IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 10 May 2024, accepted 3 June 2024, date of publication 5 June 2024, date of current version 14 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3410111

APPLIED RESEARCH

SUTMS: Designing a Unified Threat Management System for Home Networks

ASIF SIDDIQUI[1], BHASKAR P. RIMAL[2], (Senior Member, IEEE), MARTIN REISSLEIN[3], (Fellow, IEEE), DEEPAK GC[4], (Senior Member, IEEE), AND YONG WANG[1]

[1]The Beacon College of Computer and Cyber Sciences, Dakota State University, Madison, SD 57042, USA
[2]Department of Computer Science, University of Idaho, Moscow, ID 83844, USA
[3]School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287, USA
[4]School of Computer Science and Mathematics, Kingston University London, KT1 2EE Kingston upon Thames, U.K.

Corresponding author: Martin Reisslein (reisslein@asu.edu)

This work was supported in part by Dr. Rimal's Startup Grant with the University of Idaho, Moscow, ID, USA and in part by Dr. Deepak GC's KU First Grant at the Kingston University London, U.K.

ABSTRACT The cultural shift of work from on-premises to remote home offices allows hackers to access corporate data by compromising devices attached to home-based broadband routers. Currently, home devices are growing at an exponential rate, e.g., in the form of smartphones, Internet of Things (IoT) devices, and corporate laptops. Presently, there is a lack of cost-effective, practical, and lightweight Unified Threat Management (UTM) systems designed to protect home users from the wide range of existing cyber threats. This applied research article introduces a sophisticated Smart UTM System (SUTMS) that is designed to tackle the contemporary security issues encountered in home networks while running on a Raspberry Pi single-board computer. The proposed integrated SUTMS design consists of flow detection, intrusion detection, and firewall core engines, as well as optional routing and log collection engines. The flow detection engine in SUTMS discovers anomalies and detects the active protocols, which are a basis for signature optimization in the SUTMS intrusion detection system (IDS). By ingesting Indicator of Compromise (IoC) feeds, the SUTMS firewall engine provides dynamic anti-bot protection. Evaluations indicate that SUTMS with IDS signature optimization can provide 99% accuracy with approx. 55% memory utilization reduction compared to traditional signature-based IDS.

INDEX TERMS Anti-bot protection, flow detection, Internet of Things (IoT), intrusion prevention system (IPS), privacy, Raspberry Pi, security, unified threat management (UTM), vulnerabilities.

I. INTRODUCTION

A. MOTIVATION

Home broadband routers are primarily designed for Internet connectivity without a focus on built-in security. Cyber-attacks, remote work, and cloud computing have made home networks more vulnerable than ever [1]. Barracuda Sentinel observed an increase of 667% in phishing attacks in 2020 [2]. The use of home computers for work is another reason for compromises, as personnel computers often lack proper security controls [3]. With work-from-home arrangements, home networks have essentially become extensions of critical corporate networks and services. Insufficient network security measures, such as the absence of firewalls and threat management systems, coupled with the heightened vulnerability posed by Internet-connected smart devices, underscore the necessity for a robust security solution tailored to offer corporate-grade protection for small home networks [4].

This applied research article introduces the Smart Unified Threat Management System (SUTMS), designed to operate as a Unified Threat Management (UTM) system for small office and home networks. The sole purpose of SUTMS is to protect against advanced attacks (malicious payload injection) in a multi-layered approach. SUTMS inspects traffic starting from Open Systems Interconnection (OSI)

The associate editor coordinating the review of this manuscript and approving it for publication was Chuan Heng Foh.

Layer 3 (the network layer in the Internet Protocol (IP)) and extends to the higher protocol layers.

Conventional network security devices for home networks are limited to Layer-3 and Layer-4 inspection due to resource constraints, as inspections beyond Layer 4 demand extensive computational resources. Enhanced features, such as intrusion detection and anti-bot protection, require additional Central Processing Unit (CPU) cycles and paid signature subscriptions.

B. OVERVIEW OF SUTMS DESIGN

SUTMS introduces advanced security features with real-time detection and blocking of malicious traffic. SUTMS can examine all IP-based communication, including the communication of Internet of Things (IoT) devices and mobile computing devices. SUTMS consists of three core engines (flow detection, intrusion detection, and firewall), an optional routing engine, and an optional log collector that collaborate to optimize system resources and inspect traffic to detect any malicious payloads.

The flow detection engine serves as the SUTMS backbone. Traffic identification, IDS signature optimization, and anomaly detection are the key capabilities that are facilitated by flow detection. The intrusion detection engine analyzes traffic for known vulnerabilities and defined signatures. The signature database gets updated regularly as new vulnerabilities arise. The intrusion detection engine can be set to detect or prevent, depending upon the mode of device operation, i.e., intrusion detection system (IDS) or intrusion prevention system (IPS). The intrusion detection engine inspects traffic solely on the basis of signature matches. The firewall engine allows or blocks traffic based on IP addresses and ports. The SUTMS firewall dynamically blocks malicious IP addresses by ingesting Indicator of Compromise (IoC) feeds and adding them to the block list, enabling proactive prevention of malicious activity from bad actors (bots).

The routing engine, which is optional in SUTMS, routes the traffic to the desired destination. Routing can be outsourced to the existing infrastructure components, such as the broadband router or access point in a home network. WiFi, Domain Name Service (DNS) forwarding, and Dynamic Host Configuration Protocol (DHCP) services can also be configured on SUTMS. Running multiple security services on a single device with limited resources is challenging. However, using correlation services, such as flow detection, in SUTMS significantly reduces the burden of CPU-intense processes, e.g., IDS/IPS, as evaluated in Section V.

SUTMS can easily be implemented in existing home networks without any major changes. The various SUTMS deployment scenarios are presented in Section. III-D. There is also room for expanding scanning to IoT networks, such as ZigBee, Z-wave, and Bluetooth low-energy (BLE) networks. However, these expanded scanning services may require the procurement of additional hardware and software.

C. CONTRIBUTIONS AND STRUCTURE

The key contributions and the section structure of this applied research article are summarized as follows:

- Designed the SUTMS architecture for a lightweight UTM implementation, capable of delivering advanced threat detection capabilities while utilizing minimal computational power. Various UTM deployment strategies have been identified, proposed, and examined for incorporating SUTMS in different home network setups (Section III).
- Systematic steps are undertaken in SUTMS to ensure the collaboration of the various UTM modules, working together to achieve the desired efficiency (Section IV).
- Enhanced the traditional firewall service by transitioning from manual traffic blocking to dynamically preventing Command and Control (C&C) communication. This is achieved in SUTMS through the automated ingestion of IoC feeds via STIX/TAXII. Accordingly, the IPtables firewall establishes automated access control lists, allowing for dynamic adjustments based on the IoC feeds (Section IV-C5).
- Optimizing IDS signatures is achieved by channeling flows from the NTOP flow detection into the Suricata IDS. This not only streamlines IDS configuration management and reduces false positives, but also substantially reduces the utilization of computing resources (Section IV-B).
- Thorough evaluation and validation of SUTMS are conducted to ensure its performance aligns with real-world scenarios. Both the detection accuracy of the UTM engines and their utilization of computing system resources are evaluated (Section V).

We make the SUTMS code and configurations publicly available [5].

II. RELATED WORK

Karahan and Berat [6] designed a lightweight firewall on a Raspberry Pi that can inspect ports, user groups, and malicious traffic. The lightweight firewall has an easy-to-use interface and compact design, which make it suitable for home users. De la Cruz et al. [7] introduce a US$209 IDS/IPS that can be managed remotely. A Snort engine was utilized for attack detection according to signature matching. Similar studies on intrusion detection in home networks have recently been reported in [8] and [9].

Visoottiviseth et al. [10] designed Protection Internet of Things via IDS (PITI) specifically for IoT networks [11], [12], [13], [14]. PITI combines Snort detection with anomaly detection on Raspberry Pi hardware. Alarms and sounds are generated as anomalies are detected by PITI. Tirumala et al. [15] evaluated the performance of defensive controls on a Raspberry Pi, with a focus on volumetric traffic testing across network interfaces.

The propagation of malware through malicious URLs circumvents traditional security controls. Fauzi and Abdullah [16] introduced a Raspberry Pi-based phishing detection technique called Netbits. Netbits was able to identify malicious content using traffic flows. The Netbits study can be useful in mitigating botnet communication.

Denial of Service (DoS) attacks can bring down wireless networks. Kismet IDS [17] can detect 10 DoS attacks with an average attack detection time of 3.42 seconds. Integrating artificial intelligence algorithms in IDS improves accuracy [18], [19]. Siddharthan and Thangavel [20] proposed an IDS algorithm for IoT devices using ensemble learning (EL). The algorithm achieved 99.99% accuracy with only a few feature selection complications [20]. Training on datasets in machine learning is a crucial step in attaining accuracy. Training fewer and more relevant features is key to designing an efficient IDS. The IoT-based botnet detection engine BotStop [21] uses only seven features from network packets and can achieve greater than 99.99% accuracy.

The prior home network security research studies have been focused on the individual IDS, firewall, *or* anti-bot functionalities on low-resource devices, such as a Raspberry Pi. That is, the existing studies were limited to individual UTM functionalities (components) [4]. In contrast, we design a single integrated device that is capable of IDS, firewall, *and* anti-bot protection. To the best of our knowledge, the proposed SUTMS is the *first multi-dimensional UTM system* that achieves advanced inspection (encompassing IDS, firewall, *and* anti-bot protection) while efficiently utilizing computational resources (to operate on a Raspberry Pi 4 single-board computer).

Three key features distinguish SUTMS as a smart UTM: First, SUTMS employs innovative techniques to identify applications through flow detection. Second, SUTMS utilizes the identified applications to optimize resource-intensive processes, such as IDS. In contrast, traditional IDS systems often add signatures to their databases without awareness of the applications in use. Third, SUTMS establishes a baseline for applications, traffic, and services, enabling anomaly detection without the need for complex machine learning algorithms.

III. OVERVIEW OF SUTMS ARCHITECTURE

A. SUTMS ENGINES (MODULES)

1) OVERVIEW

SUTMS consists of three core engines (modules) for flow detection, intrusion detection, and firewall, as well as an optional routing module and an optional log collector. Fig. 1 shows a representation of the SUTMS modules and their interactions to develop the final processing decision. The required modules include the flow detection engine, intrusion detection engine, and firewall engine. Since SUTMS is a security device, routing is categorized as optional. Dynamic routing protocols are currently unsupported, as SUTMS is geared towards small home networks.

In SUTMS, the flow detection engine correlates data with the intrusion detection engine to improve signature efficacy and reduce CPU and memory consumption. The collaboration between the flow detection engine and the intrusion detection engine results in signature optimization. The resulting logs can be fed to collectors for reporting, analytics, and troubleshooting.

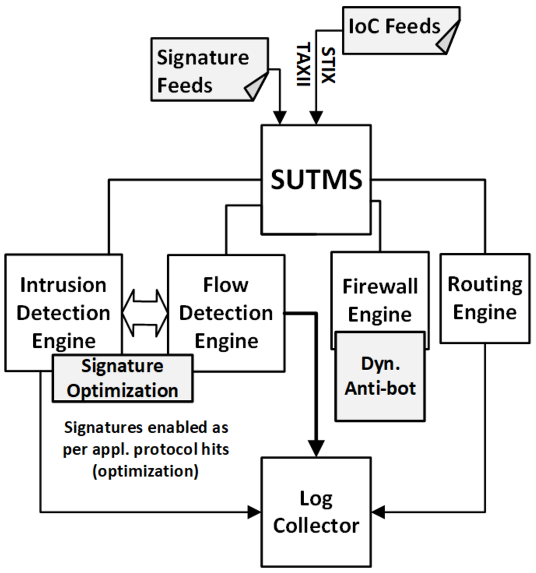


FIGURE 1. Design of SUTMS with core engines for flow detection, intrusion detection, and firewall; routing engine and log collector are optional. Flow detection (aside from detecting anomalies) optimizes the signatures for intrusion detection to reduce computational complexity, enabling SUTMS to run on a Raspberry Pi 4. With the Indicator of Compromise (IoC) feeds, the firewall provides dynamic anti-bot protection.

In summary, we note that the flow detection, intrusion detection, and firewall engines in SUTMS can flexibly employ any of the existing flow detection, intrusion detection, and firewall tools. In our evaluations of SUTMS, we employ a recommended SUTMS configuration with NTOP as the flow detection engine, Suricata as the intrusion detection engine, and IPtables combined with automated IoC feeds via STIX/TAXII as the firewall engine. These recommended choices are based on the reasoning described in the remainder of this subsection.

