Wireless Personal Communications (2022) 125:367–397
https://doi.org/10.1007/s11277-022-09554-9

Design and Implementation of a Quantitative Network Health Monitoring and Recovery System

Harshit Gujral[1] · Abhinav Sharma[2] · Pulkit Jain[2] · Shriya Juneja[2] · Sangeeta Mittal[2]

Accepted: 23 January 2022 / Published online: 25 March 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
A network health monitoring system focuses on the quantification of the network's health by taking into account various security flaws, leaks, and vulnerabilities. A plethora of propriety tools and patents are available for network health quantification. However, there is a paucity of available research and literature in this field. Thus, in this study, we present an architectural design of a network health monitoring system. The design focuses on the quantification of the network health of each end-user as well as the entire network. The network health score for each end-user is quantified by identifying (1) illicit egress-ingress traffic, (2) anomalous fingerprints, and (3) system-network vulnerabilities based on the NVD-CVSS (National Vulnerability Database, Common Vulnerability Severity Score) standards. An overall network-health score is produced, along with a prevention and recovery mechanism that is triggered upon the detection of an anomaly. The proposed system is implemented in a local area network and has demonstrated to protect the network against various threats successfully. The study is concluded by comparing the proposed tool with the popular propriety tools available in the field. The results outline that the proposed system garners features of open-source tools and enriches them by introducing a state-of-the-art architecture coupled with multiple novel features like exhaustive identification of vulnerability and detection of network aberrations using timers.

Keywords Computer network security · Automatic threat detection · Vulnerability detection in computer networks · Intrusion detection · Vulnerability scanner

✉ Harshit Gujral
harshit@cs.toronto.edu

Abhinav Sharma
sharma1997abhinav@gmail.com

Pulkit Jain
princejain17dec@gmail.com

Shriya Juneja
shriya.juneja20@gmail.com

Sangeeta Mittal
sangeeta.mittal@jiit.ac.in

[1] Department of Computer Science, University of Toronto, Toronto, Canada

[2] Department of Computer Science Engineering and IT, Jaypee Institute of Information Technology, Noida, India

1 Introduction

Network-based applications and services are subject to multiple vulnerabilities, and these vulnerabilities can be exploited for malicious access. Network administrators use various tools to safeguard the network from security threats. Intrusion detection/prevention systems (IDS/IPS) have proven to be an effective solution. As the name implies, these systems emphasize the real-time identification of threats as well as issuing a warning against the potential threats. Several such solutions have been proposed in the literature so far. Besides, several proprietary tools are available for this purpose. However, common drawbacks of these systems are (1) tunable meta-parameters, (2) no provision for the quantification of network health, and (3) limited or no insight into its working and architecture, i.e., especially relevant for propriety tools.

IDS/IPS provides the provision for tuning meta-parameters to minimize human supervision and detect false positives. However, this limits the effectiveness of these systems in detecting granular network patterns. The authors describe the importance of such granular patterns, followed by utilizing them to identify anomalous fingerprints, in their previous work [23]. None of the existing open-source solutions provides the mechanism for assessing and maintaining the overall network health, whereas none of the propriety tools provides an insight into their working and the quantification of network health.

Thus, in this work, we present the architecture of an open-source network health monitoring system. To the best of our knowledge, in the literature, limited consideration has been given to creating a network health monitoring system from scratch. This suggests a monopoly of propriety tools in this field. Here, we emphasize building an open-source network health monitoring system by utilizing novel anomalous fingerprinting identification techniques described by the authors in the previous work [23], as well as open-source tools and libraries. We achieve this by highlighting the research aspects of monitoring architecture, anomaly detection, and vulnerability identification. This would further aid researchers in realizing advanced network health monitoring systems.

The quantification of network health takes into account the health of each device (or end-user, henceforth) in the network. Figure 1 illustrates the focus areas for the quantification of the health of each end-user. Applications and services (active and passive), along with network traffic, are monitored to quantify the health of end-users in a network. An unhealthy end-user serves as a potential threat to the overall network's health, whereas a healthy user adds up to the network's health. The system quantifies network health by utilizing four core modules, namely Ingress and Egress Traffic Monitoring, Vulnerability Detection in end systems, and network fingerprint analysis. The design of the system has the potential to prevent a multitude of threats like port infiltration, malicious egress and ingress traffic, exposure to phishing websites, Denial-of-Service attacks, unpatched or vulnerable services/applications, congestion control, URL sanitization, presence of bot or botnet, and a fragile firewall.

Quantification of network health can serve as a pragmatic way to validate the effectiveness of new security patches and systems applied for protection. Existing works in network protection lack the empirical indices and hence, are unable to facilitate the estimation of the relative risk of adding (or removing) new applications, users, or services to (or from) the network. However, this can be achieved through the real-time quantification of network health. In this work, several indices have been developed to reflect malicious ingress-egress traffic, anomalous fingerprints, and vulnerabilities. A unique

Fig. 1 Area of focus of various modules

feature of the proposed approach is that defense mechanisms are adaptive to the type of vulnerability and timeliness of traffic (real-time or non-real-time transactions) for minimal overheads.

In addition to network health quantification, the proposed work has contributed to the existing state-of-art in several ways. To the best of our knowledge, it is the first network security model aggregating well-established vulnerability scanning tools and data sources like the National Vulnerability Database (NVD), Common Vulnerability Severity Score (CVSS), and Open Vulnerability and Assessment Language (OVAL) to detect signatures of malicious communication. The system employs a combination of active and passive network scanning for different transactional data. For real-time transactions, passive scanning is used due to its virtue of negligible system overhead and the introduction of no additional traffic in the link, whereas for periodic and anomalous transactions, active scanning is used for its virtue of precise and targeted results. Additionally, the layered architecture of the Vulnerability Detector System (VDS) is another chief contribution to this study.

Summarization of novelties introduced in the proposed tool are as follows:

1. The design and implementation of the system (and its various modules) for the quantification of network health while delineating the architecture and research aspects related to it. As part of it, several empirical indices to ensure network security has been developed.
2. Aggregating the data sources, such as NVD, Common, CVSS, and OVAL, for comprehensive detection of vulnerability signatures.
3. The use of benign numbers, such as Round Trip Time (RTT), for anomaly detection and fingerprinting.
4. The mechanism to detect Type-A (system-based) and Type-B (network-based) vulnerabilities using a combination of open-source tools such as OVALDi and Nessus.
5. Overcoming the bottleneck, i.e., high time consumption to scan Type-A vulnerabilities, by exploiting the overlap between Type-A and Type-B vulnerabilities.

The details of the study, its implementation, related works, and the results are organized in the rest of the paper in nine sections. Section 2 discusses some existing network security approaches. Section 3 highlights the aspects of the current study in light of related works. Sects. 4, 5, and 6 discuss the architecture and implementation results of various modules from the perspective of the network administrator and end-users. Section 7 discusses the quantification of network health by combing the results from each module. The network health score of a local area network is quantified, followed by delineating vulnerabilities captured by each module. A comparison of the proposed tool with other propriety tools is drawn in Sect. 8. Section 9 presents the conclusion of the study.

2 Related Work

Several patents have coined the idea and architecture of a Network Health Monitoring System [2, 25, 38, 51], while there are limited literature and open source tools. Thus, due to the unavailability of network health quantification approaches, we study existing approaches of IDSs, Network Fingerprinting, and Traffic Filtering, as well as vulnerability taxonomies and databases, to develop a deeper understanding of the state-of-the-art in these areas.

2.1 Intrusion Detection Systems

Two highly relevant works in this domain are Denning [15] and Staniford-Chen et al. [44]. IDSs are classified based on information sources such as host or network-based IDS. A host-based IDS analyzes events, such as process identifiers and system calls, mainly related to OS information, whereas a network-based IDS analyzes network-related such as traffic volume, service ports, and protocols.

Host-based IDS [8, 10, 27, 29, 49] and network-based IDS [17, 20, 28] have been studied well in the literature. Further, IDS can also be classified as signature-based and anomaly-based detectors [20, 47].

Al-Jarrah and Arafat [4]used neural network-based pattern recognition to identify and classify host sweep and port scan attacks. Host sweep attacks were detected based on a malicious combination of IP source address, destination address, destination port, and protocol type. For the port scan, a neural network has been trained to identify the combinations of source, destination, SYN, ACK, FIN, and other TCP Flags. Wang et al. [52] presented a survey concerning the application of big data analysis techniques for analyzing

considerable network data to create improved IDS. Rodas and To [41] presented a scalable framework for managing authentication logs from various IPS in the network.

2.2 Network Fingerprinting and Traffic Filtering

Network-based anomalies can be detected by active and passive network fingerprinting [42, 43]. Shu and Lee [42, 43] discussed in detail active and passive fingerprinting methods along with tools associated with them. Active fingerprinting approaches are more effective as a predefined set of inputs is used. Active fingerprinting is subject to additional overhead because they tend to introduce additional traffic in the network, whereas the output of passive fingerprinting is based upon stealth observation of all the input/output traces in the network [24]. Probabilistic active and passive approaches for fingerprinting were applied by Arkin and Yarochkin [5] and Beverly [6] for efficiency. OS fingerprinting is essential to determine the OSs being used in a network. This knowledge aids us in determining a list of plausible vulnerabilities associated with a particular OS.

From the end-user's perspective, malicious egress traffic occurs in the form of phishing and malware-hosted websites. They aim to obtain sensitive information by deceiving the end-users. Phishing detection at the individual packet level has been addressed by Afroz and Greenstadt [1], Blum et al. [7], Mohammad et al. [31], Parno et al. [36], Rajalingam et al. [40]. Afroz and Greenstadt [1] described three types of phishing detection, namely content-based, non-content-based, and visual similarity-based.

