoost Algorithm in Trustworthy Network for Anomaly Intrusion Detection. *Appl. Math. Nonlinear Sci.* **2022**, *8*, 1819–1830. [CrossRef]

16. Ruan, Z.; Miao, Y.; Pan, L.; Patterson, N.; Zhang, J. Visualization of big data security: A case study on the KDD99 cup data set. *Digit. Commun. Netw.* **2017**, *3*, 250–259. [CrossRef]
17. Kayacik, H.G.; Zincir-Heywood, A.N.; Heywood, M.I. A hierarchical SOM-based intrusion detection system. *Eng. Appl. Artif. Intell.* **2007**, *20*, 439–451. [CrossRef]
18. Al-Yaseen, W.L.; Othman, Z.A.; Nazri, M.Z.A. Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Syst. Appl.* **2017**, *67*, 296–303. [CrossRef]
19. Velliangiri, S. A hybrid BGWO with KPCA for intrusion detection. *J. Exp. Theor. Artif. Intell.* **2020**, *32*, 165–180. [CrossRef]
20. Feng, Y.; Li, J.; Nguyen, T. Application-layer DDoS defense with reinforcement learning. In Proceedings of the 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS), Hangzhou, China, 15–17 June 2020.
21. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 1153–1176. [CrossRef]
22. Bhardwaj, A.K.; Singh, M. Data mining-based integrated network traffic visualization framework for threat detection. *Neural Comput. Appl.* **2015**, *26*, 117–130. [CrossRef]
23. Ohnof, K.; Koikef, H.; Koizumi, K. IPMatrix: An effective visualization framework for cyber threat monitoring. In Proceedings of the Ninth International Conference on Information Visualisation (IV'05), London, UK, 6–8 July 2005; pp. 678–685.
24. Ulmer, A.; Schufrin, M.; Sessler, D.; Kohlhammer, J. Visual-Interactive Identification of Anomalous IP-Block Behavior Using Geo-IP Data. In Proceedings of the 2018 IEEE Symposium on Visualization for Cyber Security (VizSec), Berlin, Germany, 22 October 2018; pp. 1–8.
25. Small, S.G.; Medsker, L. Review of information extraction technologies and applications. *Neural Comput. Appl.* **2014**, *25*, 533–548. [CrossRef]
26. Ren, B.; Song, Y.; Zhang, Y.; Liu, H.; Chen, J.; Shen, L. Reconstruction of Complex Networks Under Missing and Spurious Noise Without Prior Knowledge. *IEEE Access* **2019**, *7*, 45417–45426. [CrossRef]
27. Zhang, Z.; Zhao, Y.; Liu, J.; Wang, S.; Tao, R.; Xin, R.; Zhang, J. A general deep learning framework for network reconstruction and dynamics learning. *Appl. Netw. Sci.* **2019**, *4*, 4950. [CrossRef]
28. Lyon, G.F. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*; Association for Computing Machinery (ACM): New York, NY, USA, 2009.
29. Kim, M.; Leskovec, J. The network completion problem: Inferring missing nodes and edges in networks. In Proceedings of the 2011 SIAM International Conference on Data Mining, SIAM, Mesa, AZ, USA, 28–30 April 2011; pp. 47–58.
30. Chen, J.; Teh, J.; Liu, Z.; Su, C.; Samsudin, A.; Xiang, Y. Towards Accurate Statistical Analysis of Security Margins: New Searching Strategies for Differential Attacks. *IEEE Trans. Comput.* **2017**, *66*, 1763–1777. [CrossRef]
31. Koganti, V.S.; Galla, L.K.; Nuthalapati, N. Internet worms and its detection. In Proceedings of the 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil, India, 16–17 December 2016; pp. 64–73.
32. Bo, C.; Fang, B.X.; Yun, X.C. Adaptive method for monitoring network and early detection of internet worms. In Proceedings of the International Conference on Intelligence and Security Informatics, San Diego, CA, USA, 23–24 May 2006; pp. 178–189.
33. Middleton, A. Stuxnet: The World's First Cyber... Boomerang? *Interstate-J. Int. Aff.* **2016**, *2015/2016*, 1/1.
34. Moore, D.; Paxson, V.; Savage, S.; Shannon, C.; Staniford, S.; Weaver, N. Inside the slammer worm. *IEEE Secur. Priv.* **2003**, *1*, 33–39. [CrossRef]
35. Foundation, W. Wireshark. 1998. Available online: https://www.wireshark.org (accessed on 18 April 2023).
36. Molina, M.; Castelli, P.; Foddis, G. Web traffic modeling exploiting TCP connections' temporal clustering through HTML-REDUCE. *IEEE Netw.* **2000**, *14*, 46–55. [CrossRef]
37. Knuth, D.E. *Seminumerical Algorithms, Vol. 2: The Art of the Computer Programming*; Addison-Wesley: Boston, MA, USA, 1981.
38. Pironio, S.; Acín, A.; Massar, S.; de La Giroday, A.B.; Matsukevich, D.N.; Maunz, P.; Olmschenk, S.; Hayes, D.; Luo, L.; Manning, T.A.; et al. Random numbers certified by Bell's theorem. *Nature* **2010**, *464*, 1021. [CrossRef]
39. KDD Cup 1999: Computer Network Intrusion Detection. Available online: https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data (accessed on 18 March 2023).