2) FLOW DETECTION WITH NTOP

Flow detection can be achieved through protocols such as NetFlow, IPFIX, and NTOP. Importantly, home routers and switches frequently lack support for NetFlow and IPFIX, requiring a collector to process flow information, which adds complexities to the implementation. In contrast, NTOP functions as a service that can be enabled on a Linux machine to collect flows and seamlessly build use cases. Therefore, we select NTOP as a flow detector for SUTMS. We emphasize that both Suricata and Snort are signature-based network security solutions designed to detect or prevent known malicious payloads. In contrast, NetFlow, IPFIX, and NTOP are utilized for flow detection. Flow detection facilitates signature optimization for intrusion detection (via Suricata) in SUTMS. Moreover, NTOP is exploited for anomaly detection in SUTMS.

It is crucial to recognize that malicious payloads may attempt lateral movement within a network, moving from one machine to another. Security controls within Local Area Networks (LANs) are often less stringent than those at the perimeter, allowing such lateral movements to potentially go unnoticed. Flow detection plays a vital role in monitoring such lateral movements and raising alarms for any anomalies.

3) INTRUSION DETECTION WITH SURICATA

Snort, Suricata, and Zeek stand out as three viable options for the intrusion detection engine. Generally, when a malicious payload is detected, an intrusion detection engine has the option to only detect/alert or to block/prevent. In the IDS mode, signatures that get matched with a malicious payload will only be detected/alerted; whereas, in the IPS mode, the same signature match will have an action of prevent/block. As a monitoring solution, Zeek excels in detecting signatures within home networks (and has some traffic analysis capabilities as well). However, Suricata and Snort go beyond mere detection by providing capabilities for actively blocking malicious traffic. Snort operates as a single-threaded system, while Suricata is multi-threaded. This makes Suricata the preferred choice for an intrusion module within the context of SUTMS.

4) IPtables FIREWALL WITH IoC FEEDS FOR ANTI-BOT PROTECTION

There are various methods for ingesting IoC feeds, with some of the most prevalent ones being Open Threat Exchange (OTX) [22], Malware Information Sharing Platform & Threat Sharing (MISP) [23], Cyber Observable eXpression (CybOX) [24], and Structured Threat Information eXpression/Trusted Automated eXchange of Indicator Information (STIX/TAXII) [25], [26]. While OTX, MISP, CybOX, and STIX/TAXII are comparable solutions, STIX/TAXII holds an advantage, particularly in converting intelligence into actionable controls, such as firewall access control lists. This is attributed to its standardized format, widespread adoption, simple configuration, and ease of transport via the TAXII protocol. Therefore, we select STIX/TAXII for the IoC ingestion for SUTMS, taking into consideration the home user's limited technical expertise.

Automated IoC feeds via STIX/TAXII turn the IPtables firewall into an anti-bot engine. More specifically, the SUTMS firewall module serves a dual purpose by providing both traditional blocking and dynamic blocking of Indicators of Compromise (IoCs). The inclusion of IoC blocking introduces an anti-bot service. This anti-bot service utilizes the STIX/TAXII structured formatting to ingest IoCs. Once the IoCs are ingested, the anti-bot service dynamically generates access control lists for the firewall module to enforce Layer 3 and Layer 4 inspection. We employ the IPtables firewall to implement this dynamic access control list. Simultaneously, the firewall engine handles the formatting, conversion, and de-duplication of the received IoCs.

B. TYPES OF TRAFFIC FLOWS IN SUTMS

A traffic flow is either an outbound (egress) traffic flow or an inbound (ingress) traffic flow. Fig. 2a depicts the egress traffic from the Local Area Network (LAN) to the Internet, and Fig. 2b depicts the ingress traffic from the Internet to the LAN.

1) EGRESS TRAFFIC

An egress traffic flow is processed, i.e., traffic detection and inspection are carried out, with the following two sequential steps:

- Traffic originates from the LAN, which includes Internet Protocol (IP) traffic from home computing devices, such as laptops, desktops, smart devices, and Internet of Things (IoT) devices.
- The flow detection engine intercepts all the traffic passing through it and determines key traffic attributes, such as the Layer 4 (TCP/UDP) protocol attributes, the application layer protocol attributes, the conversation duration, and the IP addresses, which are then used for signature optimization. Subsequently, this traffic attribute information is forwarded to the IDS, which utilizes this traffic information to refine signatures and selectively activate rule sets based on relevant protocols.

2) INGRESS TRAFFIC

Fig. 2b illustrates the flow of ingress traffic to SUTMS, which inspects ingress traffic as follows:

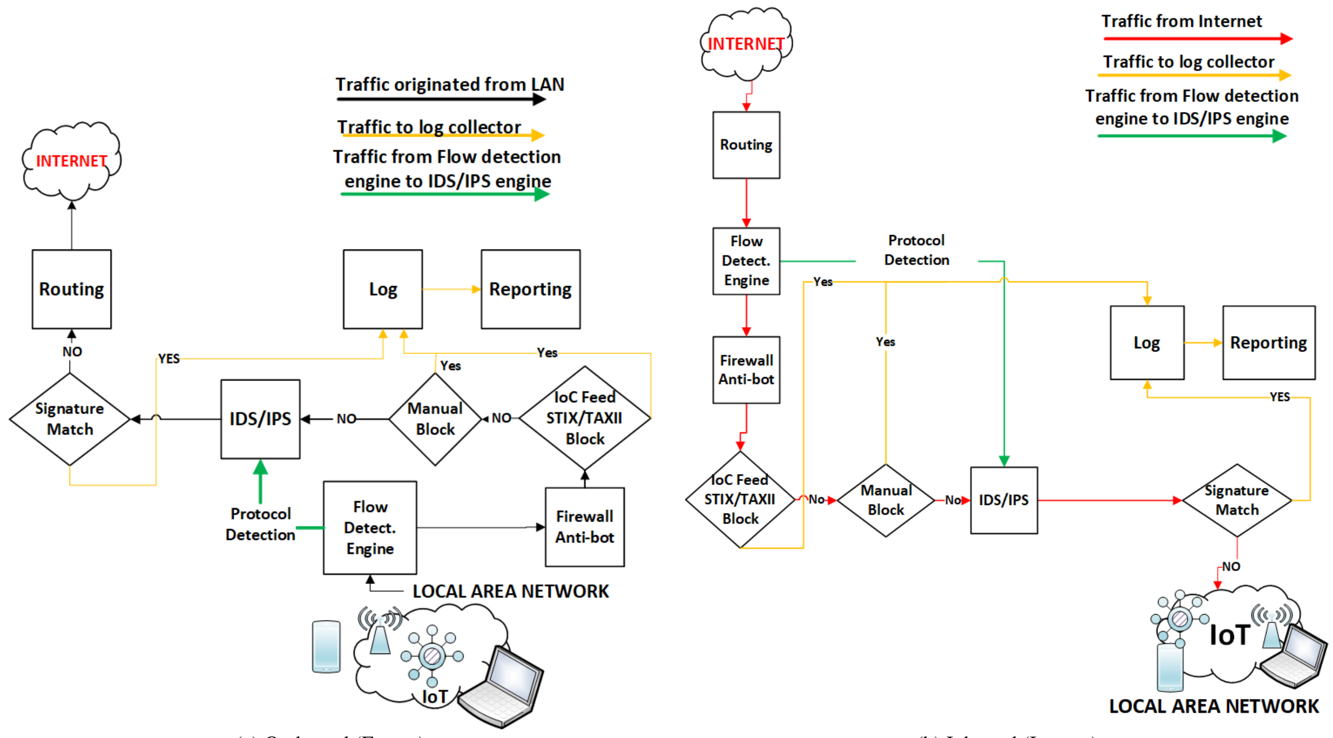
- Traffic from the Internet (ingress) is initially processed by the routing engine, which directs the traffic to the flow detection engine. The flow detection engine then communicates with the IDS engine, updating the IDS engine with the pertinent signatures.
- Subsequently, the traffic is processed by the firewall engine. If no IoC or manual matches are found, then the traffic is cleared for processing by the IDS engine.
- The IDS engine examines traffic based on the enabled signatures. If no matching signatures are identified, then the traffic is deemed acceptable to proceed to its final destination within the LAN.

3) LOGGING TRAFFIC

Apart from managing egress and ingress traffic, there is also traffic directed towards log collectors. This traffic to log collectors can either be stored locally on SUTMS, or can be routed to an external third-party log collector. External log collection is an optional component. Additionally, logs can be integrated with security log correlation tools, such as a Security Incident Event Management (SIEM) system. These logs serve various purposes, including reporting, trending, and data analytics.

C. OVERVIEW OF SUTMS INSTANTIATION

The core function of SUTMS is to provide advanced UTM inspection capabilities for home networks while minimizing



(a) Outbound (Egress) (b) Inbound (Ingress)

FIGURE 2. **Flow of outbound and inbound traffic via SUTMS.**

the utilized computational resources, reducing costs, ensuring easy deployment, and optimizing services. While SUTMS is adaptable to a wide range of hardware and software, its design for home networks carefully considers the viability and effectiveness of the various options. We outline the design choices below.

1) HARDWARE

We implemented and evaluated the SUTMS design on a Raspberry Pi 4 with 8 GB of RAM, 32 Gigabyte (GB) of Secure Digital (SD) memory storage, and a Quad-core Cortex-A72 1.5 GHz processor [27]. The Raspberry Pi 4 has multiple connectivity interfaces, including WiFi, Gigabit Ethernet, and Bluetooth, which allow the Raspberry Pi 4 to act as a communication gateway for a variety of computing devices, including IoT devices.

2) SOFTWARE

We run the Ubuntu 22.04 LTS operating system on the Raspberry Pi 4 and the following software modules: a) GNU ntopng as flow detection engine for protocol and anomaly detection, b) Suricata version 6.0.4 as intrusion detection (IDS) engine for real-time scanning and detection/prevention of malicious traffic, c) Linux IPtables as firewall engine, d) STIX/TAXII feeds from Anomali for integrating anti-bot capabilities into the firewall engine, e) Routing and access point services, g) Apriori algorithm (optional component) that can be applied to IDS outputs (alerts) for fine-tuning and noise elimination, and h) Elasticsearch and Logstash for log aggregation (optional component).

D. DEPLOYMENT STRATEGIES

Integrating SUTMS into a Raspberry Pi enables support for a diverse range of communication protocols and deployment scenarios. By combining Ethernet and WiFi capabilities, along with the addition of other modules, SUTMS running on a Raspberry Pi extends support to IoT devices and a variety of smart home devices. SUTMS can be deployed based on module functionality. For example, if the sole requirement is to run the IDS in detection mode, then the SUTMS device can be deployed as an out-of-band solution. On the other hand, if active blocking is necessary, an inline deployment option is also available.

1) INLINE DEPLOYMENT WITH SUTMS WiFi ACCESS POINT

With inline deployment, all traffic is forced to traverse the SUTMS device. As illustrated in Fig. 3(a), the inline deployment with SUTMS WiFi access eliminates the need for an access point, as SUTMS is responsible for WiFi access along with the UTM inspection services. Home networks where the Internet Service Provider (ISP) handoff is not Ethernet require an external modem; currently, the Raspberry Pi does not have a built-in modem.

2) INLINE DEPLOYMENT WITH EXISTING WiFi ACCESS POINT

This is the most common form of deployment as most of the home networks already have a WiFi access point.

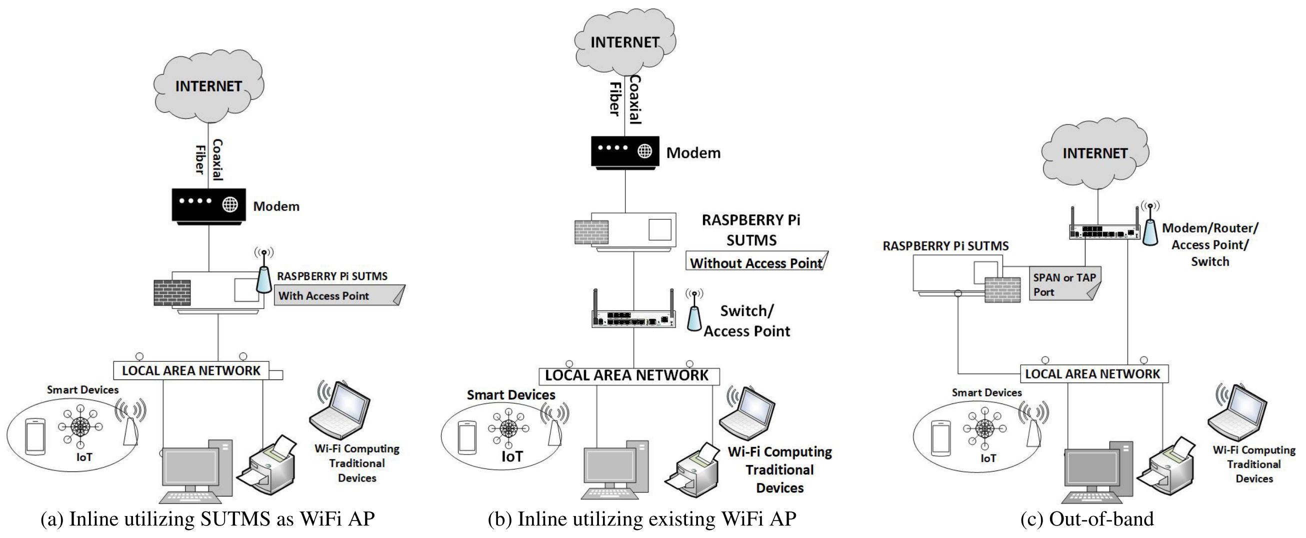


FIGURE 3. Deployment options of SUTMS.