A visual similarity-based model is developed by Chen el al. [11] that uses screenshots of web pages as input to detect phishing websites. Images are described by using Contrast Context Histogram (CCH), followed by a k-mean algorithm, to cluster the nearest key points. Zilberman et al. [54] described various deployment strategies for scrubbing centers for egress traffic filtering. Dyer et al. [18] developed the first programmable traffic obfuscation system named 'Marionette'. This system is capable of filtering encrypted traffic.

2.3 Vulnerability Taxonomies and Databases

Igure and Williams [26] wrote about taxonomies of attacks and vulnerabilities that exist in computer systems and networks. We follow their proposed taxonomy to design the VDS. It includes analysis of attack impact, attack features, and exploitabilities that might be exploited. Corral et al. [14] proposed an automated vulnerability detection mechanism for a local network of hosts, servers, and applications running in them. The authors proposed to use existing network security tools for various steps of vulnerability analysis. This included Nessus- and Nmap-based service identification, IDS Response, general and specific vulnerabilities. A consensus system that automatically does security testing by embedding these tools was developed. Our solution approach is similar, but more exhaustive.

Gawron et al. [21] proposed creating a local vulnerability database by combining the NVD database, as well as OVAL definitions, to yield more meaningful and accurate results. The authors defined a logical representation of pre- and post-vulnerability conditions. A use case of browser vulnerabilities has been provided. Vu et al. [50] proposed a more detailed and dynamic approach to vulnerability analysis. They have pointed out that apart from the inherent working of vulnerability or exploit, the environment in which that vulnerability could trigger, and its changing characteristics could also be considered in vulnerability analysis. We extend these approaches to profiling vulnerabilities.

Qu et al. [39] first gave the concept of quantifying network vulnerabilities by computing individual host and communication link vulnerability index and aggregating it to the network vulnerability index. The paper also provided the idea of using the index value as a threshold for triggering recovery methods to let the network survive even in the case of an attack. It is complex to envision the combined effect of all vulnerabilities present in the system and preconditions that may lead to the exploitation of one or more vulnerabilities. Gawron et al. [21] gave a solution to visually present the vulnerabilities in terms of the attack graph by taking an example of vulnerabilities present in a browser app. In this work, we have proposed a hybrid IDS that works both at the host and network levels. To the best of our knowledge, an IDS that can quantify the network health is not available in the literature or as an open-source product.

3 The Current Study

Network health is a qualitative term that has been quantified in this study. Sound network health emphasizes the safety of each device from internal and external threats while prioritizing the safety of the network as a whole. Applications and services are secured by developing a layered architecture of modules around them. These modules aim to prevent and identify: illicit egress-ingress traffic, network-system vulnerabilities, along with anomalous fingerprints across active/passive services. Moreover, a recovery mechanism is designed to ensure sound network health when the prevention fails.

The prevention mechanism chiefly employs either or combination of two techniques, one, memorization, and second, anomaly detection. Memorization is a deterministic approach that works by accruing signatures against the existing vulnerabilities. Any potential match to that signature is considered a definite vulnerability. For instance, using Google's safe website lookup to filter benign websites is a memorization technique that uses Google's collection of signatures for benign websites. Along with Google's website lookup, the system also employs signatures from popular vulnerability databases, i.e., NVD and OVAL.

Anomaly detection is a probabilistic technique that flags the end-users based on anomalous behavior. For instance, it involves conducting probabilistic classification of any previously unseen website into benign, suspected, and illicit on the fly. This classification is usually based upon the learning derived from the previously collected signatures. In addition to machine learning classifiers, factors like port scanning, RTTs have been utilized for anomalous fingerprint detection.

Upon the identification of the threat, the network goes into recovery mode. Recovery can be triggered by one or more modules depending upon the nature of the risk. External threats are blacklisted and blocked based upon the severity of the threat. However, internal threats are quarantined from the network until further sanity checks.

The architecture of the system is composed of five modules, as illustrated in Fig. 2. Network health monitoring modules have been designed for end-users as well as a network administrator. Each of these modules contributes to the quantification of network health. Table 1 expounds on the uniqueness of the proposed system's architecture when compared with similar open-source tools and libraries. It is evident from Table 1 that our tool garners various existing features of these open-source tools and enriches them by introducing state-of-the-art architecture coupled with multiple novel features like exhaustive identification of vulnerability, detection of network aberrations using timers, etc.

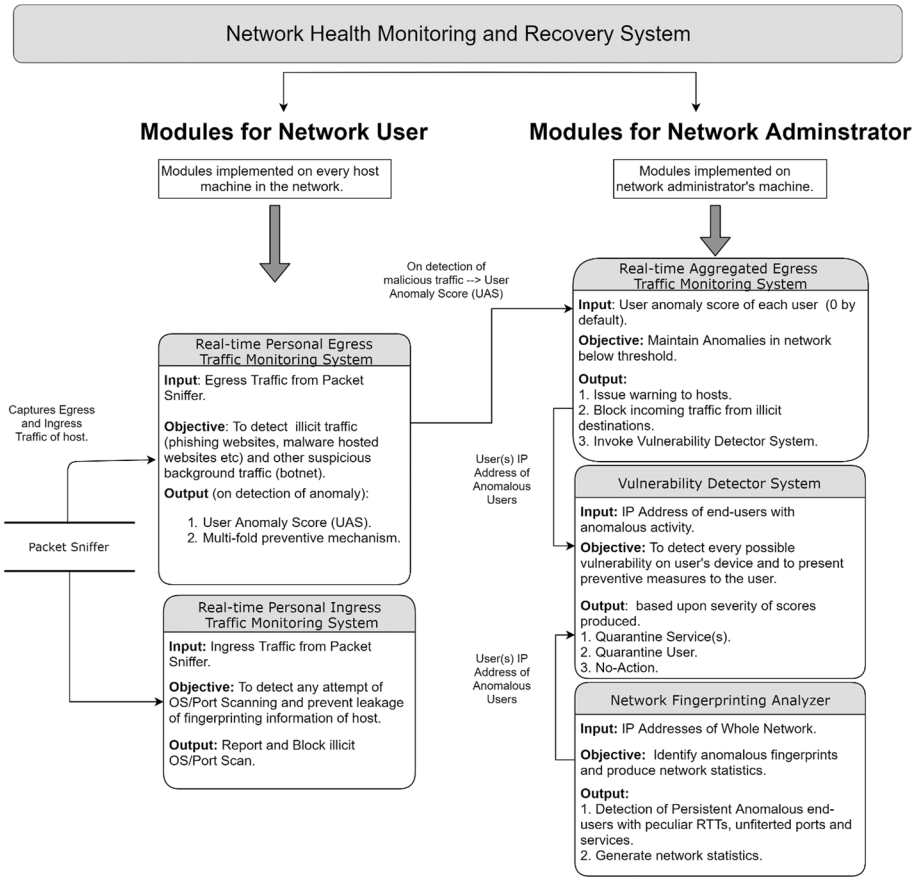


Fig. 2 Data-flow of the proposed network health monitoring system

4 Architecture

The network-health monitoring and recovery system has been implemented as parallel modules for end-users and the network administrator. Figure 2 details the architecture of the system in brief. Egress and ingress traffic at each end device is monitored. Egress traffic is forwarded to the Real-time *Personal* Egress Traffic Monitoring System (RPETMS), while ingress traffic is forwarded to the Real-time *Personal* Ingress Traffic Monitoring System (RPITMS). Results of these modules are aggregated at the network administrator's level for pattern detection and identification.

At the network administrator level, all malicious egress traffic is mapped into the Real-time *Aggregated* Egress Traffic Monitoring System (RAETMS), and Vulnerability Detector System (VDS) is consequentially triggered for an exhaustive identification of vulnerabilities. The system also includes a Network Fingerprinting Analyzer (NFA) module that focuses on the detection of OS-specific anomalies.

The modules are proactive and work towards intrusion prevention to minimize downtime. The anomaly scores are presented on the administrator's dashboard for highly objective network health monitoring. Long-term scores can be used to plan the

Table 1 A comprehensive comparison among open-source tools and the proposed system

Features/Tools	OVALDi	NESSUS basic scan	Detection of malicious egress and ingress traffic	Proposed system
Primary target	Local system vulnerabilities (Type A)	Network vulnerabilities (Type B)	Malicious traffic and phishing websites	Type A, Type B vulnerabilities, malicious traffic, phishing websites, and anomalous fingerprints
Time consumption	Depends on the volume of applications in the system	Depends on active and passive services exposed on the network layer	Depends on the network traffic and usage	A continuous process
Real-time scan	Not feasible	Feasible	Feasible	Feasible
Ease of use	Easy for the network administrator	Easy for the network administrator as well as end-users	Easy for the network administrator	Easy for the network administrator
Cost	Freeware	Proprietary Software	Open-source	Open-source
Visualization of vulnerabilities	As HTML files	On dashboard	No provision	On dashboard
Scheduled monitoring	No provision	Provision is present in the proprietary version	No provision	Provision is present
Complete network scan	No provision	Limited provision of 10 scans in the free version	No provision	Provision is present
Proactive / Reactive tool	Reactive	Reactive (free version)	Proactive	Proactive and Reactive (if aversion fails)
Agility—automatically detects new vulnerabilities	No agility (signature dependence)	Agile	Agile	Agile (except for Type-A vulnerability signatures)

procurement and installation of the additional security infrastructure. The core code of all the modules is publically available at GitHub repository newtein/network_monitoring [34].

5 Architecture of Modules for End-Users

The architecture for end-users comprises RPETMS and RPITMS modules installed on their devices. These modules focus on passive monitoring and receive input from a python-based packet sniffer.