40. Priyalakshmi, V.; Devi, R. Analysis and Implementation of Normalisation Techniques on KDD'99 Data Set for IDS and IPS. In Proceedings of the International Conference on Data Science and Applications: ICDSA 2022, Kolkata, India, 26–27 March 2022; pp. 51–70.
41. Prajapati, P.K.; Singh, I.; Subhashini, N. Network Intrusion Detection Using Machine Learning. In *Futuristic Communication and Network Technologies: Select Proceedings of VICFCNT 2021*; Springer: Singapore, 2023; Volume 1, pp. 55–66.
42. Keserwani, P.K.; Govil, M.C.; Pilli, E.S. An effective NIDS framework based on a comprehensive survey of feature optimization and classification techniques. *Neural Comput. Appl.* **2023**, *35*, 4993–5013. [CrossRef]
43. Shi, L.; Li, X.; Gao, Z.; Duan, P.; Liu, N.; Chen, H. Worm computing: A blockchain-based resource sharing and cybersecurity framework. *J. Netw. Comput. Appl.* **2021**, *185*, 103081. [CrossRef]
44. Achar, S.J.; Baishya, C.; Kaabar, M.K. Dynamics of the worm transmission in wireless sensor network in the framework of fractional derivatives. *Math. Methods Appl. Sci.* **2022**, *45*, 4278–4294. [CrossRef]
45. Sánchez-Patiño, N.; Gallegos-Garcia, G.; Rivero-Angeles, M.E. Teletraffic Analysis of DoS and Malware Cyber Attacks on P2P Networks under Exponential Assumptions. *Appl. Sci.* **2023**, *13*, 4625. [CrossRef]

46. Li, Z.; Rios, A.L.G.; Trajković, L. Detecting internet worms, ransomware, and blackouts using recurrent neural networks. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 2165–2172.
47. Revathi, S.; Malathi, A. A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. *Int. J. Eng. Res. Technol. (IJERT)* **2013**, *2*, 1848–1853.
48. Szymaniak, M.; Presotto, D.; Pierre, G.; van Steen, M. Practical large-scale latency estimation. *Comput. Networks* **2008**, *52*, 1343–1364. [CrossRef]
49. Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*; John Wiley & Sons: Hoboken, NJ, USA, 1990.
50. Duato, J.; Yalamanchili, S.; Ni, L. *Interconnection Networks: An Engineering Approach*; Morgan Kaufmann Pub. Inc.: Albuquerque, NM, USA, 2002.
51. Duato, J.; Robles, A.; Silla, F.; Beivide, R. A comparison of router architectures for virtual cut-through and wormhole switching in a NOW environment. *J. Parallel Distrib. Comput.* **2001**, *61*, 224–253. [CrossRef]
52. Falcon, A.; Faraboschi, P.; Ortega, D. An adaptive synchronization technique for parallel simulation of networked clusters. In Proceedings of the ISPASS 2008-IEEE International Symposium on Performance Analysis of Systems and software, Austin, TX, USA, 20–22 April 2008; pp. 22–31.
53. Lahti, C.B.; Peterson, R. *Sarbanes-Oxley Compliance Using COBIT and Open Source Tools*; Syngress: Amsterdam, The Netherlands, 2005.
54. Ruff, L.; Kauffmann, J.R.; Vandermeulen, R.A.; Montavon, G.; Samek, W.; Kloft, M.; Dietterich, T.G.; Müller, K.R. A unifying review of deep and shallow anomaly detection. *Proc. IEEE* **2021**, *109*, 756–795. [CrossRef]
55. Simard, R.; L'Ecuyer, P. Computing the two-sided Kolmogorov-Smirnov distribution. *J. Stat. Softw.* **2011**, *39*, 1–18. [CrossRef]
56. Yang, X.; Lin, J.; Yu, W.; Moulema, P.M.; Fu, X.; Zhao, W. A novel en-route filtering scheme against false data injection attacks in cyber-physical networked systems. *IEEE Trans. Comput.* **2013**, *64*, 4–18. [CrossRef]
57. Tang, Q.; Zheng, C.; Lu, Q.; Yang, W.; Yuan, Q.; Chen, X. Taking over malicious connection in half way by migrating protocol state to a user-level TCP stack. In Proceedings of the 2017 8th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 4–6 April 2017; pp. 228–233.
58. Kim, H.; Kim, J.H.; Kang, I.; Bahk, S. Preventing session table explosion in packet inspection computers. *IEEE Trans. Comput.* **2005**, *54*, 238–240.
59. Paxson, V.; Allman, M.; Chu, J.; Sargent, M. *Computing TCP's Retransmission Timer*; Technical Report, RFc 2988; 2000. Available online: https://www.rfc-editor.org/rfc/rfc6298 (accessed on 18 March 2023).
60. Stoer, J.; Bulirsch, R. *Introduction to Numerical Analysis*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013; Volume 12.
61. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canda, 8–10 July 2009; pp. 1–6.
62. Meijer, L. On DOTS: Entity Component System—Unity Software. 2019. Available online: https://blogs.unity3d.com/2019/03/08/on-dots-entity-component-system (accessed on 18 March 2023).