We can leverage the existing network infrastructure and install SUTMS between the ISP modem and the WiFi Access point, see Fig. 3(b). This deployment mode requires minimum configuration changes on endpoints, e.g., the WiFi services remain untouched.

3) OUT-OF-BAND SUTMS DEPLOYMENT

Out-of-band implementation of SUTMS is also possible. However, the protection capabilities are significantly reduced. Specifically, intrusions are still detected, and flow traffic is monitored; however, SUTMS will not be able to block any traffic. Out-of-band implementations are usually done either via a SPAN port or a physical TAP must be installed, see Fig. 3c. Port SPAN, a term coined by Cisco, refers to the mirroring of network traffic [28]. Essentially, traffic from one port is replicated and sent to another port. This other port enables passive monitoring for analysis, IDS, or monitoring tools, thus facilitating out-of-band analysis. Similarly, a Test Access Port (TAP) serves a comparable function, requiring the installation of physical hardware, i.e., the TAP device [28]. In contrast, SPAN is achieved through switch configuration.

IV. DETAILS OF SUTMS ENGINES

A. FLOW DETECTION ENGINE

The flow detection engine fingerprints applications based on observed flow and protocol characteristics. For instance, the HTTP protocol exhibits specific Layer 3, Layer 4, and application traits. Applications are identified through an application detection service, while anomalies in flows can be pinpointed as deviations from the expected behavior. SUTMS takes advantage of the open-source probing service NTOP. NTOP has two main purposes. First, NTOP is used to identify abnormal traffic patterns; second, the identified protocols are utilized to optimize Suricata signatures.

TABLE 1. NTOP built-in security alerts, e.g., relating to the Transport Layer Security (TLS) and Domain Name System (DNS) protocols [29].

Alert Key	Alert Key String	Alert Name
23	`alert_tls_certificate_mismatch`	TLS Certificate Mismatch
24	`alert_tls_old_protocol_version`	Obsolete TLS Version
25	`alert_tls_unsafe_ciphers`	Unsafe TLS Ciphers
6	`alert_dns_data_exfiltration`	DNS Data Exfiltration
64	`ndpi_dns_large_packet`	DNS Packet Larger than 512 bytes
58	`alert_lateral_movement`	Lateral Movement
62	`ndpi_clear_text_credentials`	Clear-Text Credentials

1) APPLICATION DETECTION SERVICE

NTOP can be configured to sniff flows on the internal and external interfaces; specifically, *eth0* will be programmed for capturing internal ingress traffic, and *wlan0* will give flows for internet-bound egress traffic.

2) ALERTS ON ANOMALIES

NTOP application detection capabilities can be used not only for tuning IDS signatures but also for identifying anomalies. Table 1 [29] lists default security alerts that can be triggered based on any observed anomalies. Alert keys 23, 24, and 25 inspect encrypted traffic without decryption. Transport Layer Security (TLS) certificate mismatches, legacy versions, and insecure ciphers can be an indication of vulnerable applications and possible anomalies.

Alerts 6 and 64 deal with payload abnormalities of DNS packets. Hackers tend to use DNS for data exfiltration [30] and command and control (C&C) communication. Custom alerts can also be created based on the flows observed by SUTMS. Anomalies can arise from deviations in traffic patterns from the norm. For instance, alerts can be issued for the appearance of new flows that have not been observed within the past 30 days or when the traffic rate exceeds normal levels.

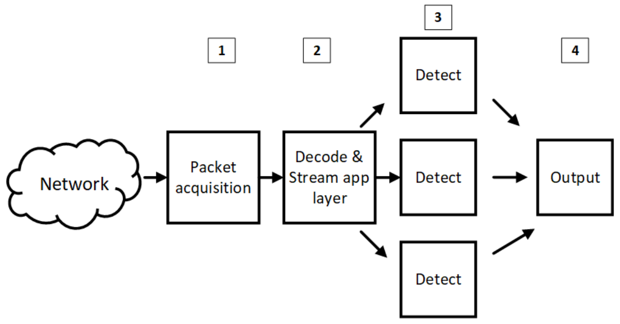


FIGURE 4. Suricata 6.0.4 architecture with four thread modules [34].

B. SUTMS IDS ENGINE TECHNICAL DETAILS

1) SURICATA OVERVIEW

We selected Suricata version 6.0.4 as intrusion detection engine. The multi-threaded architecture of Suricata and its integration with other engines, e.g., Logstash [31] and Kibana [32], make it a better choice over Snort. Suricata was created to overcome some of the drawbacks of Snort. Snort is single-threaded, while Suricata is multi-threaded [33]. Moreover, Suricata takes advantage of multiple CPUs and can distribute processing across multiple threads.

Suricata has four thread modules, see Fig. 4 [34]. The packet acquisition thread captures packets using Libpcap similar to Snort [35]. Packets are decoded and classified, and the flow state is established by the decode & stream app layer thread. Suricata can parse protocols, such as HTTP, at the application layer using a stateful parser and can remove malicious content [36], [37]. The detection thread runs multiple instances of threads concurrently, rules are scanned, and alerts will be sent via the output threat for any possible matches. The Suricata rule format is very similar to Snort, except for the ability to specify application layer protocols, such as HTTP or DNS.

2) SURICATA SUTMS IMPLEMENTATION

The Raspberry Pi 4 has a single-core architecture, which cannot utilize the benefits of multi-threading. Suricata rules are downloaded automatically by running the command `sudo suricata-update`. Thereby, the signature download process automatically detects the version and writes the rules into the `var/lib/suricata/rules/suricata.rules` file. Another notable fact is that by default (as of March 2024), 37075 rules will get enabled out of 48388. Scanning 37075 rules will require intense processing power, which will significantly degrade the performance of UTM devices. To mitigate any negative impact on performance, SUTMS takes advantage of the NTOP flow detection to enable only the signatures for the protocols that are used in an environment.

3) IDS SIGNATURE OPTIMIZATION BASED ON FLOW DETECTION

IDS inspects traffic based on numerous signatures which characterize patterns or behaviors indicative of malicious activity within protocols or applications. Signatures are regularly created and updated by vendors and the open-source community to address new vulnerabilities, exploits, and bugs as they are discovered. Fig. 1 illustrates how these signatures are ingested into SUTMS via signature feeds, allowing the SUTMS IDS engine to automatically update its signature database.

The more signatures are enabled, the more computational resources (e.g., CPU, memory) are consumed by the IDS engine. The flow detection engine has the capability to identify the protocols and applications that are utilized within a given home network. Once detected, the flow detection engine can communicate with the IDS engine to selectively enable signatures that are relevant for the identified protocols, e.g., HTTP and FTP, while leaving other protocols, e.g., Telnet and the Trivial File Transfer Protocol (TFTP), disabled.

By default, Suricata enables signatures based on numerous factors, including severity, performance impact, and false positive rate. Our goal is to create a ready-to-deploy IDS that minimizes the need for user intervention and is readily deployable in home networks without dedicated network administration support. However, users have the flexibility to override these actions by modifying Suricata configurations. Many of these signatures may not be pertinent to a given home network. For instance, signatures related to dynamic routing protocols, Telnet, and TFTP may be unnecessarily enabled if these protocols are not used in the network. Indeed, data derived from the NTOP flow detection, see Sec. V-B, revealed that only certain protocols are typically used in home networks. Consequently, without flow detection, the IDS expends computational resources to process numerous unused signatures. On the other hand, with flow detection, the signatures for unused signatures can be disabled. This signature optimization significantly reduces the computational resources required for the SUTMS IDS engine.

In SUTMS, we have created a novel script that automatically disables the unused signatures by updating the Suricata rules file according to the detected protocols. Specifically, to automate the disabling of unwanted signatures, we created a script that executes a two-step process: First, the script edits the Suricata `disable.conf` file to set all newly downloaded (ingested) rules to the disable state (by commenting their signatures). Second, the script employs regular expressions (Regex) to activate (enable) rules that relate to protocols that are in use in the home network in accordance with the protocol data retrieved from the flow detection engine. In particular, the script edits the `suricata.rules` file in the `/etc/suricata` directory to set the rules relating to the used protocols to the enable state (by un-commenting their signatures). We note that the `disable.conf` file does not exist by default in Suricata; rather, we changed the default behavior of Suricata to ensure that newly downloaded rules are disabled by editing the `disable.conf` file. In particular, the `disable.conf` file is a separate file from the `suricata.rules` file; the `disable.conf` file modifies the `suricata.rules` file upon the download of new rules (signatures). Whereby, we modified the Suricata default behavior to disable all newly downloaded rules in a first step, and then we selectively enable only the rules that are pertinent to the used protocols in a second step, Consequently, in SUTMS, relatively few signatures, namely only the relevant signatures for the given home network, are enabled, resulting in low CPU and memory usage by the IDS.

Also, we disable rule matching with the Perl Compatible Regular Expression (`pcre`) option. Matching rules with the `pcre` option is a resource-intensive process that can overload the UTM device. However, disabling the `pcre` rule matching option may reduce the IDS inspection capabilities. Nevertheless, in our evaluations in Section V with the common CICIDS2017 dataset, events that could only be detected with `pcre` rule matching were not triggered. There are alternative matching options, such as the `content` keyword matching, which is capable of performing similar inspection functions. We employ these alternatives as part of the SUTMS IDS inspection, see [5] and [38], to avoid excessive computational burdens while still achieving high accuracy.

Overall, these signature optimizations, which we achieve by strategically disabling signatures, reduce the number of Suricata rules from 37075 to 5565 (as of March 2024, for the settings in [5]).

C. SUTMS FIREWALL ENGINE TECHNICAL DETAILS

1) HOME NETWORK CONTEXT

Towards defining firewall rules for a typical home network context, Table 2 defines private subnets, namely `192.168.0.0/16` with a subnet mask of `255.255.0.0`, `172.16.0.0/12` with a subnet mask of `255.240.0.0`, and `10.0.0.0/8` with a subnet mask of `255.0.0.0`. These private subnets are reserved for use within home and corporate networks [39]. In typical home networks, the commonly used address space is `192.168.0.0/16`, which is often automatically assigned by WiFi home routers. The corresponding `-dport` refers to standard (well-known) destination port numbers used for applications, such as web traffic. Commonly, well-known ports include 80 for HTTP and 443 for HTTPS, which are two dominant protocols in home networks. Allowing traffic from `192.168.0.0/16` to `dports 80, 443` permits internet traffic access. Also, Table 2 specifies the destination IP address 8.8.8.8 for the local DNS server (with destination port 53); this IP address must be changed according to the IP address of the ISP's local DNS server. However, this setup may not cover other protocols, such as audio/video streaming and IoT protocols. In such cases, the NTOP flow detection engine proves useful for profiling additional protocols.