5.1 Real-time Personal Egress Traffic Monitoring (RPETMS)

Browsing phishing websites or malware-hosted websites can deceive end-users into divulging their sensitive information or infecting their system with malware. Considering these threats, all outgoing traffic to phishing or malware-hosted websites is considered illicit egress traffic.

Further, the egress traffic is classified into benign, suspected (suspected-illicit), and illicit. This classification is achieved by implementing a twofold process. First, the egress traffic is filtered using Google's safe browsing lookup, and second, a machine learning model is used to further validate the filtered traffic. Google's safe browsing lookup is a blacklist of suspected phishing, malware, and unwanted software pages by Google [22]. If a website is present in the Google Safe Browsing lookup, then it is categorized as illicit. Otherwise, it is categorized as non-illicit.

There is a fair chance of encountering a new malicious or phishing website that is not yet present in Google's lookup. Thus, the features of all non-illicit websites are further tested using a machine learning model. A binary classification model is trained on the UCI Phishing Website dataset [32, 33] to classify the traffic into benign and suspected-illicit. The classification model is trained using the standard random forest supervised machine learning algorithm implemented in the python-based sci-kit-learn library [37]. The features used in the machine learning model can be classified into three categories: URL-based features, Site reputation-based features, and host-based features.

URL-based features include site information that can be observed from the URL of a website. It has been observed that URLs containing the IP address, @, long URLs, and prefix-suffix generally land on sites with malicious intent. Site reputation-based features include global PageRank and country PageRank obtained from Amazon's Alexa.com [3]. Benign sites have higher PageRank than illicit ones. Host-based features are backed by the empirical observation that the minimum span of a benign domain is 6 months, and illicit websites are registered for a shorter span than this. On the fly extraction of these features is a time-consuming process, which increases the latency. Thus, there is a scope for optimization in the process.

Therefore, as the outcome of this two-phase verification, the egress traffic is categorized into benign, suspected, and illicit with a score of 0, 1, and 2, respectively. Table 2 delineates the strategy to (1) compute the User Anomaly Score (UAS) and (2) categorize egress traffic into Benign, Suspected, or Illicit. These individual UAS are aggregated at the network administrator's device.

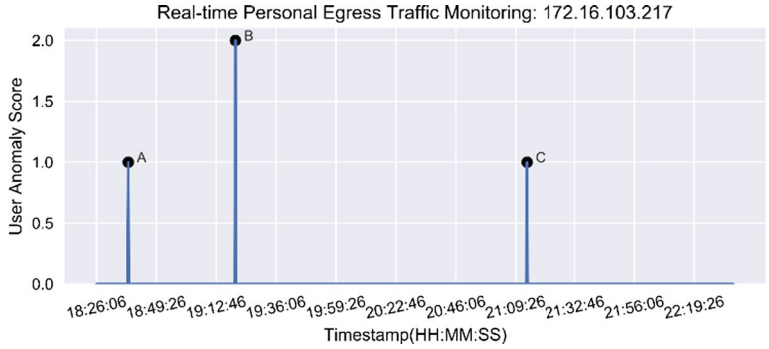
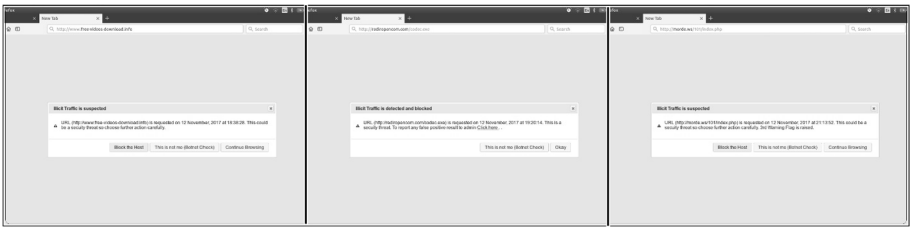


Fig. 3 Real-time Personal Egress Traffic Monitoring Graph for an end-user



(a) **(b)** **(c)**

Fig. 4 Prompt (**a**), (**b**), and (**c**) received by end-user 172.16.103.217 at points A, B, and C, respectively of Fig. 3

Table 2 Categorization of egress traffic

Prediction by the predictive model[a]	Google safe browsing lookup[a]	Resultant user anomaly score (UAS)	Categorization
0	0	0	Benign
1	0	1	Suspected
0/1	1	2	Illicit

[a]0 depicts benign, and 1 depicts a malicious site

5.1.1 Recovery

Whenever traffic with a UAS value of 1 is detected from egress monitoring, a twofold preventive mechanism is invoked. Firstly, a warning against illicit traffic is generated to inform the end-user about malicious activity. It is done to enlighten the end-user of its activity if it is a voluntary activity or end-users device is an unintentional part of a botnet. The end-user is prompted with a choice of further action that includes one of three options: (1) temporarily block the illicit host, (2) execute botnet detection and prevention check, and (3) continue browsing. Secondly, this activity lodges an alert to the network administrator. This alert is lodged in the form of a warning flag against the end-user.

When a packet with UAS of 2 is detected, then this packet is immediately dropped while temporarily filtering further packets from the illicit source. Moreover, a warning-flag is generated against the end-user to the network administrator. An example of the implementation of this module is illustrated in Figs. 3 and 4.

Network simulation of the RPETM module for an end-user machine with IP address 172.16.103.217 is illustrated in Fig. 3. At points A and C, the end-user had encountered a suspected packet (UAS equals 1) and was required to take further action. The subsequent prompt is displayed in Fig. 4a and c, respectively. Similarly, at point B, the end-user had encountered an illicit packet (UAS equals 2). The packet was blocked instantaneously, and its acknowledgment is shown in Fig. 4b. At each of points A, B, and C, a warning flag is lodged with the network administrator. At point C, when the 3rd warning flag is raised against the same end-user, VDS is invoked.

5.2 Real-time Personal Ingress Traffic Monitoring System (RPITMS)

The module primarily focuses on protecting the end-users from scanning attacks launched using active port scanners like Nmap. Each ingress packet is monitored for distinct control flags combination in real-time. Scanning tools probe end-user with a distinct combination of packets and analyze their response to remotely identify sensitive information such as open ports, operating systems, active services, etc. The signature of traces used in these scans is pre-determined from Nmap. These signatures were then reverse engineered to detect the scan type. Further action is determined on the basis of the severity of the identified scan type (Table 3).

The objective of this module is to detect, block, and report fingerprinting events. We have employed a trust-based three-way TCP handshake to detect illicit network scanning or fingerprinting activity. As a result, the source IP address of a packet is categorized as trusted or untrusted IP. When a TCP handshake is being initiated from the end-user to the server, then the server's IP address will be considered as a Trusted IP, whereas when the server initiated the connection, then the server's IP will be regarded as an Untrusted IP.

The payload of packets from untrusted IPs is monitored for a malicious combination of control flags. Table 3 details various Nmap scans and flag combinations associated with them. For instance, -A is a combination of the port scan, operating system scan, and version scan [30]. This scan type provides sensitive device-specific information that can be used to initiate further attacks.

5.2.1 Recovery

The detection of malicious control flag combination in an ingress packet implies a scanning attempt. This results in the execution of a twofold preventive mechanism. First, the response of the end-user corresponding to the scan is blocked. This devoid sharing sensitive information about the end-user with the attacker. Second, the event is also reported to the network administrator. Although port scanning can be remotely initiated, it requires the attacker to be on the same local network as the end-user. When the network administrator detects frequent reports of such scans from multiple IP addresses to the same end-user, then the admin issues an instantaneous but temporary block of that end-user, as this

Table 3 TCP header control flag combination associated with various types of the Nmap scan

Active services		Inactive services	
Type of the scan	Control flag combination	Type of scan	Control flag combination
-A	[SYN], [RST], [ACK], [ACK, FIN], [ACK,PSH], [URG,PSH,FIN], [URG,PSH,SYN,FIN], [ACK, RST]	-A	[SYN], [ACK], [URG,PSH,FIN]
-O	[SYN], [URG,PSH,FIN], [RST], [ACK], [URG,PSH, SYN, FIN]	-O	[SYN], [ACK], [URG,PSH,FIN]
-sS	[SYN], [RST]	-sS	[SYN]
-Pn	[SYN], [RST]	-Pn	[SYN], [RST]
-sT	[SYN], [ACK], [ACK, RST]	-sT	[SYN]
-sV	[SYN], [RST], [ACK], [ACK, FIN], [ACK,PSH],[ACK, RST]	-sV	[SYN]
-sA	[ACK]	-sA	[ACK]

indicates a DDoS attack underway. This implies downtime to a particular end-user but ensures the overall network health.

6 Architecture of Modules for the Network Administrator

The architecture for end-users comprises three modules: RAETMS, NFA, and VDS. These aggregator modules are installed on the network administrator's device.

6.1 Real-time Aggregated Egress Traffic Monitoring System (RAETMS)

The module is an aggregated apparatus of the end-user module RPETMS (discussed in Sect. 5.1). The UASs across all the end-users are collected from RPETMS. Notably, the UAS for an end-user is reported to RAETMS only after the detection of illicit egress traffic on the end-user's device. A periodic Mean Anomaly Score (MAS) is calculated in every 60 s. For n end-users, UAS and the number of times UAS is reported (Instances) are aggregated in time t. Thus, the MAS for ith end-user can be calculated by Eq. (1),

$$\text{Mean Anomaly Score} = \frac{\sum_0^t \text{UAS}[i]}{\sum_0^t \text{Instances}[i]}, \tag{1}$$

where 't' is the monitoring period. A global parameter Aggregated Anomaly Threshold (AAT) is defined for the overall network. It acts as a threshold for MAS, and upon breaching this threshold, the end-user is considered vulnerable to the network. Figure 5 delineates the flowchart of RAETMS.