2) FIREWALL PROCESSING

We deployed the IPtables open-source stateful firewall for IP address and port number inspection. Any inbound traffic

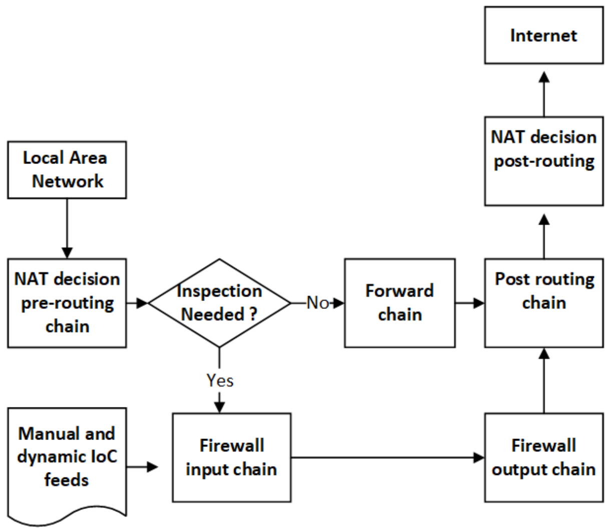


FIGURE 5. Flow of outbound traffic via SUTMS firewall engine.

initiated from the Internet will be blocked, and outbound traffic initiated from the LAN will be permitted on a need-to-know basis. Fig. 5 depicts the flow of outbound traffic initiated from the LAN destined for the internet. The flow of traffic through IPtables is processed as follows:

- The pre-routing chain is responsible for Network Address Translation (NAT) decisions before applying firewall rules. NAT can be dynamic or one-to-one static. Address translation of multiple IP addresses to a single-source IP address requires Port Address Translation (PAT) [40].
- If no firewall inspection is required, then the traffic will be forwarded directly to the outgoing interface. The traffic will be dropped if there is no route to the destination. To forward traffic, the IPtables must have IP-forwarding enabled.
- In case of firewall inspection, the traffic will be directed to the firewall input chain and processed according to the applied firewall rules. The SUTMS IPtables module consists of two sources of rules. An administrator adds one set of rules manually according to required functions. The NTOP flow detection engine can identify the protocols used in a home network, facilitating the development of manual firewall rules. While associating protocols in home networks may be challenging for non-technical users, observing the protocols from NTOP enables the creation of appropriate manual firewall rules. The other set of rules gets updated dynamically by incorporating IoC feeds from open-source threat intelligence platforms, e.g., Anomali via STIX/TAXII, see Sec. IV-C5 for details. The dynamic ingestion of dynamic IoC feeds into the IPtables input chain protects against bots.
- Allowed packets will go through packet processing tasks and be sent to the output chain. Traffic exiting SUTMS

TABLE 2. **IPtables rules for processing in firewall input chain in Figure 5.**

DNS Rule: `-A INPUT -p udp -m udp -s 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12 -d 8.8.8.8 -dport 53 -j ACCEPT`
Allow Web Traffic: `-A INPUT -p tcp -m tcp -m multiport -s 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12 -j ACCEPT -dports 80, 443`
SUTMS Management Rule: `-A INPUT -p tcp -m tcp -m multiport -s 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12 -j ACCEPT -dports 22, 10000`
NTOP Access Portal: `-A INPUT -p tcp -m tcp -s 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12 -dport 3000 -j ACCEPT`
Stealth Rule: `-A INPUT -j DROP`

will be inspected by the firewall output chain. Firewall input and output chains ensure bidirectional inspection of LAN traffic. This ensures that both incoming and outgoing traffic undergo inspection. All other packets will be forwarded to the post-routing chain.

- The post-routing chain is responsible for NAT packets after the processing of the firewall rules, whereas the pre-routing chain will perform NAT before the firewall rules are applied.

3) MANUAL SUTMS FIREWALL RULES

The necessary rules listed in Table 2 are applied to the firewall's input chain, where incoming traffic is first processed upon entering the firewall. In the absence of corresponding rules in the output chain, the traffic will be allowed after being processed by the input chain rules.

DNS Rule: The rule is required for internal hosts to communicate with DNS servers.

Allow Web Traffic: This rule allows HTTP and HTTPS traffic from the RFC 1918 address space [39] to the Internet.

SUTMS Management Rule: This rule allows management access to the SUTMS device via SSH and HTTPS over port 10000.

NTOP Access Portal: This rule allows access to the NTOP web portal for analyzing applications and traffic flows captured by the SUTMS framework.

Stealth Rule: This rule works as a catch-all rule and drops any traffic that does not have a matching rule.

NTOP plays a significant role in firewall rule development. Once protocols are identified, manual rules can be created using the port numbers commonly associated with IP address ranges used in home networks, such as 192.168.0.0/16. However, it is essential to baseline the traffic with NTOP over a period of time to detect all ports used by home network applications. This approach ensures the development of robust manual IPtables rules tailored to the specific home network setting.

4) RULE VERIFICATION AND LOGGING

The firewall rules can be verified by executing a command as detailed in [5] or by accessing the Web User Interface (WebUI) of SUTMS, e.g., Webmin. Webmin is an open-source web-based interface to manage Linux machines remotely via a web browser [41]. Webmin is highly customizable, and various modules can be developed according to user requirements. The web interface provides easy access to home users to add/delete and modify firewall rules.

The firewall rules of Table 2 are, by default, not persistent. To make the rules applied at reboot, the `iptables-persistent` package has to be installed [42]. The file `/usr/share/netfilter-persistent/plugins.d/15-ip4tables` must be pointed to the rules file `/etc/iptables/SUTMS.rules`.

The inspected traffic can be logged by creating a new chain and forwarding the messages to that chain. Denied log messages have a "denied" prefix in the message, and a severity log level of 4 will be able to capture sufficient details. The log levels are `debugged`, `info`, `notice`, `warning`, `err`, `crit`, `alert`, and `emerg` [43].

The command `-A FWLOGS -j LOG -log-prefix "denied: " -log-level 4` processes all packets to the chain "FWLOGS" and denied messages will be logged. The "FWLOGS" chain can be created either via the command line interface or Webmin. Log messages are stored in the `/var/log` directory and can be modified by editing `/etc/rsyslog.conf`. Forwarding the logs to a Syslog server is recommended, as storing firewall logs locally can fill up the disk space on the Raspberry Pi SD card.

5) INGESTING STIX/TAXII FEEDS: DYNAMIC ANTI-BOT PROTECTION

SUTMS takes advantage of Structured Threat Information eXpression (STIX) and Trusted Automated eXchange of Indicator Information (TAXII) [25]. STIX is a machine-readable language of threat intelligence information, and TAXII is a way to share that information [25], [26]. STIX-TAXII allows ingesting IoCs automatically. The feeds can allow dynamic blocking of bad actors by using the inspection capabilities of IPtables. The feeds can be collected using an API or manually via initiating an HTTP GET request to the STIX-TAXII server. Many open-source STIX-TAXII threat intelligence feeds are available. Some of the notable ones are MITRE [44], ANOMALI [45], and the U.S. Cyber Security and Infrastructure Security Agency feed [46]. SUTMS ingests feeds via an API call provided by Anomali [45], see [5].

6) IOC FEED FORMATTING AND AUTOMATION

The downloaded IoC feed will be stored in a local directory and can be accessed with Linux commands, such as `more` or `cat`. Once SUTMS has IoCs, proper formatting and automation are required so that the corresponding rules can be dynamically created by IPtables (anti-bot functionality).

IoCs can be converted into a format that can easily be embedded into IPtables by a simple while loop specified in Appendix A, Listing. 1. The program specified in Listing. 1 performs three main functions: i) downloading the IoCs, ii) extracting IP addresses from the entire content, and iii) removing any duplicate IP addresses.

D. ROUTING ENGINE

A home network typically has a WiFi access point that provides both WiFi access and routing service. SUTMS also possesses the capability to route traffic. The following scenarios are suitable for activating the SUTMS routing service: i) Home network lacks a dedicated routing device; ii) Existing WiFi access point does not support routing; iii) Home users seeking redundancy may opt to utilize the SUTMS routing service alongside the existing routing device; and iv) Inline deployment of SUTMS (see Sec. III-D for details). SUTMS does not support dynamic routing protocols, such as Open Shortest Path First (OSPF) [47], the Enhanced Interior Gateway Routing Protocol (EIGRP) [48], or the Border Gateway Protocol (BGP) [49], as these are enterprise protocols that are rarely used in home networks.

Home networks typically consist of one or two networks and a route to the Internet; this routing task can be easily accomplished by manually configuring static routes on SUTMS. For instance, SUTMS can have a static route for `192.168.0.0/16` (commonly used in home networks) and a default route (`0.0.0.0/0`) pointing to the internet router. Whereby, `/16` represents the subnet mask `255.255.0.0`, covering a large range of IP addresses within the `192.168.x.x` range, while `/0` signifies all possible IP addresses, indicating the default route.

Static routes are typically preferred for a home network, as the static routes avoid the complexity of dynamic routing protocols and have a minimal impact on the SUTMS system resources. In contrast, dynamic routing protocols are used in environments with hundreds of routes, where the manual configuration of static routes would be challenging. In addition to managing a large number of routes, organizations often implement dynamic routing for redundancy. If one route fails, then traffic can automatically reroute via a different route. However, in a typical home network with only one internet connection, dynamic routing does not offer the same redundancy benefits.

E. LOGGING ENGINE

Logs are crucial for troubleshooting, security analysis, and forensic investigations. Logging requires additional storage and processing power. Therefore, we designed SUTMS to only provide the limited logging functionalities illustrated in Figure 6. Further log analysis would have to be conducted via a third-party system.

In our SUTMS evaluations, we followed the typical practice of home network devices to store logs locally, whereby the logs are often erased after a couple of days due to limited disk space. More specifically, we used the built-in default logging setup for NTOP, Suricata, and Iptables, which stored logs on the Raspberry Pi in the `/var/log` directory (in corresponding subdirectories, such as `/var/log/ntop` for NTOP, `/var/log/suricata` for Suricata, and `/var/logs/iptables` for Iptables).

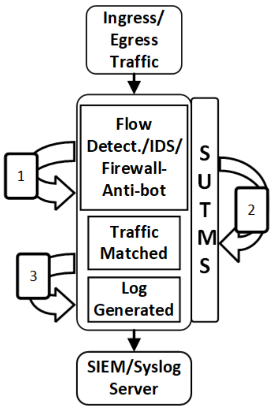


FIGURE 6. SUTMS logging engine: Step 1: Traffic is inspected by Flow detection, IDS, and Firewall engines; Step 2: Traffic matches by any of these engines will generate logs; Step 3: Logs will be sent to third-party systems, e.g., SIEM, Syslog server.

A local UTM device, such as a Raspberry Pi, has only limited storage which necessitates the periodic deletion of the logs. Optionally, SUTMS can send logs to an external log collector, e.g., with Syslog [50]. External log collection enhances the SUTMS security services by enabling features that require reliable long-term log storage: i) Review of historical information, such as attacks and intrusions, for forensic and auditing purposes; ii) Correlation of logs from IDS and firewall (possibly with additional security-related logs, e.g., from Virtual Private Network (VPN), endpoint protection, and antivirus tools) in centralized additional security tools, such as Extended Detection and Response (XDR) [51]; iii) Forecasting of future attacks based on historical attack trends using machine learning. We also note that when corporate employees work from home, i.e., when a home network becomes an extension of a corporate network, then corporations may require that the SUTMS logs be sent to the corporate external log collector (server). At the corporate log server, the enhanced security services can then be conducted under the auspices of the corporate employer.

SUTMS can send the log data via common external logging protocols, such as Syslog [50], to a centralized third-party log server, e.g., a logstash server [31]. In particular, SUTMS can be configured to send the NTOP, IDS/IPS, and firewall alerts and logs to an external third-party log server, see the information flow illustration in Fig. 6. It is recommended to integrate the external logging with a Security Incident Event Management (SIEM) system, and this integration may be required for corporate machines. The logging data can be integrated with common SIEM systems, such as Splunk [52], ELK Stack [53], or AlienVault OSSIM [54].

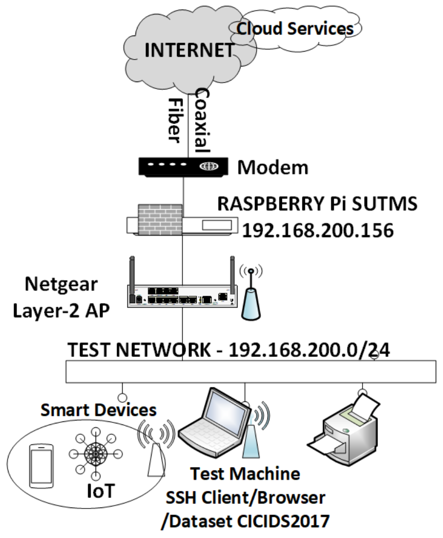


FIGURE 7. Topology of SUTMS evaluation network.

V. SUTMS EVALUATION AND VALIDATION

This section evaluates the SUTMS detection accuracy and computational efficiency using a standard dataset both with and without IDS signature optimization. The evaluations are performed on a special-purpose network that is representative of typical real-world home and Small Office, Home Office (SOHO) networks.

A. TEST NETWORK AND DATASET

For the SUTMS evaluations, we have deployed a network consisting of various computing devices, including smartphones, IoT, laptops, and printers. The test network topology is illustrated in Fig. 7. SUTMS is installed in the in-line mode utilizing the existing WiFi AP, see Fig. 3(b), and acts as a central inspection point. SUTMS has access to the Internet for signature updates, cloud integration, STIX/TAXII feeds, and remote security management. Any traffic destined to the Internet, regardless of device type, is inspected. The network provides both WiFi and Ethernet access. SUTMS can be configured via SSH version 2 or by directly connecting to a console (keyboard, mouse, and monitor). The test machine or SUTMS can be used to run the dataset, and the output of the results is exported to Elasticsearch [31] and Kibana [32] for analyses and graphical representation.