6.1.1 Recovery

When MAS of the end-user crosses AAT at any instance, then a threefold preventive mechanism is invoked. First, the network instantaneously blocks further packets from the illicit source. Second, the VDS is invoked against the end-user. Third, a warning is issued to the end-user describing the nature of the illicit activity. In addition, a warning flag against the end-user is lodged with the network administrator. An example of the implementation of

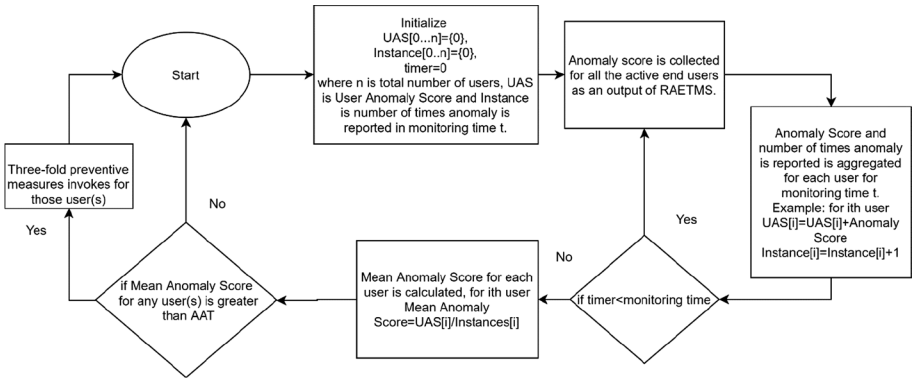


Fig. 5 Flowchart of the real-time aggregated egress traffic monitoring system

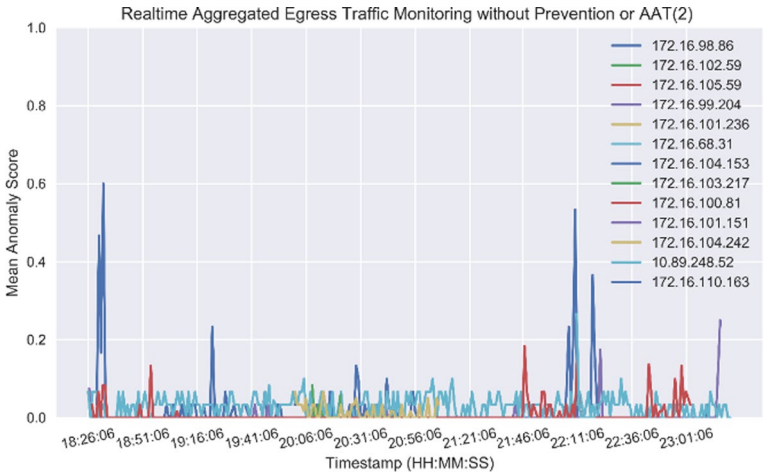


Fig. 6 Mean anomaly score of the network computed by the real-time egress traffic monitoring system without any preventive mechanism

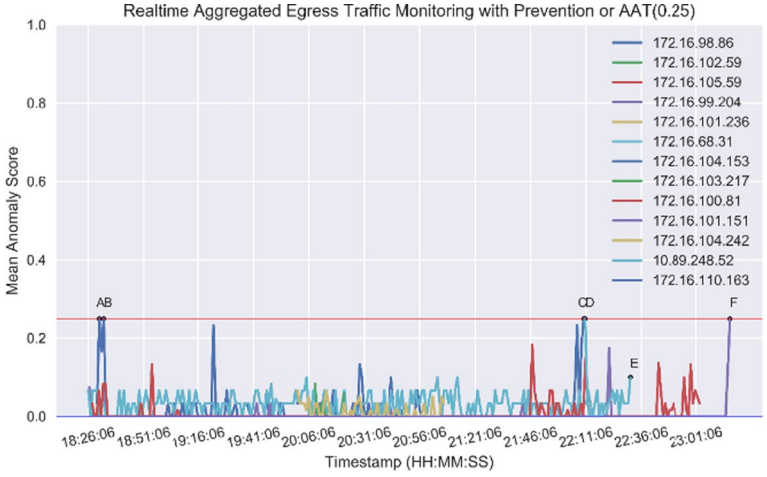


Fig. 7 Mean anomaly score of the network computed by the real-time egress traffic monitoring system with a threshold-based preventive mechanism

this module with and without threefold preventive measures is illustrated in Figs. 6 and 7, respectively.

Figure 6 illustrates the MAS of 13 end-users, obtained by RAETMS in a network simulation over a window of about 4.5 h, without any prevention mechanism. Figure 7 shows a similar network simulation with a threefold prevention mechanism using AAT as 0.25. The low value of AAT has been kept to keep false negatives at the minimum. Notably, most threats would be successfully detected and remedied at the end-user's level by RPETMS.

Figure 7 shows the observed simulation of RAETMS with a threefold preventive mechanism at AAT(0.25). This mechanism is invoked at points A, B, C, D, E, and F. The peaks previously observed in Fig. 6 were successfully remedied at the threshold. Whenever a high MAS is detected, the VDS is simultaneously invoked to determine the underlying cause of the anomalous behavior.

6.2 Network Fingerprint Analyzer

It produces an active fingerprint of the network and is invoked periodically in 4 h. It utilizes '-A' command of the Nmap to conduct active fingerprinting. It includes a version scan (-sV), an operating system scan (-O), a scripting engine scan (-sC), along with traceroute details. These traceroute details include RTT and the number of hops [30]. The output of Nmap, recorded in the XML format, is parsed and divided into 4 logical categories, as depicted in Table 4. The purpose of the module is to produce network statistics while identifying anomalous fingerprints in the network. The end-users with anomalous fingerprints are further scanned using the VDS.

Network statistics include the number of active hosts, OSs, and active services in the network. The module primarily focuses on three anomalous fingerprints. First, unusually high RTT while establishing a connection to an end-user. Second, the presence of unfiltered ports that indicate a fragile-firewall. And third, the presence of vulnerable or outdated services. In addition, the module is extended for the prediction of IP-ID sequences and network traffic using correlations among RTTs [23].

6.2.1 Recovery

The recovery mechanism is adaptive to the type of anomaly identified. Firstly, an end-user with usually high RTT is temporarily quarantined from the network. Secondly, the presence of unfiltered ports is remedied by blocking those ports until the end-user revamps its firewall. And finally, an update service alert is produced to the end-users with outdated services. It is followed by quarantining the end-user until the issue persists. Besides, in all the cases, the VDS is invoked for the end-user.

Table 4 Logical categorization of the Nmap scan output

T1: Basic Information	T2: Unfiltered Port Information	T3: OS Information	T4: Traceroute Information
MAC address	MAC address	MAC address	MAC address
IPv4 address	Port Number	Operating System	Number of Hops
Scan Start Time and End Time	Protocol	OS-Family	Round Trip Time (RTT)
Number of Filtered Ports and Unfiltered Ports	State (Open/Close)	Vendor	Time to Live (TTL)
Vendor	Running Service Name and Version		

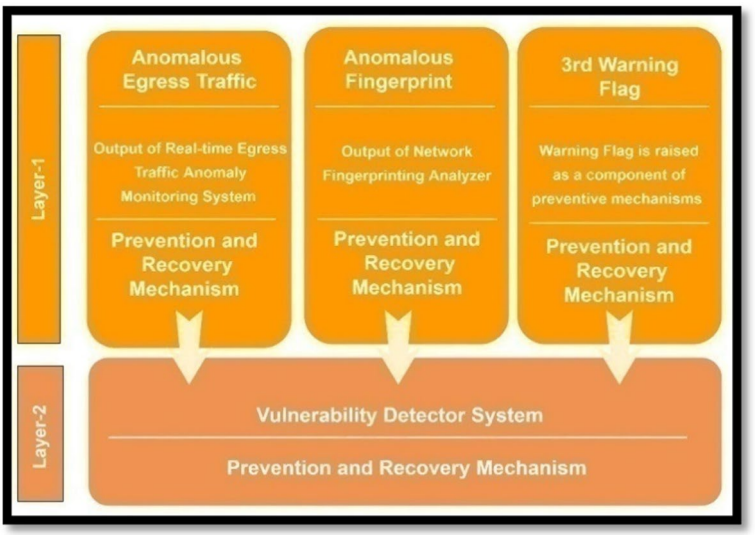


Fig. 8 Conditions for invoking the vulnerability detector system

6.3 Vulnerability Detector System (VDS)

The VDS aims at scoring an end-user device based upon system and software vulnerabilities present. We have divided vulnerabilities into four categories, namely Type A, Type B, Type A.B, and Type A + B (Table 6). The VDS is triggered alongside the detection of various anomalies in the network, as illustrated in Fig. 8. An Anomaly Flag (AF) is raised whenever an anomaly is identified. The anomaly occurs in the system in three forms: Anomalous Egress Traffic (AT), Anomalous Fingerprinting (AF), and Third Warning Flag (3WF).

An anomalous system state can represent the presence of vulnerabilities in the system software and services. However, all of these are not equally damaging. Common Vulnerability Scoring System (CVSS) v3.0 standard defines the severity of vulnerabilities from Low to Critical according to the risks associated with them [13]. The National Vulnerability Database (NVD) [9, 35] is the repository of vulnerability management data represented using the Security Content Automation Protocol (SCAP). The NVD provides CVSS 'base scores', which represent the inherent risk associated with each vulnerability.

6.3.1 Identification of Type-A Vulnerabilities

The Type-A vulnerabilities can be identified by assessing the local device and executing tools like OVALDi. The OVALDi (Open Vulnerability and Assessment Language Interpreter definitions) is a command-line tool. OVAL definitions help to determine the presence of vulnerabilities or configuration issues on the system [21]. We have parsed the XML output of OVAL and divided it into two logical schemas, as shown in Table 5a. The CVE-ID obtained from the output is mapped to the NVD to obtain the corresponding CVSS Score and Risk Factor (Table 6).

Table 5 (a) Logical schema designed from OVAL scan output (b) logical schema designed from NESSUS basic scan output

(a) Logical schemas from the OVAL scan output		(b) Logical schema from the NESSUS scan output	
Basic details	Vulnerability specific details	Basic details	Vulnerability specific details
MAC address	CVE ID	MAC address	CVE ID
OS name and version	CVSS score (Obtained by mapping CVE ID to NVD)	OS name and version	CVSS score and risk factor
Scan start time	Risk factor (Low, Medium, etc.)	Scan start time	Port and protocol
Scan end time	Vulnerability class and title	Scan end time	Service, version, and synopsis

6.3.2 Identification of Type-B Vulnerabilities

The Type-B vulnerabilities of active services can be remotely discovered by using network scanning tools like Nmap, Nessus, etc. The Type B vulnerability score is calculated from the Nessus Basic Scan for vulnerability exploits on the open ports. These vulnerability tests are written in the NASL (Nessus Attack Scripting Language), which is a scripting language optimized for the custom network interaction [16]. The results obtained from the scan were divided into two logical schemas, as described in Table 5b. The Type B vulnerability score is very crucial as it is associated with active services that are either idly listening to connection or communicating over the network through ports [46].

The vulnerabilities of Type-A nature are present in the end user's device in the form of vulnerable or outdated software, services, or applications. These can only be compromised by directly accessing the target device. In contrast, the Type-B vulnerabilities are subject to services or applications communicating over the network. These can be remotely exploited, which makes patching these vulnerabilities crucial. Further, the Type A.B vulnerabilities (the intersection of Type A and Type B vulnerabilities) are essential as they define the most prominent and easily compromisable active vulnerabilities in a system. Lastly, the Type A + B (the union of Type A and Type B vulnerabilities) defines an exhaustive set of vulnerabilities. The VDS analyzes vulnerabilities of Type A, B, A.B (most prominent), and A + B (exhaustive). Notably, Type B and Type A.B vulnerabilities can be remotely exploited.

6.3.2.1 Recovery The recovery steps are driven by results derived from the analysis of detected vulnerabilities. Firstly, when Type B and Type A.B scores are in the medium-risk range, then the vulnerable services of the end-user are blocked by the network administrator. This command is referred to as the Quarantine Service(s). Secondly, when Type B and A.B scores indicate high-risk, then the user is temporarily blocked from the network until it follows a set of recovery measures. These measures are collected from the Nessus and Oval Community. This command is referred to as the Quarantine User.

7 Network Health Score

Results of the proposed network health monitoring and recovery system were observed on a semi-controlled network of 20 end-users and one network administrator. Here, a semi-controlled network implies that although an end-user is free to install or use any service,

Table 6 Scoring four sets of vulnerabilities

Sno	Vulnerability	Scoring
1	Type A	$Type\ A\ Vulnerability\ Score = \frac{\sum CVSS\ Score * Weight(Risk\ Factor)}{\sum Weight(Risk\ Factor)}$, where function Weight(Risk Factor) equals 4, 3, 2, and 1 for risk factors Critical, High, Medium, and Low, respectively. For the CVSS Score, the risk factor has been obtained by mapping the corresponding CVE-ID of vulnerability to NVD
2	Type B	$Type\ B\ Vulnerability\ Score = \frac{\sum CVSS\ Score * Weight(Risk\ Factor)}{\sum Weight(Risk\ Factor)}$, where function Weight(Risk Factor) equals 4, 3, 2, and 1 for risk factors Critical, High, Medium, and Low, respectively, and the CVSS Score and Risk Factor are obtained from the Nessus Basic Scan
3	Type A.B	$Type\ A.B\ Vulnerability\ Score = Type\ A \cap Type\ B\ Vulnerabilities$ Scoring is done by using the above formulas by keeping into account common vulnerabilities that are present in both Type A and B sets
4	Type A + B	$TypeA + BVulnerabilityScore = TypeA \cup TypeBVulnerabilities$ Scoring is done by using the above formulas by keeping into account all unique vulnerabilities that are present in both Type A and B sets

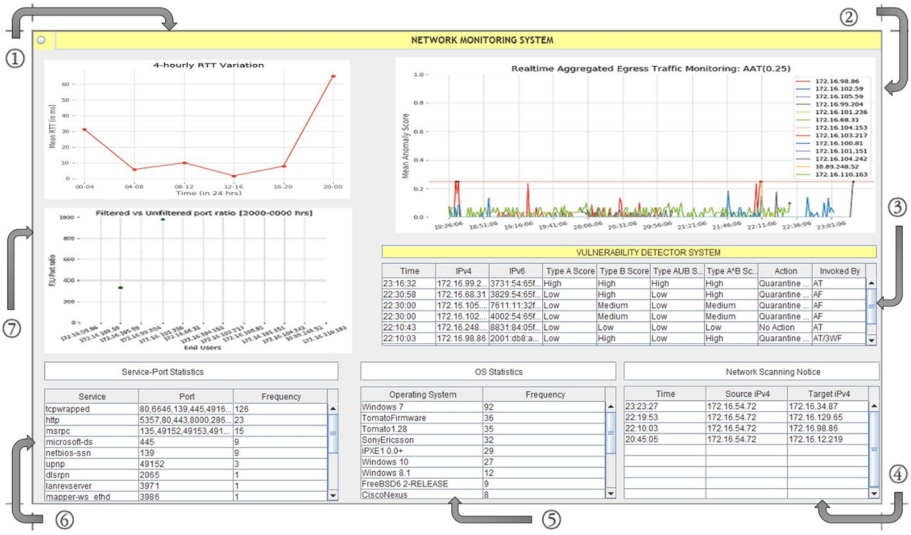


Fig. 9 Network Administrator's Dashboard of the proposed system with labeled components

VULNERABILITY DETECTOR SYSTEM

S.No.	Time	IPv4	IPv6	Type A Score	Type B Score	Type AUB Score	Type A^B Score	Action	Invoked By
1	23:16:32	172.16.99.204	3731:54:65fe:2::a7	High	High	High	High	Quarantine User	AT
2	22:30:582	172.16.68.31	3829:54:65fe:2::a7	Low	High	Low	High	Quarantine User	AF
3	22:30:00	172.16.105.59	7611:11:32fe:2::a7	Low	Medium	Low	Medium	Quarantine Service(...	AF
4	22:30:00	172.16.102.59	4002:54:65fe:2::a7	Low	Medium	Low	Medium	Quarantine Service(...	AF
5	22:10:43	172.16.248.52	8831:84:05fe:2::j9	Low	Low	Low	Low	No Action	AT
6	22:10:03	172.16.98.86	2001:db8:a0b:12f0::1	Low	High	Low	High	Quarantine User	AT/3WF
7	18:33:03	172.16.98.86	2001:db8:a0b:12f0::1	Low	Low	Low	Low	No Action	AT
8	18:31:03	172.16.98.86	2001:db8:a0b:12f0::1	Low	Medium	Low	Medium	Quarantine Service(...	AU

Fig. 10 Vulnerability detector system action table

application, or environment, the consequence is determined by the network administrator. Figure 9 shows the network administrator's dashboard, which comprises seven labeled components.

The first component is a 4-hourly RTT variation graph, and the 2nd component is a real-time aggregated usage graph observed from RAETMS (also see Fig. 7). For clarity, only IPs with significant activities are shown. The 3rd component shows the corresponding VDS action table (also see Figs. 10, 12). Components 4, 5, and 6 represent network scanning block notices from the RPITMS, a Service-Port frequency meter, and an OS-Frequency meter, respectively. Finally, Component-7 portrays the filtered/unfiltered ratio graph observed at every 4th hour.

7.1 Quantification of network health

The vulnerabilities detected by each module are represented by a terminal score or overall health score for each end-user. It reflects the health of an end-user at any instance.

The real-time weighted egress traffic score or $\boldsymbol{\alpha}$ is calculated for an end-user by Eq. (2),

$$\alpha = \frac{4 * (\textit{No. of Illicit Packets}) + 1 * (\textit{No. of Suspected Packets})}{5}, \tag{2}$$

where the number of illicit and suspected-illicit packets is calculated by RAETMS.

To integrate the effect of the NFA module, an anomalous fingerprinting score or $\boldsymbol{\beta}$ is calculated for each end-user at periodic intervals by Eq. (3),

$$\beta = \frac{10 * (Number\ of\ Unfiltered\ Ports) + 10 * [G - 1]}{20}, \tag{3}$$

where G is derived from Eq. (4).

$$G = \left(\max \left(\frac{RTT(User)}{E[RTT(User)]}, \frac{RTT(User)}{E[RTT(Network)]} \right) \right) \tag{4}$$

In Eq. (3), E[RTT(User)] is the expected value or mean of end-user's RTT collected over a period of time and E[RTT(Network)] is the expected value of the network's RTT. The term $\frac{RTT(User)}{E[RTT(User)]}$ will reflect the anomalous variation of end-user from its own expected behavior (normal behavior), while the term $\frac{RTT(User)}{E[RTT(Network)]}$ corresponds to an anomalous variation of the end-user from the network.