We utilize the manual firewall rules in Table 2. In particular, HTTP and HTTPS ports are allowed for outbound web browsing traffic, and DNS traffic (UDP traffic to destination port 53) destined for the trusted DNS server is also allowed. SUTMS management traffic (TCP traffic to destination port 22 for SSH, and TCP traffic to destination port 10000 for Webmin) as well as NTOP access (TCP traffic to destination port 3000) are allowed from the internal network, i.e., `192.168.200.0/24`. Whereby, the private subnet 192.168.0.0/16 in Table 2 is more generic representing the entire /16 subnet, whereas for our evaluation we used the corresponding /24 subnet, see Fig. 7. Traffic that did not match the rule base was blocked and logged for analysis.

TABLE 3. Percentages of home network applications detected by NTOP (utilized for optimizing Suricata IDS signatures in Phase II of the IDS evaluation).

Protocol	Usage (%)
HTTP	36.1
NTOP	57.6
SSH	3.1
Other	3.3

Specialized datasets are required to evaluate UTM solutions. We selected the CICIDS2017 dataset [55], [56], [57]. The dataset is used to evaluate the threat detection accuracy and utilization of computational resources. The 8 GB dataset in pcap format includes common and relatively newer attacks and was last updated on 2019-09-10. The reasons for selecting the CICIDS2017 dataset are.

- It is built up of a unique profiling mechanism, i.e., the B-Profile system [56]. It allows the simulation of human behavioral traffic patterns and benign attacks.
- It generates data from 25 users and commonly used protocols, e.g., HTTP, HTTPS, SSH, FTP, and email. The number of users and protocols closely match a typical home network.
- It includes common home network attacks, e.g., Distributed DoS (DDoS), brute force attacks, HTTP/HTTPS exploitation, C&C communication, and DDoS.
- The data in pcap format is easier to manage and simulate than actual user traffic.
- The dataset is designed for IDS evaluation. However, the size and quality of the dataset are significant enough for stress testing and for evaluating the inspection capabilities of SUTMS IDS with other modules enabled, i.e., flow detection and firewall.

B. FLOW DETECTION ENGINE EVALUATION

The applications discovered during the processing of the CICIDS2017 dataset in our test network are listed in Table 3. NTOP represents management traffic to SUTMS and can be ignored. Protocols categorized as 'Other' are linked with broadcast traffic, including the Dynamic Host Configuration Protocol (DHCP) and the Address Resolution Protocol (ARP). Such broadcast traffic is commonly observed in LAN environments and typically does not require signature enablement.

HTTP constitutes about 36.1% of the overall traffic. Traffic identified in Table 3 will allow us to only enable IDS signatures for the HTTP protocol (in Phase II of our IDS evaluation), which greatly reduces the utilization of computational resources by the UTM engines.

C. INTRUSION DETECTION ENGINE EVALUATION

The SUTMS IDS engine (which we operate in the IDS mode in the evaluation) is built upon Suricata and evaluated against the CICIDS2017 dataset [55], [56], [57]. The evaluation has two phases: In Phase I, signature detection and system resources are evaluated with default Suricata configurations.

TABLE 4. Numbers of SUTMS IDS events for different event types over the course of four 45-minute segments of processing CICIDS2017 dataset [55], [57] in evaluation Phase I (without IDS signature optimization).

IDS Event	Time (min)				Totals
Types	**00–45**	**45–90**	**90–135**	**135–180**	
Anomaly	4	7		12	23
DCE/RPC		453		150	603
DHCP		500	1340	1167	3007
DNS	19,384	27,160	56,230	115358	218132
FileInfo				151	151
Flow – FP	1769	3178	2739	1325	9011
Flow – TN	28031	49246	43118	22932	143327
FTP		97		11133	11230
FTP_data		97		10043	10140
HTTP	765		173	1566	2504
KRB5		500	459	1312	2271
SMB				147	147
SNMP				650	650
SSH		224	76	170	470
TLS (HTTPS)	1342		348	1420	3110
Totals	51543	80276	104836	168121	**404776**

TABLE 5. SUTMS security events in Phase I, described with default Suricata rule names.

Signatures fired (From highest to lowest number of hits)
SURICATA TCPv4 invalid checksum
SURICATA STREAM CLOSEWAIT FIN out of window
SURICATA Kerberos 5 weak encryption parameters
SURICATA HTTP unable to match a response to request
ET INFO Observed DNS query to .cloud TLD
SURICATA STREAM FIN1 FIN with wrong seq
SURICATA TLS invalid record/traffic
SURICATA UDPv6 invalid checksum
SURICATA Applayer detects protocol only one direction
SURICATA FRAG IPv4 fragmentation overlap
SURICATA HTTP invalid response chunk len
SURICATA HTTP request line incomplete
ET DNS Query for .cc TLD
SURICATA STREAM SHUTDOWN RST invalid ack
ET INFO Observed zero SSL SSL/TLS Certificate
ET INFO Observed DNS query to .biz TLD

In Phase II, results from the NTOP flow detection engine are integrated into IDS, and the default rule set is modified according to the detected application protocols. Specifically, in Phase II, SUTMS will be evaluated against the modified set of configurations, while the core services (NTOP and firewall) will run as part of the UTM evaluation. We evaluate the threat detection accuracy separately for the IDS and firewall engines, as well as the computing resource utilization for the entire SUTMS. Throughout Phase I, all the UTM services will be running to ensure device integrity. It is not beneficial to test the IDS without the flow detection and firewall engines running because all the components in a UTM device are interdependent regarding resource allocation. Cockpit [58] and Netdata [59] are utilized for monitoring CPU utilization, memory (RAM) utilization, system load, and disk I/O.

1) THREAT DETECTION PHASE I

a: OVERVIEW

Phase I evaluates the IDS engine with default Suricata configurations. Over the span of 180 minutes of the dataset execution, SUTMS detected 404,776 events, see Table 4. The SUTMS IDS engine has detected a few anomalies, e.g., 4 anomalies in the 00–45 min time span. These anomalies are primarily attributed to variations in signature patterns.

In contrast, anomalies identified by the flow detection engine stem from a comprehensive traffic analysis of traffic. Specifically, there were 7 and 12 anomalies discovered by the flow detection engine in the time spans 45–90 and 135–180 minutes, respectively, see Table 4. While four instances of anomalies were discovered during the first 45 minutes of the evaluation, the majority of anomalies were detected during the latter part of the evaluation. This discovery pattern is primarily attributed to anomalies being identified relative to a baseline. Towards the end of the dataset, traffic patterns deviated from the normal patterns observed in the initial 135 minutes, resulting in a higher number of anomalies being observed. In a typical home network, once the normal traffic behavior (e.g., usage, protocols, data transfer rates) is determined, anomalies can be identified based on deviations from this baseline. The longer SUTMS runs, the more opportunity it has to train and identify anomalies with greater confidence. This capability is crucial for detecting advanced persistence threats that may evade detection by traditional IDS and firewall systems.

From Table 4, we observe that the DNS, Flow (FP, TN), and FTP/FTP_data event types represent approximately 54%, 38%, and 5%, respectively, of the total number of events. Another important aspect to note is Transport Layer Security (TLS) traffic, which represents HTTPS traffic. Despite its share being less than 1%, detecting HTTP traffic is crucial for identifying web vulnerabilities.

Furthermore, a high number of DNS events presents an opportunity to correlate these occurrences with firewall anti-bot hits. An increase in DNS events may signal potential data exfiltration and C&C communication. Consequently, the number of firewall hits rises, as the SUTMS firewall is equipped to detect C&C communication through STIX/TAXII IoC feeds, cf. the increase in firewall hits in the time span 120–180 minutes in Figure 8.

Table 5 lists the types of detected signatures. IDS successfully detects matching signatures and hits, whereby *"SURICATA TCPv4 invalid checksum"* and *"SURICATA STREAM CLOSEWAIT FIN out of window"* have the highest numbers of recorded hits. Invalid checksum and out-of-window packets can indicate a DDoS attack. DNS alerts are useful in investigating data exfiltration and C&C communication. A high number of events and alerts can waste system resources and generate False Positives. In Phase I, we evaluated the utilized system resources without IDS signature optimization, i.e., without IDS ingesting data from the NTOP flow detection engine.

b: IDS ACCURACY

The effectiveness of signature detection can be quantified with the following characteristic event counts that we evaluate for the full 180-minute dataset: The number *TP* of True Positive detections counts the number of successfully detected signatures. *TP* is straightforward to measure since SUTMS detected all the dataset attacks and protocols. The CICIDS2017 dataset primarily comprises the FTP, HTTP, SSH, TLS (HTTPS), and email protocols, which are typical for home networks. SUTMS detected all the IDS events for FTP, HTTP, SSH, and TLS (HTTPS), as detailed in Table 4. Specifically, based on Table 4 (Phase I):

$$TP = HTTP + HTTPS + FTP + FTP\_data + SSH \quad (1)$$
$$= 2504+3110+11230+10140+470 = 27454. \quad (2)$$

However, Suricata employed in SUTMS does not comprehensively cover email protocols. Accordingly, the Simple Mail Transfer Protocol (SMTP) and the Internet Message Access Protocol (IMAP) are captured within flows by SUTMS rather than as individual protocols (in future research, signatures for such email protocols could manually be added to detect these protocols accurately by Suricata). Since not all flow traffic pertains to email, flows can contribute to the number *FP* of False Positive detections, whereby *FP* counts the number of instances of signatures being incorrectly labeled as attacks. There were 9011 mail protocol-related events observed as flow – FP in Phase I, that were categorized incorrectly, i.e.,

$$FP = \text{Miscategorized email events} = 9011. \quad (3)$$

The number *FN* of False Negative detections counts the number of missed attacks. Since SUTMS did not miss any attacks,

$$FN = 0. \quad (4)$$

The number *TN* of True Negative detections counts the number of instances of signatures being correctly labeled as benign. In particular, many events in Table 4 are not related to the CICIDS2017 dataset but are associated with the required Ubuntu Operating System (OS) and SUTMS services, namely the DHCP, SNMP, DNS, Distributed Computing Environment / Remote Procedure Calls (DCE/RPC), DHCP, DNS, FileInfo, Kerberos: The Network Authentication Protocol (KRB5), Server Message Block (SMB) and Simple Network Management Protocol (SNMP). These events related to SUTMS services are correctly categorized as benign and thus considered as *TN*, i.e.,

$$TN = Total - (TP + FP + FN) \quad (5)$$
$$= 404,776 - (27454+9011+0) = 368311. \quad (6)$$

TP, TN, and FN are critical accuracy characteristics. TP and TN characterize accurate detection, while FN highlights the IDS's inability to detect malicious traffic. Ideally, higher TP and TN numbers and lower FN numbers are desired. Based on the preceding signature detection characteristics, we obtain the SUTMS IDS accuracy for the evaluation Phase I as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$
$$= \frac{27454 + 368311}{27454 + 368311 + 9011 + 0} \quad (8)$$
$$= 0.9777 \approx 97.7\%. \quad (9)$$

The detection of IDS events across a variety of protocols highlights SUTMS's superior accuracy in protocol detection. As illustrated in Table 4, eleven different protocols were successfully identified (whereby Anomaly, FileInfo, Flow-FP, and Flow-TN are not protocols).

Another critical aspect of the CICIDS2017 dataset is its capability to simulate brute force, DoS, DDoS, infiltration, Heartbleed, bot, and scan attacks. All these attacks are detected and referenced in Table 5, except for infiltration, bot, and scan attacks. The infiltration, bot, and scan attacks are mitigated by the firewall's anti-bot module and have been successfully blocked. However, it is essential to note that a single attack can generate multiple signatures, and there were instances where signatures were generated for benign traffic. These factors also contribute to false positives. For example we observe from Table 5 that both "SURICATA UDPv6 invalid checksum" and "SURICATA UDP packet too small" alerts are triggered, yet only one of the alerts is associated with a UDP application-based DoS attack and the other alert is benign.

c: FIREWALL ACCURACY

SUTMS can detect and block attacks by leveraging both the IDS and firewall engines. While the threat detection evaluation has so far focused on the IDS engine, we now turn to the firewall engine. The firewall engine inspects traffic based on Layer 3 and Layer 4 protocols. Rules implemented in the firewall input chain determine whether traffic is allowed or denied. Logs are generated to track rule hits, validating the firewall's functionality. The firewall accuracy is assessed in terms of TP, TN, FP, and FN; whereby, in firewall terminology, TP refers to successfully blocking malicious traffic, while TN denotes successfully permitting legitimate traffic according to the rule base. FP and FN typically arise from misconfigurations or malfunctions of the firewall. The analysis of the firewall logs indicated that throughout our evaluations, the SUTMS firewall engine operated according to the rules. Hence, no FP or FN were observed, resulting in an accuracy of one; we conservatively report an accuracy of 99.99% in Table 8, which is consistent with common Service Level Agreements (SLAs) and network availability levels [60].

2) THREAT DETECTION PHASE II

a: IDS ACCURACY

Phase II evaluates the IDS engine after optimizing the signatures based on the application data from NTOP, i.e., in Phase II of the evaluation, the IDS signatures were tuned according to the traffic observed by the NTOP engine. Flow

TABLE 6. Numbers of SUTMS IDS events in evaluation Phase II (with IDS signature optimization based on NTOP flow detection, see Table 3).

IDS Event	Time (min)				Totals
Types	**00–45**	**45–90**	**90–135**	**135–180**	
Anomaly	1	2		4	7
DCE/RPC			12	120	132
DHCP		50		1117	1167
DNS	14,312	11,510	41,230	85010	152062
FileInfo				111	111
Flow – FP	786	682	48	683	2199
Flow – TN	21758	20438	1098	25912	69206
FTP	12			13	25
FTP_data	4			9	13
HTTP	15		13	1166	1194
KRB5		420		112	532
SMB	12			147	159
SNMP				60	60
SSH			76	120	196
TLS (HTTPS)	1122		38	1118	2278
Totals	37726	33120	42576	115919	**229341**

TABLE 7. SUTMS security events in Phase II.

Signatures fired (From highest to lowest number of hits)
SURICATA HTTP unable to match a response to request
SURICATA Kerberos 5 weak encryption parameters
SURICATA STREAM bad window update
SURICATA STREAM excessive retransmissions
SURICATA STREAM TIMEWAIT ACK with wrong seq
SURICATA HTTP gzip decompression failed
SURICATA STREAM FIN1 FIN with wrong seq
SURICATA TLS invalid record type
SURICATA TLS invalid record/traffic
SURICATA HTTP invalid response chunk len
SURICATA UDPv6 invalid checksum
SURICATA STREAM reassembly overlaps with different data
SURICATA Applayer detect protocol only one direction
SURICATA STREAM packet with invalid timestamp
ET DNS Query for .cc TLD
SURICATA STREAM SHUTDOWN RST invalid ack
SURICATA FRAG IPv4 fragmentation overlap
SURICATA UDP packet too small
SURICATA Applayer wrong direction first data

events were consequently reduced as only flows for tuned protocols were observed. Overall, the IDS signature tuning nearly halved the number of events to 229,341, see Table 6.

DNS constitutes approximately 66%, flows 31%, and TLS (HTTPS) 1% of the total events. The most frequently detected signatures at the top of Table 7 correspond to HTTP, TLS, and UDP (possible DNS) protocols. The integration of the NTOP data into the IDS engine has enhanced the detection capabilities by targeting common protocols and reducing False Positives. Specifically, based on Table 6, we evaluated the quantitative IDS detection metrics analogously to Phase I. For Phase II, we obtained: $TP = 3706$, $FP = 2199$, $FN = 0$, and $TN = 223436$, resulting in an Accuracy of 0.9904, i.e., approx. 99.0%.

Inspecting encrypted traffic is a challenge. Detection of Kerberos and TLS-based signatures such as *"SURICATA Kerberos 5 weak encryption parameters"* and *"SURICATA TLS invalid record type"*, respectively, is an indication of early detection of weak ciphers and TLS compliance failures before the encryption process starts. It could also highlight any outliers in the form of anomalies, new ciphers, weak authentication, and possible C&C communication.

b: SUMMARY

Overall, our signature detection results in both Phases I and II indicate a negligible number of False Negative (FN) detections; actually, none were observed. However, a notable occurrence of False Positive (FP) detections may persist. The FP detections could be addressed through manual fine-tuning of the signature database.

Another significant impact of signature refinement is the reduction in the total number of security events. This decrease enables a clearer focus on alerts with medium or higher severity levels. Alerts are typically categorized by severity levels, such as informational, low, medium, high, and critical. In a typical home network, there may be numerous informational and low-level alerts, making it challenging to prioritize critical or high-severity alerts. One approach to address this issue is to establish a threshold for informational and low-level alerts. For example, for every ten informational events (i.e., messages that can be valuable for the incident response team), only one alert would be triggered, with a counter incrementing to represent ten instances of the same event, rather than generating ten individual alerts.

3) UTILIZED COMPUTING RESOURCES IN PHASE I

We measure the CPU utilization in percent, the memory usage in GB, the system load defined as the total number of processes that run on the CPU at a given time (i.e., number of processes in the CPU run queue), and the disk input/output (I/O) in operations/second while the dataset runs. During the IDS test window, all the UTM services, i.e., flow detection, intrusion detection, and firewall engine, including routing engine and logging, were enabled to inspect and analyze traffic to measure actual usage of the computing system resources. Table 8 presents the summary statistics, i.e., the mean, the standard deviation, and the maximum value, of the measured resource utilization metrics. Specifically, we measured the peak CPU and memory utilizations as well as the averages of the system load and disk I/O over 10-minute intervals of the SUTMS operation on the Raspberry Pi 4. We conducted the measurements with SNMP and coordinated the metric collection and statistics with Netdata, whereby we employed the default Netdata settings that poll SNMP every second. The measurement resulted in 18 samples for each metric over the 180 minutes of the processing of the CICIDS2017 dataset [55]. We evaluated the summary statistics (mean, standard deviation, and maximum) from these 18 samples.

a: CPU UTILIZATION

The peak CPU utilization jumped from the baseline (idling) level of 3%-4% to 22% as the dataset inspection started. The peak CPU utilization over the considered 10-minute time intervals remained nearly constant over the entire 180-minute evaluation with an overall mean peak CPU utilization of 23% and a standard deviation of 1.1%.

TABLE 8. Summary statistics mean (standard deviation) [maximum] of utilized computing resources over 180 minutes of SUTMS processing of CICIDS2017 dataset [55], whereby the peak CPU and memory utilizations as well as average process load and disk I/O operations per second over 18 10-min time periods are evaluated, e.g., the mean peak CPU utilization is the mean of 18 measured peak CPU utilizations (each covering 10-min).

Operat. Scenario	Peak CPU Util. [%]	Peak Mem. Util. [GB]	Avg. Load [# proc.]	Avg. Disk I/O [ops./sec.]	Accuracy
All UTM Svcs., Phase I	23.08 (1.11) [26.0]	2.03 (0.17) [2.34]	2.04 (0.41) [2.52]	1.52 (2.48) [6]	99
All UTM Svcs., Phase II	20.40 (0.34) [21.3]	0.92 (0.06) [1.01]	1.25 (0.11) [1.50]	0.67 (1.92) [6]	99
Firewall & NTOP	8.38 (1.20) [9.2]	0.13 (0.03) [0.2]	0.93 (0.01) [1]	0.38 (0.72) [2.34]	99.99

b: MEMORY UTILIZATION

We observed an initial spike in peak memory utilization to 2.34 GB and then stabilized peak memory utilization around 2 GB, see Table 8. The observed initial spike in memory utilization is mainly due to the allocation of resources for the initial Suricata processes to run. Once the startup processes, such as the capture and detection engines, have been loaded, memory usage stabilizes.

c: SYSTEM LOAD

The system load counts the total number of processes that run on the Raspberry Pi 4 CPU on average during a 10-minute interval. As Table 8 indicates, the process load stayed nearly constant at two processes. More specifically, we observed 2.5 processes for the first 30 minutes of the CICIDS2017 dataset processing, as reported in more detailed tables available from [5], which we cannot include due to space constraints. Further, for minutes 30–150, there were 1.8 to 2.2 processes, and then 1.2 to 1.4 processes for minutes 150-180. The reduction of the running processes after approximately 30 minutes suggests that most detections occur during the initial stages of the dataset inspection. The further reduction after 150 minutes suggests that a similar detection pattern may persist in later stages, and that could free up some system load from a process standpoint.

d: DISK I/O

Disk read and write can be problematic, especially when dealing with systems with non-volatile flash memory (SD-Card), such as the Raspberry Pi. Table 8 indicates that the SUTMS disk I/O exhibits some variability with maximum 10-minute averages of up to 6 operations per second. More detailed evaluations [5] revealed that there are less than 0.1 operations per second for most of the 180-minute evaluation; however, there were five 10-minute measurement intervals with averages of 5 to 6 disk I/O operations per second. The elevated disk I/O suggests that the dataset is inspected in blocks, whereby the processing of large data blocks leads to increased disk I/O operations.

e: SUMMARY OF UTILIZED SYSTEM RESOURCES IN PHASE I

Overall, the Raspberry Pi 4 performed the full SUTMS functionalities for a realistic home network dataset while experiencing a maximum peak CPU utilization of 26%, a maximum peak memory (RAM) usage of 2.34 GB, a maximum 10-minute average load of 2.52 CPU processes, and a maximum 10-minute average of 6 disk I/O operations/s. Throughout, the Raspberry Pi 4 performed well within the specifications (capabilities) of its computing system in Phase 1, which did not optimize the IDS signatures.

TABLE 9. Traffic detected by SUTMS NTOP engine for firewall and NTOP evaluation.

Protocol	Usage %
SSL	66.8
Google Services	11.4
Ubuntu	7.7
Other	7.5
SSH	6.6

4) UTILIZED SYSTEM RESOURCES IN PHASE II

In Phase II of the evaluation, all the conditions were kept the same as in Phase I except for the addition of signature optimization for IDS based on the NTOP flow detection engine. The IDS signature optimization slightly reduced the mean peak CPU utilization to 20.4%, while the peak memory usage was approximately halved to a mean of 0.92 GB, and the mean 10-minute average process load was reduced to 1.25 processes. The disk I/O still has maximal 10-minute averages of 6 operations per second, while the mean has been reduced to less than half.

Overall, we conclude that signature refinement (Phase II) can substantially reduce the computing system resources required for SUTMS operation. Whereby, mean memory utilization and system process load were reduced the most, i.e., by 55% and 38%, respectively, without compromising the detection accuracy, see Table 8. In future work, the CPU utilization could be reduced further by disabling low-priority signatures.

In both phases, it is advisable to consider sending the logs to third-party log collection engines, such as Syslog or SIEM. The Raspberry Pi 4 with 32 GB will be able to save logs for weeks, but eventually, it will run out of space. An automated script can also be created to remove logs older than a week or two, depending on a predefined data retention policy.

D. FIREWALL AND NTOP ENGINE EVALUATION

1) FIREWALL RULES

SUTMS is equipped with an open-source IPtables firewall inspection engine, which is configured with two types of rule sets. Specifically, allowed rules for traffic detected by NTOP (see Table 9) were manually configured. Whereby, Table 9 is specifically utilized for the development of firewall rules, focusing on the traffic observed in home networks. In contrast, Table 3 was employed to optimize the Suricata IDS signatures during Phase II of the evaluation.

It is worth noting that when comparing the applications and protocols discovered between these two tables, we find some differences. For instance, in Table 9, the Secure Socket Layer (SSL) protocol encompasses multiple protocols, including HTTP, HTTPS, and NTOP (given that NTOP management uses HTTP port 3000), resulting in a relatively high percentage of 66.8% SSL traffic. In contrast, Table 3 separates NTOP from HTTP, whereby NTOP accounts for approximately 57.6% of the traffic. Yet, it is important to note that Table 3 includes Google services within the NTOP category, as some Google services utilize the HTTP protocol. This discrepancy indicates that NTOP may categorize applications or protocols differently based on the port used. For example, HTTP traffic over a non-standard HTTP port may be classified as a distinct application. NTOP has the capability to associate protocols with applications, such as Google and Ubuntu OS services, but this categorization is not always accurate, as seen in the two tables. Nevertheless, this issue can be addressed by manually adjusting default NTOP protocol signatures to better suit home networks. This adjustment should not pose a significant challenge, as home networks typically rely on a limited number of protocols, allowing for relatively easy customization

As NTOP serves as an intermediary tool providing protocol information to the IDS engine for signature optimization, effective NTOP protocol detection ensures that no IDS signatures are missed. However, incorrect categorization by NTOP may lead to False Positives, impacting the overall accuracy. For example, miscategorized protocols may inadvertently trigger enabled signatures. To mitigate such issues, conducting baseline traffic analysis over a period of 6 to 12 weeks can train NTOP to accurately classify all services and manually correct any misclassifications. In terms of firewall operations, miscategorization has no impact as the firewall operation relies solely on static Layer 3 and Layer 4 information, which remains unchanged. The dynamic firewall anti-bot feature with the STIX/TAXII feed will not be affected by protocol miscategorization since IoCs are in the form of IP addresses, domains, and URLs, with protocol not being included in the feeds. IoCs also exhibit high fidelity, so all traffic to and from IoCs should be blocked regardless of the utilized protocol.

Overall, the NTOP evaluations underscore the significance of the SUTMS flow detection engine with its myriad benefits, including optimizing IDS signatures, generating firewall rules, and detecting anomalies.