The weighted vulnerability score or γ is produced by VDS for each user by Eq. (5).

$$\gamma = \frac{4 * (Type\ A.B\ Score + Type\ B\ Score) + 1 * (Type\ A\ Score + Type\ A + B\ Score)}{5} \tag{5}$$

The number of scanning attempts or $\boldsymbol{\theta}$, makes that the target end-user more vulnerable,

$$\theta = No.\ of\ scanning\ attempts\ against\ user. \tag{6}$$

The terminal score corresponding to each end-user is calculated by a weighted summation of scores of individual modules.

$$Terminal\ Vulnerability\ Score = \frac{\alpha + \beta + 2 * \gamma + 2 * \theta}{6} \tag{7}$$

These empirical weights are calculated after a series of network-simulation experiments. A high terminal score corresponds to the high vulnerability of the end-user. The observed terminal score of the end-users is depicted in Fig. 11. Furthermore, an overall-network

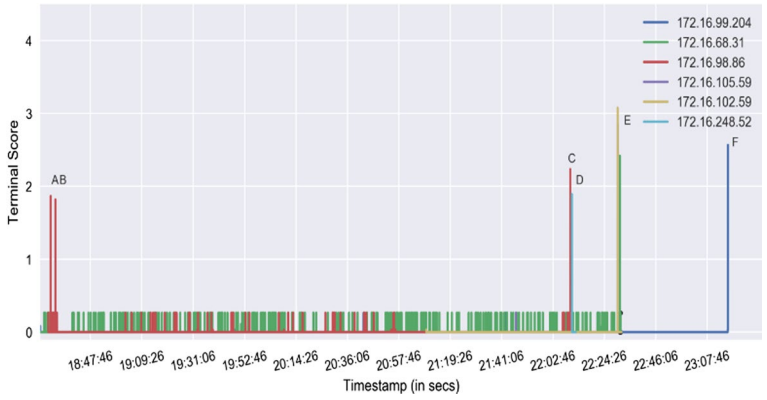


Fig. 11 Observed terminal score of the hosts

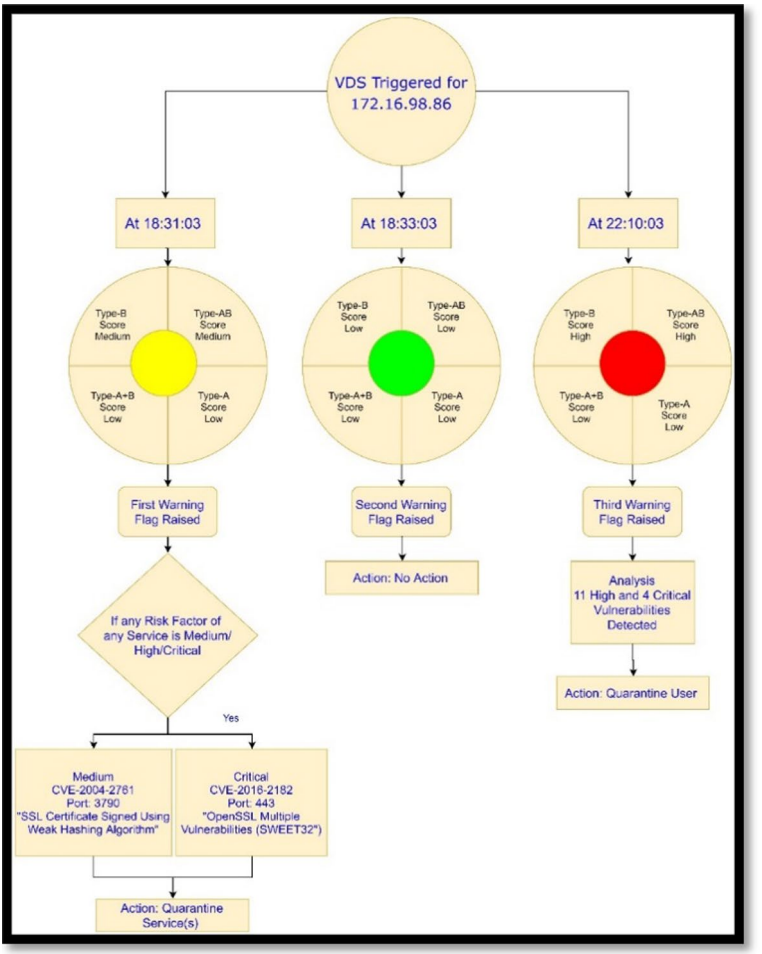


Fig. 12 VDS Stimulation for end-user 172.16.98.86 at points A, B, and C of Fig. 7

score is produced by the expectation value, i.e., the mean of terminal-score of all active end-users by the Eq. (8).

$$Overall\ network\ health\ score = E\left[Terminal\ Score\ of\ \ all\ active\ end\ users\right] \tag{8}$$

Points A, B, and C in Fig. 11 correspond to the same end-user with IPv4 address '172.16.98.86'. The actions taken by the VDS for this end-user are summarized in Fig. 12. Point D represents end-user '10.89.248.52' at 22:10:43; and the VDS has resulted in no-action due to the low-risk factor of detected vulnerabilities. Point F represents the end-user '172.16.99.204' at 23:16:32; and the VDS has resulted in Quarantine User due to high Type B and Type A.B scores. The aforementioned data is also depicted in the VDS Action Table in Fig. 10 (also see component-3 of Fig. 9). At Point E, for the end-users 172.16.102.59 and 172.16.105.59, a finite port ratio is detected (Eq. 9, Component 7, and

Fig. 9). The VDS resulted in Quarantine Service(s) due to a medium risk of Type B and Type A.B vulnerabilities.

7.2 Hosts with Anomalous Egress Traffic

Real-time Egress Traffic for the end-user '172.16.103.217' is depicted in Fig. 3. At points A (18:38:28), B (19:20:14), and C (21:13:52), VDS is invoked as a part of the threefold preventive and recovery mechanism. At all these points, the VDS has resulted in the Quarantine Service(s), i.e., vulnerable services running at that instant are blocked.

The egress traffic is monitored at the end-user level and is further aggregated at the administrator level, as shown in Fig. 3. At points, A, B, C, D, and F of Fig. 7, malicious egress traffic is detected, and AAT is kept at 0.25. As a part of the threefold preventive mechanism of RAETMS, all packets from the host of malicious traffic are temporarily blocked, a warning flag is raised, and the further action is determined by the VDS (Figs. 11, 12).

In Fig. 12, Points A, B, and C correspond to the end-user with IPv4 address '172.16.98.86'. At point A (18:31:03), the VDS results in quarantine service due to the presence of Type B and A.B vulnerabilities. The first warning flag is raised against the end-user to the network administrator. The identified vulnerabilities are CVE-2004-2761 and CVE-2016-2182. These vulnerabilities are of medium and critical nature, respectively. At point B (18:33:03), no action is taken by the VDS due to low Type B and A.B scores, but the second warning flag is raised. At point C (22:10:03), Type B and Type A.B lie in the high-risk factor zone, and the third earning flag is raised. As a consequence of the 3rd warning flag, the end-user is quarantined. This is also evident by comparing the traffic after Point C in Figs. 6 and 7, respectively.

7.3 Hosts with Anomalous Ingress Traffic

Component-4 of Fig. 9 represents malicious ingress traffic in the network. This is a consequence of -A and -O Nmap scans. As a part of the twofold preventive mechanism, response from the end-user is blocked so that no information will be shared and the source IP of the scan is detected. This IP address is automatically blocked by adding it to the IP table blacklist by using the command,

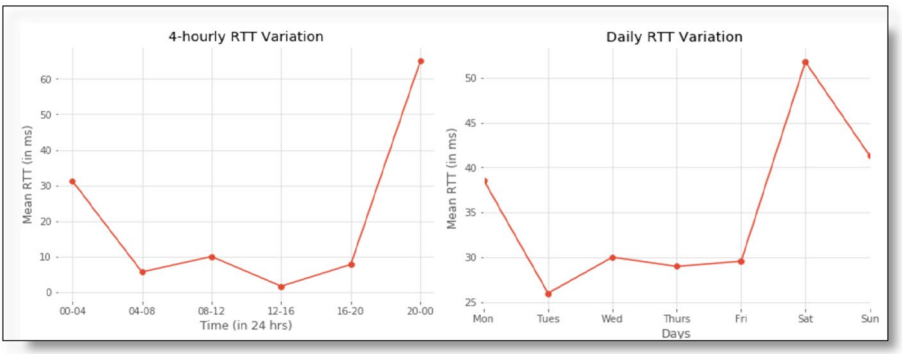
$$\text{ipTables} - \text{A INPUT} - \text{s } 172.16.54.72 - \text{j DROP},$$

where 172.16.54.72 is the IP addresses to be blocked. Thus, any foreseeable packets from this IP would be dropped by the system. This is a temporary block determined by the network administrator.

7.4 Hosts with Anomalous Fingerprint

The NFA module performs the active fingerprinting. We have primarily focused on three components of an anomalous fingerprint: anomalous RTT, filtered to unfiltered port ratio or presence of unfiltered ports, and the presence of vulnerable services in the network.

Figure 13a, b depicts 4-h mean RTT variation and daily RTT variation in the network, respectively. These RTT variations are also a part of the network administrator's dashboard



(a) **(b)**

Fig. 13 **a** 4-h Variation **b** Daily RTT variation

depicted by the 1st component of Fig. 9. The RTT depicts the duration between sending a request (an SYN packet) and receiving the response (an ACK packet). A high RTT indicates congestion in the network. The RTT of every end-user is monitored, and the mean RTT is displayed to the network administrator. When the RTT of a user increases significantly from other users in the network, then as a deliberate and preventive step against the congestion, the end-user is temporarily blocked, and the VDS is invoked.

At Point-E of Fig. 7, the end-user with IP address '172.16.68.31' resulted in a sudden increase in the RTT. As a preventive measure, the VDS is triggered by the NFA (AF). The VDS resulted in a high Type B and Type A.B scores. Thus, the end-user is quarantined (also see the second row of Fig. 10).

$$Anomalous \Rightarrow \frac{Number\ of\ Filtered\ Port}{Number\ of\ Unfiltered\ Ports} \neq \infty(infinity) \tag{9}$$

When filtered to unfiltered port ratio is finite or unfiltered ports are not zero, then as part of the preventive mechanism, the VDS is triggered (see point E, Fig. 11, Eq. 9). The 7th component of Fig. 9 outlines the graph of filtered to unfiltered-port ratio.

The network statistics are recorded in components 5 and 6 of Fig. 9. Component-6 is used to broadly monitor services and their frequency that are running in the network. When a vulnerable service is detected, then it is blocked as a preventive measure, while component-5 provides the OS statistics of the hosts in the network.

8 Comparison with the Contemporary Tools

The preceding sections have widely discussed the exhaustive architecture of the proposed system. In addition, several examples were outlined to test the performance of the modules. Table 7 delineates the empirical statistics observed by executing the system for an arbitrary end-user. It clearly illustrates that the system stands out in comparison with the OVALDi and Nessus. The proposed system focuses on an in-depth classification of identified vulnerabilities, followed by quantifying network health using scores gathered from each module.

Table 7 Experimental results of capturing vulnerabilities

Features	OVALDi	NESSUS Basic Scan	Proposed system
Execution mode	Manual	Manual	Automatic
Target vulnerabilities	Type A	Type B	Type A and Type B
The number of vulnerabilities detected	425	65	Type A: 425 Type B: 65 Type A + B: 470 Type A.B: 20
Output	Lists all the vulnerabilities in an XML and HTML file	Lists all the vulnerabilities in a CSV file (along with other formats)	VDS calculates weighted vulnerability score using a predefined vulnerability classification. Further, a network health score for the end-user is derived
Prevention	No provision	No provision	Every module has a preventive mechanism
Recovery	No provision. (Provides details of the vulnerability but fails to provide a user-specific solution)	It provides a solution to the vulnerability in the form of a text file	Disconnects the end-user from the network. Until the end-user resolves the vulnerability based upon the recommendation provided by the system
Approximate time taken to detect vulnerabilities	1 h 10 min	11 min	Executes continuously and flags suspicious activity for an in-depth analysis

In addition to the preventive mechanism of each module, the proposed system is equipped with an integrated mechanism for recovery.

In addition to the open-source tools like OVALDI and NESSUS, we aim to extend the comparison to popular propriety tools in the network security arena. Cisco offers a proprietary closed-source network health monitoring tool [12]. It captures network health through real-time monitoring of devices in the network. It determines the health by focusing on the availability, performance, and capacity of the devices. Any detected issue is classified as 'Chronic', 'Critical', or 'Service affecting Issue'. The health ratings are computed by adjusting these issues into an empirical formula, which yields a risk index. A complementary health index is derived from the risk index. The average of the health indices across all the issues provide a final overall health rating for the network. Notably, the empirical formula and specifics of the working of this proprietary tool are largely unbeknownst to the research community. The primary focus of the tool is to keep the network up and running. This virtue makes it a good fit for data centers and controlled environments where maximum yield is the necessity.

On the contrary, the proposed system derives network health from the health of each end-user. It quarantines the end-user when the end-user becomes a threat to the network. Thus, the availability, performance, and capacity of the network for each end-user is not the goal of our system. The proposed system aims to ensure the network's security in a semi-controlled environment where the end-users are free to install/access various kinds of services. This virtue of our tool makes it a good fit for the corporate and academic environment.

Although our tool has established novelty on several parameters, it did not explore some of the widely implemented and popular features of the network security tools due to the limited bandwidth available for our research. Some of these features are listed below.

8.1 Ability to Monitor a Wide Range of Devices

Tools, such as SolarWinds NPM (Network Performance Monitor) [53] and Nagios Core [19], are known for monitoring a wide range of devices such as mobiles, printers, scanners, etc. along with personal computers. Both of these tools come with a proprietary license, and it becomes costly with the integration of additional features. For example, many features are not available on a free version of Nagios. The features such as wizards or interactive dashboards are available on Nagios XI, which has a proprietary license, whereas our system is focused on monitoring personal computers only.

8.2 Customized Alerts

Nagios sends alerts whenever a critical infrastructure component fails and recovers, thereby providing administrators with a notice of important events. These alerts can be delivered via email, SMS, or custom script. This is an essential development from Nagios as Nagios cannot manage the network but monitor the network. In addition to monitoring, a typical network management system manages the accounting, configuration, and security aspects of the network. Although our system invokes a wide range of recovery mechanisms whenever an illicit activity is identified, currently, our system has no provision for providing alerts via email or SMS.

8.3 Collection of hardware data

Zabbix is a free FOSS software [48]. It focuses on collecting network and hardware data to provide comprehensive reporting. The hardware data comprises the CPU, memory, and disk metrics. While our system is focused on collecting data from the network, it aims to quantify the health of each end-user and the network.

8.4 Firewalls Management

Panorama is a tool that manages all the firewalls [45] irrespective of where they are: at the perimeter, in a data center, or the cloud. Adding new firewalls or combing firewalls to Panorama is easy. It can push a lot of the configuration and settings so that the end-user does not have to do it manually. While our decentralized system manages a single firewall that is implemented uniformly over the network, there is a scope of firewall enhancement in our system on the lines of Panorama.

8.5 Customizable Visualizations

SolarWinds NPM is famous for presenting customizable dashboards for the network administrator. It provides detailed reports and graphs derived from the analytical data, whereas our system provides a limited dashboard to the network administrator. The dashboard of our system is chiefly focused on presenting the network administrator with the Network Health Score, i.e., a definitive score derived from multiple modules. Besides, the dashboard comprises visualizations from each module.

To sum up, our focus is not to provide a popular network monitoring system but to broaden the foundation for the research community. Throughout our research and literature survey, we have noticed a lacuna in the literature related to Network 'Health' Monitoring Systems. On the contrary, there exist multiple propriety tools and patents. Hence, this work is a step forward to quantify health of the network while highlighting all the research aspects relating to it.

9 Conclusion

The study proposed an architecture for the network health monitoring system. The quantification of network health is performed by accounting for three kinds of network security threats: malicious web-traffic (egress-ingress), computer vulnerabilities (network-system), and anomalous device fingerprints. The proposed system is implemented for end-users as well as the network administrator. The system comprises five modules: two modules installed on the end-user devices and three modules installed on the network administrator device. All the modules are equipped with the prevention, detection, and recovery mechanisms. As a part of each module, several empirical indices have been developed to profile an end-user. Then, indices from each module are combined to reflect the overall health of an end-user. The expected value of health of each end-user corresponds to the overall network health. Results of implementation of this system on a local-area network have indicated that the proposed system is useful in curbing attacks and objectively depicts the security status of the network at any time.

Author Contributions HG: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing, and Visualization. AS: Software, Formal analysis, and Investigation. PJ: Software. SJ: Software. SM: Validation, Writing - Original Draft, Writing - Review & Editing, and Supervision.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Afroz, S., & Greenstadt, R. (2011, September). Phishzoo: Detecting phishing websites by looking at them. In 2011 IEEE fifth international conference on semantic computing (pp. 368–375). IEEE.
2. Aki, Y., & Saito, H. (2008). U.S. Patent No. 7,353,269. Washington, DC: U.S. Patent and Trademark Office.
3. Alexa Internet, Inc (1996), Web traffic and ranking site, https://alexa.com, Retrieved on 12 December, 2017.
4. Al-Jarrah, O., & Arafat, A. (2014). Network intrusion detection system using attack behavior classification. 2014 5th International Conference on Information and Communication Systems, ICICS 2014. https://doi.org/10.1109/IACS.2014.6841978
5. Arkin, O., &Yarochkin, F. (2002). Xprobe v2. 0: A "Fuzzy" Approach to Remote Active Operating System Fingerprinting. 2011–06]. http://www.sys-security.com/archive/papers/Xprobe2.pdf.
6. Beverly, R. (2004). A robust classifier for passive TCP/IP fingerprinting. In *International workshop on passive and active network measurement* (pp. 158–167). Springer, Heidelberg. https://doi.org/10.1007/b96961
7. Blum, A., Wardman, B., Solorio, T., & Warner, G. (2010). Lexical feature based phishing URL detection using online learning. In *Proceedings of the 3rd ACM workshop on artificial intelligence and security-AISec'* (vol 10, pp. 54). https://doi.org/10.1145/1866423.1866434
8. Boer, P. De, &Pels, M. (2005). *Host-based intrusion detection systems*. Amsterdam University.
9. Booth, H., Rike, D., & Witte, G. A. (2013). The National Vulnerability Database (NVD): Overview (No. ITL Bulletin-).
10. Chawla, A., Lee, B., Fallon, S., & Jacob, P. (2018, September). Host based intrusion detection system with combined CNN/RNN model. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 149–158). Springer, Cham.
11. Chen, K. T., Chen, J. Y., Huang, C. R., & Chen, C. S. (2009). Fighting phishing with discriminative keypoint features. *IEEE Internet Computing, 13*(3), 7269.
12. Cisco (2010). Network Health Framework: A Proactive Approach, 2010. Software. https://www.cisco.com/en/US/services/ps6889/Cisco_NHFWhitePaper.pdf.
13. CVSS, Common Vulnerability Scoring System v3.0: Specification Document, https://www.first.org/cvss/specification-document , Retrieved on 12 December, 2017.
14. Corral, G., Zaballos, A., Cadenas, X., &Grane, A. (2005). A distributed vulnerability detection system for an intranet. In *CCST'05. 39th annual 2005 international carnahan conference on security technology, 2005.* (pp. 291–294). IEEE. https://doi.org/10.1109/CCST.2005.1594863
15. Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering, 13*(2), 222–232. https://doi.org/10.1109/SP.1986.10010
16. Deraison, R. (2000). *The nessus attack scripting language reference guide*. Tenable Network Security, Inc.
17. Deshpande, P. S., Sharma, S. C., & Peddoju, S. K. (2019). A network-based intrusion detection system. In *Security and data storage aspect in cloud computing* (pp. 35–48). Springer, Singapore.
18. Dyer, K. P., Coull, S. E., &Shrimpton, T. (2015). Marionette: A programmable network traffic obfuscation system. In: *USENIX security symposium* (pp. 367–382).
19. Galstad, E., (2002). Nagios. Software. https://www.nagios.org/