Returning to the firewall rules, we note that another rule set is automatically created according to the IoC feeds using STIX/TAXII. The rules are generated using the conversion script specified in Section IV-C (see also Appendix A). This way, the STIX/TAXII feeds are automated, optimized, and customized. In the end, there is a default block rule to make sure any traffic that is not matched is blocked; see Table 2.

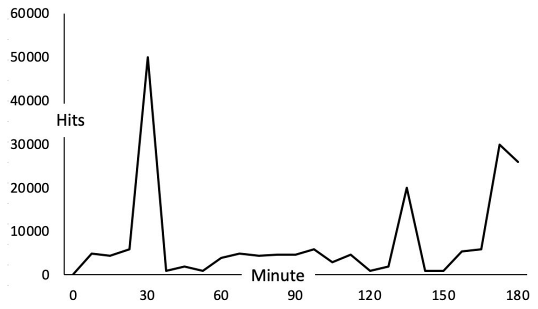


FIGURE 8. Firewall and NTOP evaluation: SUTMS firewall rule hits per minute during 180 minutes of executing the CICIDS2017 dataset.

2) DETECTION PERFORMANCE

Figure 8 shows the firewall hits per minute, i.e., the number of packets processed by the firewall input/output rule chain per minute. The average rule hits observed was around 5000 per minute, with a maximum of approximately 50,000 hits/minute, see Fig. 8. Notably, the total number of firewall rule hits in Fig. 8 is significantly higher than the total number of IDS events in Tables 4 and 6, primarily because the firewall inspects traffic at Layers 3 and 4, while IDS events occur at the application layer. This discrepancy arises from the dataset's substantial volume of traffic, where not all traffic qualifies for IDS inspection, as not every flow contains a payload. Conversely, every traffic stream must pass through Layers 3 and 4, and the firewall either accepts or blocks traffic at these lower layers before reaching the application layer for IDS inspection. For example, during a scan, there may only be a few scan events detected by the IDS, whereas at Layers 3 and 4, the entire subnet along with all TCP/UDP ports from 0 to 65535 will be scanned. This explains the notably higher number of firewall hits compared to IDS events.

When the dataset is executed, the hits per minute fluctuate according to the event trace in the dataset, indicating that the firewall is effectively enforcing its rules. The IPtables firewall processes traffic in real-time based on input and output chain rules without employing a buffer. However, since we utilize the (Ubuntu) Linux kernel for IPtables, the kernel manages a dedicated buffer for incoming packets on interfaces. In scenarios of high-volume traffic, the Linux kernel employs various mechanisms, such as interrupt handling, quality of service, and dynamic memory allocation for bursty traffic. These mechanisms ensure efficient packet processing without compromising the performance of overlay services, such as IPtables.

3) UTILIZED COMPUTING RESOURCES

One major difference between the IDS evaluation in Sections V-C and the Firewall/NTOP evaluation is that during the IDS evaluation, all the core engines (including NTOP flow detection and IPtables firewall along with routing and logging) were also running. In contrast, in the Firewall/NTOP testing, the IDS engine was disabled to measure in isolation the impact of the Firewall/NTOP core services on the overall UTM device (while routing and logging were also still running). For the firewall/NTOP evaluation, traffic was generated from the CICIDS2017 dataset [55], as for the IDS evaluations. SNMP polling was used to monitor the computational resources utilized by the firewall every second.

The evaluation of the utilized system resources in Table 8 indicates that the mean peak CPU utilization and the average disk I/O are roughly halved compared to the IDS Phase-II testing. In particular, Firewall/NTOP only have approximately 40% of the mean peak CPU utilization of the IDS Phase II evaluation, indicating that the IDS inspection engine consumes approximately 60% of the CPU resources in the IDS Phase II evaluation. Also, the system load was slightly lower for Firewall/NTOP only than for the IDS Phase-II evaluation.

Remarkably, the mean peak memory usage of the firewall and NTOP is less than one-seventh of the corresponding IDS Phase II memory usage; which, in turn, indicates that the IDS inspection engine consumes the vast majority of the memory resources when running the full set of UTM services.

VI. CONCLUSION AND FUTURE DIRECTIONS

A. SUMMARY

This applied research article introduced the SUTMS design for Unified Threat Management (UTM) in home networks. We make the SUTMS code and configurations publicly available [5] so as to facilitate the practical use of the SUTMS design and further research on UTM for home networks. For typical home network security events, SUTMS with IDS signature optimization based on flow detection (Phase II) achieved 99% accuracy with significant reductions in memory utilization, process load, and disk I/O compared to out-of-the-box Suricata configurations (Phase I). The evaluation of individual components exposed IDS as the single most resource-intense process. We upgraded the firewall engine from a traditional access controls list to dynamic STIX/TAXII feeds. These dynamic feeds introduced the capability of proactive blocking of bad actors (bots). The integration of the NTOP flow detection engine served multiple purposes: flow detection allows the SUTMS to identify anomalies, and application awareness assists the SUTMS in fine-tuning (optimizing) IDS signatures, ultimately reducing CPU and memory usage.

Generally, running multiple inspection engines efficiently on a single device with limited resources is a main challenge for SOHO UTM appliances. SUTMS successfully addressed the performance issues without compromising accuracy. SUTMS integrates traditional IDS and firewall inspection capabilities with modern prevention techniques, including application awareness, anti-bot protection, and anomaly detection, through innovative methods to develop next-generation home network protection solutions. Specifically, SUTMS utilizes application data from a flow detection engine to optimize IDS signatures, thus reducing false positives and enhancing system resource efficiency, so that SUTMS can readily run on a Raspberry Pi 4. That is, the flow detection engine is not only beneficial in detecting anomalies but also assists in tuning IDS signatures according to the active protocols and applications. Also, the firewall's anti-bot feature adds a layer of blocking by dynamically preventing C&C communication and data exfiltration attempts.

B. FUTURE RESEARCH DIRECTIONS

Building on the SUTMS design presented in this article, the following improvements should be examined in future research.

1) SECURE SOCKETS LAYER (SSL) INSPECTION

According to the F5 Networks malware report [61], 80% of internet traffic is encrypted, and approximately 46% of malware was hidden using encryption techniques, e.g., SSL or TLS. However, advanced malware variants obfuscate themselves within the payload using encryption techniques, requiring full packet decryption. New avenues of SOHO UTM devices will need to be explored in future research to optimize SSL processing and simplify the encryption/decryption mechanism.

2) STIX/TAXII THREAT FEED VALIDATION

Threat intelligence feeds are crucial for blocking Indicators of Compromise (IoCs). IoCs typically appear as IP addresses, domains, URLs, and hashes. SUTMS relies on open-source feeds, such as Open Threat Exchange (OTX) [22] and Anomaly [45]. Open-source platforms typically lack verification, real-time updates, and targeted adversaries, such as home network/system attackers and industry-specific hacking groups. Human intervention will be required to review IoCs geared toward home networks and client machines before applying the IoCs to access control lists. Commercial threat intelligence vendors, such as Crowdstrike [62] and ThreatConnect [63], extract data from various intelligence sources and validate and recommend tailored profiles for SOHO networks to process IoC lists of manageable size according to customer needs. However, a major issue lies in the fragmentation of IoC lists across various platforms. Crowdstrike, Threatconnect, and open-source intelligence platforms each maintain their own IoC lists, often resulting in overlap and the inclusion of false positives or items with low confidence levels.

There is a pressing need for a centralized platform capable of vetting IoCs from different sources, including both commercial off-the-shelf (COTS) and open-source platforms, prior to their integration into UTM systems, whereby it is important to note that these IoC lists can be substantial, potentially comprising thousands of IP addresses, domains, and URLs. UTM systems for home networks have only limited computing system resources; therefore, only IoCs with high fidelity should be ingested. More broadly, future research should develop a solution offering automated threat response grounded in validated IoCs and customized to suit diverse customer networks, including both home and enterprise environments. IoC lists often contain duplicates, false positives, or items specifically tailored for enterprise networks, which may not be pertinent to home networks. Leveraging blockchain or Artificial Intelligence (AI) technologies could enable the vetting of IoCs within a centralized system, categorizing them based on customer relevance and subsequently applying them automatically to the various types of UTM systems, e.g., home systems vs. enterprise systems.

Another limitation of home network UTM systems is the difficulty in effectively blocking hashes due to constrained UTM device resources. IoCs also encompass hashes for known malicious files. By blocking these hashes, we can effectively halt the propagation of malware. However, inspecting hashes may necessitate additional processing, as the UTM system must communicate with a database of blacklisted hashes and cross-reference them before taking action. This database can either be local or accessed via the Internet; for instance, Palo Alto Networks offers a service named "WildFire" that performs similar functions. The challenge arises in detecting unknown hash files that are malicious but not yet cataloged in any database, whether open-source or COTS. Addressing this requires the development of an anomaly detection system capable of identifying zero-day hashes. Such a system may utilize certain characteristics to identify potential threats before classifying them as malicious.

In summary, future research should develop open-source threat intelligence platforms that are equivalent to commercial feeds. Also, techniques for inspecting hashes without overloading home network UTM devices should be developed. Overall, validating threat feeds poses a significant challenge for the threat intelligence field [64], and addressing these challenges should be the focus of future research, which may proceed in the following steps:

- Analyze threat feeds from various sources using STIX/TAXII.
- Develop solutions for validating threat feeds, such as solutions based on adaptations of blockchain mechanisms [65], [66], [67], [68] or machine learning techniques [18], [19].
- Verify and evaluate these solutions to gauge the accuracy and suitability of the developed solution.

3) IDS OPTIMIZATION

Our evaluation indicates that IDS is the single most resource-intensive process in SUTMS. Optimizing IDS signatures is an integral part of SUTMS. We have only enabled the signatures of applications that were discovered via NTOP flow detection so as to reduce any unnecessary processing. However, services utilized by home networks are dynamic and can change frequently [69]. There will also be services left uncategorized due to the lack of pattern detection. Application detection engines require constant updates, and there is a need for open-source application detection engines that process, validate, and categorize new applications as they emerge. The integration of such application detection engines with UTM systems will improve UTM device efficiency and help defend against zero-day attacks.

Another optimization related to IDS is to integrate data mining algorithms, such as Apriori [70], into the SUTMS IDS engine to reduce false positives. More specifically, since IDS generates numerous alerts, it can be helpful to reduce false positives by optimizing (tuning) the SUTMS IDS engine with the Apriori algorithm [70], [71] so as to identify the critical alerts. If external logging is employed, only the critical log messages that meet the Apriori criteria would then be sent to a third-party log collector. Examining the tradeoff between the increased computational resource consumption for running Apriori on the local UTM device and the reduction of the false positives (and the resulting implications, e.g., reduced logging traffic) is an interesting direction for future research.

APPENDIX A
IoC TO FIREWALL RULE CONVERSION

```
#!/bin/sh
i=0
while [ $i -le 1 ];
do
  curl -kv -o 'ioc_updates' -H 'Content-Type:
        application/json' 'https
        ://192.168.200.160:8080/api/v1/
        intelligence' -d '{"token":"797
        d09613cbf91bd6d48aadb8bc1a66e", "query":"
        confidence>5
0 AND severity=very-high AND date_last>-1d", "
     type":"csv", "size":100}'
  perl -lne '/\b[0-9.]{7,15}\b/ && print $&'
        ioc_updates >blacklisted_IP
"$i";
i=$(($i+1));
done
```

LISTING 1. Program for IoC to iptables rule conversion.

In Listing 1, 192.168.200.160 is the IP address of the IoC server hosted in the LAN, which can be changed to IP addresses of the Internet IoC provider. Moreover, 8080 is a port used to connect to the IoC server, while "797d09613cbf91bd6d48aadb8bc1a66e" is a randomly generated token for downloading the IoCs, 100 is the number of downloaded IoCs, and "[0–9],[7.15]" is a pattern matching field to extract IP addresses from the IoC file.

REFERENCES

[1] J. Yang and L. Sun, "A comprehensive survey of security issues of smart home system: 'Spear' and 'Shields,' theory and practice," *IEEE Access*, vol. 10, pp. 124167–124192, 2022.

[2] *Threat Spotlight: Coronavirus-Related Phishing*. Accessed: Jan. 1, 2023. [Online]. Available: https://blog.barracuda.com/2020/03/26/threat-spotlight-coronavirus-related-phishing

[3] L. VanHulle, "Cyberattacks are a remote possibility: Dealerships can fight work-from-home vulnerabilities," *Automot. News*, vol. 94, no. 6933, pp. 1–14, May 11, 2020.

[4] A. Siddiqui, B. Rimal, M. Reisslein, and Y. Wang, "Survey on unified threat management (UTM) systems for home networks," *IEEE Commun. Survey Tuts.*, Mar. 2024, doi: 10.1109/COMST.2024.3382470.