20. García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers and Security, 28*(1–2), 18–28. https://doi.org/10.1016/j.cose.2008.08.003
21. Gawron, M., Amirkhanyan, A., Cheng, F., &Meinel, C. (2015). Automatic vulnerability detection for weakness visualization and advisory creation. In *Proceedings of the 8th international conference on security of information and networks-SIN'* (vol. 15, pp. 229–236). https://doi.org/10.1145/2799979.2799986
22. Google Safe Browsing API, https://developers.google.com/safe-browsing/v4/lookup-api, Retrived on 12 December, 2017.
23. Gujral, H., Mittal, S., & Sharma, A. (2019). *Wireless Personal Communications*. https://doi.org/10.1007/s11277-019-06135-1
24. Han, S. H., Kim, M. S., Ju, H. T., & Hong, J. W. K. (2002). The architecture of NG-MON: A passive network monitoring system for high-speed IP networks. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (vol. 2506, pp. 16–27). https://doi.org/10.1007/3-540-36110-3_5
25. Hughes, D. A., Bheemarajaiah, S., Ennis, D. J., Merwin, D. H., Muralt, R., Ozduygu, O., & Singh, P. K. (2018). U.S. Patent No. 10,164,861. Washington, DC: U.S. Patent and Trademark Office.
26. Igure, V. M., & Williams, R. D. (2008). Taxonomies of attacks and vulnerabilities in computer systems. *IEEE Communications Surveys and Tutorials, 10*(1), 6–19. https://doi.org/10.1109/COMST.2008.4483667
27. Kothari, S., Parmar, H., Das, E., Panda, N., Ahmed, A., Marchang, J. (2011). Host based intrusion detection system. In *International conference on mechanical engineering and technology (ICMET-London 2011)* (pp. 4), Garry Lee, ASME, New York.
28. Kumar, S., Viinikainen, A., & Hamalainen, T. (2016). Machine learning classification model for network based intrusion detection system. In *2016 11th international conference for internet technology and secured transactions (ICITST)* (pp. 242–249). IEEE.
29. Liu, M., Xue, Z., Xu, X., Zhong, C., & Chen, J. (2018). Host-based intrusion detection system with system calls: Review and future trends. *ACM Computing Surveys (CSUR), 51*(5), 1–36.
30. Lyon, G. (2008). *Nmap network scanning: Official Nmap project guide to network discovery and security scanning* (Book)
31. Mohammad, R. M., Thabtah, F., & McCluskey, L. (2015). Tutorial and critical analysis of phishing websites methods. *Computer Science Review, 17*, 1–24. https://doi.org/10.1016/j.cosrev.2015.04.001
32. Mohammad, R. M., Thabtah, F., & McCluskey, L. (2015). Phishing websites features. Unpublished. http://eprints.hud.ac.uk/24330/6/RamiPhishing_Websites_Features.pdf.
33. Mohammad, R. M., McCluskey, L., Thabtah, F. (2015). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/phishing+websites]. Irvine, CA: University of California, School of Information and Computer Science.
34. Newtein (Harshit Gujral). GitHub Repository. https://github.com/newtein/network_monitoring.
35. NVD, National Vulnerability Database, NVD-Data Feeds, https://nvd.nist.gov/vuln/data-feeds, Retrieved on 12 December, 2017.
36. Parno, B., Kuo, C., &Perrig, A. (2006). Phoolproof Phishing Prevention (pp. 1–19). https://doi.org/10.1007/11889663_1
37. Pedregosa, et al. (2011). Scikit-learn: Machine learning in python. *JMLR, 12*, 2825–2830.
38. Probin, R. J., & Crisp, M. L. (2017). U.S. Patent No. 9,614,860. Washington, DC: U.S. Patent and Trademark Office.
39. Qu, G., Rudraraju, J., &Modukuri, R. (2002). A Framework for Network Vulnerability Analysis. Communications, 2(4), 1–6. Retrieved from http://www.actapress.com/PaperInfo.aspx?PaperID=24366&reason=500.
40. Rajalingam, M., Ali Alomari, S., & Sumari, P. (2012). Prevention of phishing attacks based on discriminative key point features of WebPages. *Putra Sumari International Journal of Computer Science and Security, 61*, 2012–2021.
41. Rodas, O., & To, M. A. (2015). A study on network security monitoring for the hybrid classification-based intrusion prevention systems. *International Journal of Space-Based and Situated Computing, 5*(2), 115–125.
42. Shu, G., & Lee, D. (2006). Network protocol system fingerprinting-a formal approach. In *IEEE International Conference on Computer Communications*.
43. Shu, G., & Lee, D. (2005). Defending against internet host fingerprintingtoward an outermost barrier of cyberspace security. Working Together: Research & Development (R&D) Partnerships in Homeland Security.

44. Staniford-Chen, S., Tung, B., Porras, P., Kahn, C., Schnackenberg, D., Feiertag, R., &Stillma, M. (1998). The common intrusion detection framework-data formats.
45. Thorvaldsen, A., & D'Ambra, D. (2010). Panorama. Software. https://www.panorama9.com/.
46. U.S. Attorney's Office (2010), "Iceman" Computer Hacker Receives 13-Year Prison Sentence, https://archives.fbi.gov/archives/pittsburgh/press-releases/2010/pt021210b.htm, Retrieved on 29 March, 2018.
47. Van, N. T., & Thinh, T. N. (2017, July). An anomaly-based network intrusion detection system using deep learning. In *2017 international conference on system science and engineering (ICSSE)* (pp. 210–214). IEEE.
48. Vladishev, A. (2001). Zabbix. Software. https://www.zabbix.com/
49. Vokorokos, L., & BaláŽ, A. (2010). Host-based intrusion detection system. In *INES 2010-14th international conference on intelligent engineering systems, proceedings* (pp. 43–47). https://doi.org/10.1109/INES.2010.5483815.
50. Vu, H. L., Khaw, K. K., & Chen, T. Y. (2014). A new approach for network vulnerability analysis. *The Computer Journal, 58*(4), 878–891. https://doi.org/10.1093/comjnl/bxt149
51. Walker, R. C., Amrutur, B., Mottishaw, P., Joiner, C. S., Chesler, L. A., & Hardcastle, I. (2005). U.S. Patent No. 6,975,617. Washington, DC: U.S. Patent and Trademark Office.
52. Wang, L., & Jones, R. (2017). Big data analytics for network intrusion detection: A survey. *International Journal of Networks and Communications, 7*(1), 24–31.
53. Yonce, D., & Yonce, D. (1999). Network Performance Monitor. Software. https://www.solarwinds.com/network-performance-monitor.
54. Zilberman, P., Puzis, R., & Elovici, Y. (2017). On network footprint of traffic inspection and filtering at global scrubbing centers. *IEEE Transactions on Dependable and Secure Computing, 14*(5), 521–534.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Harshit Gujral is a Ph.D. student in the Department of Computer Science at the University of Toronto. With a background in Software Engineering and Cybersecurity, his current research interests involve building new or improving existing AI models for studying the feasibility of environmental public policies for addressing climate change. During the coronavirus pandemic, he has studied the health outcomes of exposure to air pollutants. He has pursued a Bachelors in Information Technology with Honours from Jaypee Institute of Technology, Noida, India, visited the Department of International Relations at Koç University, Istanbul, Turkey, and worked as a Data Scientist before beginning his doctoral journey.

Abhinav Sharma is currently working as a Software Engineer 2 in PayPal. He has completed his bachelor's degree in Information Technology from Jaypee Institute of Information Technology, Noida, India. His area of research interest includes Algorithms, Data Analytics, Computer Network, Statistics, and Information Technology.

Pulkit Jain is currently working as an Engineer at Samsung R&D. He has pursued his Bachelor of Technology in Information Technology from Jaypee Institute of Information Technology, Noida, India. He has been part of many Database Management and Web development Projects in brainstorming work environment of start-ups. His research interests are Network Security, Network Communication, Agile Software Development, and cryptography

Shriya Juneja is a developer by passion, a tech savant, a problem solver, and a nature enthusiast, She has pursued her Bachelor of Technology in Information Technology from Jaypee Institute of Information Technology, Noida, India. Her primary strength involves the art of transforming theoretical concepts into pragmatic ideas of real-world applications. Shriya's focus areas include web performance, security, and scale.

Dr. Sangeeta Mittal is serving as faculty in the Department of Computer Science & Engineering and IT at Jaypee Institute of Information Technology, Noida. She has a rich academic experience of 18 years. She is a member of the main computing professional body, Association for Computing Machinery (ACM), and a life member of the Computer Society of India (CSI). She has research interests in the areas of cyber security and computer networks. She has published papers in international journals and conferences of repute in areas of Wireless sensor networks, context-aware sensing, IoT, Fog computing, applications of machine learning and AI, network communication protocols, mobile security, and cryptography. She has also contributed book chapters to books published by International publishers in the area of computer networks and security. She serves on the review panel of many reputed and peer-reviewed journals like Parallel and Distributing Computing, PLoS One, Soft Computing, and many others.