[5] A. Siddiqui, B. P. Rimal, M. Reisslein, and D. Gc. (2024). *SUTMS—Unified Threat Management System for Home Networks Repository*. Accessed: Mar. 14, 2024. [Online]. Available: https://github.com/asif18577/SUTMS

[6] O. Karahan and B. Kaya, "Raspberry Pi firewall and intrusion detection system," *J. Intell. Syst., Theory Appl.*, vol. 3, no. 2, pp. 21–24, Dec. 2020.

[7] J. E. C. de la Cruz, C. A. R. Goyzueta, and C. D. Cahuana, "Intrusion detection and prevention system for production supervision in small businesses based on Raspberry Pi and Snort," in *Proc. IEEE 28th Int. Conf. Electron., Electr. Eng. Comput. (INTERCON)*, Sep. 2020, pp. 1–4.

[8] R. Alasmari and A. A. Alhogail, "Protecting smart-home IoT devices from MQTT attacks: An empirical study of ML-based IDS," *IEEE Access*, vol. 12, pp. 25993–26004, 2024.

[9] D. Rani, N. S. Gill, P. Gulia, F. Arena, and G. Pau, "Design of an intrusion detection model for IoT-enabled smart home," *IEEE Access*, vol. 11, pp. 52509–52526, 2023.

[10] V. Visoottiviseth, G. Chutaporn, S. Kungvanruttana, and J. Paisarnduangjan, "PITI: Protecting Internet of Things via intrusion detection system on Raspberry Pi," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2020, pp. 75–80.

[11] N. M. Allifah and I. A. Zualkernan, "Ranking security of IoT-based smart home consumer devices," *IEEE Access*, vol. 10, pp. 18352–18369, 2022.

[12] H. Jmila, G. Blanc, M. R. Shahid, and M. Lazrag, "A survey of smart home IoT device classification using machine learning-based network traffic analysis," *IEEE Access*, vol. 10, pp. 97117–97141, 2022.

[13] Q. I. Sarhan, "Systematic survey on smart home safety and security systems using the Arduino platform," *IEEE Access*, vol. 8, pp. 128362–128384, 2020.

[14] S. Zaman, K. Alhazmi, M. A. Aseeri, M. R. Ahmed, R. T. Khan, M. S. Kaiser, and M. Mahmud, "Security threats and artificial intelligence based countermeasures for Internet of Things networks: A comprehensive survey," *IEEE Access*, vol. 9, pp. 94668–94690, 2021.

[15] S. S. Tirumala, N. Nepal, and S. K. Ray, "Raspberry Pi-based intelligent cyber defense systems for SMEs: An exploratory study," in *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Cham, Switzerland: Springer, 2022, pp. 3–14.

[16] M. A. F. M. Fauzi and L. M. Abdullah, "Malicious/phishing URL detection system in a network with Raspberry Pi (NETBITS)," *Int. J. Perceptive Cognit. Comp.*, vol. 8, no. 2, pp. 30–36, 2022.

[17] G. A. J. Saskara, I. M. E. Listarta, G. S. Santyadiputra, P. B. Megawanta, and P. A. W. A. P. Giri, "Performance of Kismet wireless intrusion detection system on Raspberry Pi," in *Proc. 4th Int. Conf. Vocational Educ. Technol.*, 2022, pp. 1–9.

[18] H. B. Abdullahi, "Developing intelligent cyber threat detection systems through advanced data analytics," *Int. J. Innov. Sci. Res. Techn.*, vol. 9, no. 2, pp. 456–465, Feb. 2024.

[19] H. Liao, M. Z. Murah, M. K. Hasan, A. H. M. Aman, J. Fang, X. Hu, and A. U. R. Khan, "A survey of deep learning technologies for intrusion detection in Internet of Things," *IEEE Access*, vol. 12, pp. 4745–4761, 2024.

[20] H. Siddharthan and D. Thangavel, "A novel framework approach for intrusion detection based on improved critical feature selection in Internet of Things networks," *Concurrency Comput., Pract. Exper.*, vol. 35, no. 1, Jan. 2023, Art. no. e7445.

[21] M. M. Alani, "BotStop: Packet-based efficient and explainable IoT botnet detection using machine learning," *Comput. Commun.*, vol. 193, pp. 53–62, Sep. 2022.

[22] AT&T Cybersecurity. (2024). *OTX: AT&T Alien Labs Open Threat Exchange*. Accessed: Jan. 18, 2024. [Online]. Available: https://cybersecurity.att.com/open-threat-exchange

[23] (2024). *Malware Information Sharing Platform & Threat Sharing*. Accessed: Jan. 18, 2024. [Online]. Available: https://www.misp-project.org/

[24] (2024). *Cyber Observable EXpression (CybOX)*. Accessed: Jan. 18, 2024. [Online]. Available: https://cyware.com/security-guides/cyber-threat-intelligence/what-is-cyber-threat-intelligence-sharing-and-why-should-you-care-cfdc

[25] M. Apoorva, R. Eswarawaka, and P. Reddy, "A latest comprehensive study on structured threat information expression (STIX) and trusted automated exchange of indicator information (TAXII)," in *Proc. 5th Int. Conf. Frontiers Intelligent Computing, Theory Appl.*, 2017, pp. 477–482.

[26] O. Krauss and K. Papesh, "Analysis of threat intelligence information exchange via the STIX standard," in *Proc. Int. Conf. Electr., Comput., Commun. Mechatronics Eng. (ICECCME)*, Nov. 2022, pp. 1–6.

[27] Raspberry Pi Ltd. (2021). *Raspberry Pi 4 Model Specifications*. Accessed: Jul. 15, 2022. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[28] J. Zhang and A. Moore, "Traffic trace artifacts due to monitoring via port mirroring," in *Proc. Workshop end-to-end Monitor. Techn. Services*, May 2007, pp. 1–8.

[29] (2022). *High Performance Network Monitoring Solutions Based on Open Source and Commodity Hardware*. Accessed: Aug. 8, 2022. [Online]. Available: https://www.ntop.org

[30] I. Vaccari, S. Narteni, M. Aiello, M. Mongelli, and E. Cambiaso, "Exploiting Internet of Things protocols for malicious data exfiltration activities," *IEEE Access*, vol. 9, pp. 104261–104280, 2021.

[31] (2022). *We're the Creators of the Elastic (ELK) Stack—Elasticsearch, Kibana, Beats, and Logstash. Securely and Reliably Search, Analyze, and Visualize Your Data in the Cloud or On-Prem*. Accessed: Sep. 21, 2022. [Online]. Available: https://www.elastic.co/

[32] (2021). *Kibana*. Accessed: Sep. 21, 2022. [Online]. Available: https://aws.amazon.com/opensearch-service/the-elk-stack/kibana

[33] W. Park and S. Ahn, "Performance comparison and detection analysis in Snort and Suricata environment," *Wireless Pers. Commun.*, vol. 94, no. 2, pp. 241–252, May 2017.

[34] (2022). *Suricata 6.0.4 Documentation*. Accessed: Jul. 4, 2022. [Online]. Available: https://suricata.readthedocs.io/en/suricata-6.0.4/configuration/suricata-yaml.html?highlight=thread

[35] Q. Hu, S.-Y. Yu, and M. R. Asghar, "Analysing performance issues of open-source intrusion detection systems in high-speed networks," *J. Inf. Secur. Appl.*, vol. 51, Apr. 2020, Art. no. 102426.

[36] C. Hoover, "Comparative study of Snort 3 and Suricata intrusion detection systems," M. S. thesis, Dept. Computer Science and Computer Eng., Univ. Arkansas, Fayetteville, AR, USA, 2022.

[37] A. A. E. Boukebous, M. I. Fettache, G. Bendiab, and S. Shiaeles, "A comparative analysis of Snort 3 and Suricata," in *Proc. IEEE IAS Global Conf. Emerg. Technol. (GlobConET)*, May 2023, pp. 1–6.

[38] (2024). *Suricata 6.0.4 Documentation: 8.7 Payload Keywords*. Accessed: May 10, 2024. [Online]. Available: https://docs.suricata.io/en/latest/rules/payload-keywords.html

[39] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, *Address Allocation for Private Internets*, document IETF RFC 1918, 1996.

[40] A. Bansal and P. Goel, "Simulation and analysis of network address translation (NAT) & port address translation (PAT) techniques," *Int. J. Eng. Res. Appl.*, vol. 7, no. 7, pp. 50–56, Jul. 2017.

[41] (2022). *Webmin Management Tool*. Accessed: Jul. 15, 2022. [Online]. Available: https://www.webmin.com

[42] (2022). *Ubuntu IPtables Persistent Package*. Accessed: Jul. 15, 2022. [Online]. Available: https://packages.ubuntu.com/bionic/admin/iptables-persistent

[43] (2005). *Linux Packet Filtering and IPtables*. Accessed: Jul. 15, 2022. [Online]. Available: https://www.linuxtopia.org/Linux_Firewall_iptables/x4983.html

[44] Mitre Corp. (2012). *Standardizing Cyber Threat Intelligence Information With the Structured Threat Information EXpression (STIX)*. Accessed: Jul. 7, 2022. [Online]. Available: https://www.mitre.org/sites/default/files/publications/stix.pdf

[45] (2012). *Anomali Threat Feeds*. Accessed: Jul. 7, 2022. [Online]. Available: https://www.anomali.com/resources/what-are-stix-taxii

[46] (2015). *Cybersecurity and Infrastructure Security Agency (CISA)*. Accessed: Jul. 8, 2022. [Online]. Available: https://www.cisa.gov/ais

[47] J. Moy, *OSPF Version 2*, document RFC 2328, 1997.

[48] D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch, and R. White, *Cisco's Enhanced Interior Gateway Routing Protocol*, document RFC 7868, 2016.

[49] K. Lougheed and Y. Rekhter, *Border Gateway Protocol (BGP)*, document RFC 163, 1990.

[50] R. Gerhards, *The Syslog Protocol*, document RFC 5424, 2009.

[51] (2023). *What is XDR?*. Accessed: May 6, 2024. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/what-is-xdr/

[52] Splunk a CISCO Comany. (2023). *Splunk Enterprise Security*. Accessed: Jun. 2, 2023. [Online]. Available: https://www.splunk.com/en_us/products/enterprise-security.html

[53] Elastic—The Search Analytics Company. (2023). *Elastic Security for SIEM & Security Analytics*. Accessed: Jun. 2, 2023. [Online]. Available: https://www.elastic.co/security/siem

[54] AT&T Business. (2023). *AlienVault OSSIM—The World's Most Widely Used Open-Source SIEM*. Accessed: Jun. 2, 2023. [Online]. Available: https://cybersecurity.att.com/products/ossim

[55] (2017). *Canadian Institute for Cybersecurity*. Accessed: Sep. 22, 2022. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[56] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.

[57] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "A detailed analysis of the CICIDS2017 data set," in *Communications in Computer and Information Science*. Cham, Switzerland: Springer, 2019, pp. 172–188.

[58] Cockpit. (2022). *Cockpit Makes it Easy to Administer Your Linux Servers via a Web Browser*. Accessed: Sep. 21, 2022. [Online]. Available: https://cockpit-project.org

[59] Netdata: Monitor. Troubleshooting. (2022). *Slash Your Time to Detect, Troubleshoot and Resolve Infrastructure Performance Anomalies*. Accessed: Sep. 21, 2022. [Online]. Available: https://www.netdata.cloud/

[60] H. A. Damanik and M. Anggraeni, "Implementation scheme SLA and network availability mechanism for customer service provider," *Jurnal Penelitian Pos dan Informatika*, vol. 10, no. 2, pp. 125–144, Dec. 2020.

[61] F5 Netw. (2021). *Half the World's Malware is Now Encrypted*. Accessed: Dec. 5, 2022. [Online]. Available: https://www.f5.com/company/blog/half-the-world-s-malware-is-now-encrypted

[62] Crowdstrike. (2022). *Threat Intelligence Products*. Accessed: Dec. 5, 2022. [Online]. Available: https://www.crowdstrike.com/products/threat-intelligence/

[63] ThreatConnect. (2022). *Smarter Security, Maximum Impact*. Accessed: Dec. 5, 2022. [Online]. Available: https://threatconnect.com/

[64] E. Aljbour, A. Dabit, M. Al-Fayoumi, and Q. A. Al-Haija, "UNI-CERT: A unified computer emergency response teams model for malware information sharing platform," in *Proc. IEEE 5th Int. Conf. Power, Intell. Comput. Syst. (ICPICS)*, Jul. 2023, pp. 404–410.

[65] L. Almuqren, K. Mahmood, S. S. Aljameel, A. S. Salama, G. P. Mohammed, and A. A. Alneil, "Blockchain-assisted secure smart home network using gradient-based optimizer with hybrid deep learning model," *IEEE Access*, vol. 11, pp. 86999–87008, 2023.

[66] F. F. Alruwaili, M. A. Alohali, N. Aljaffan, A. A. Alhashmi, A. Mahmud, and M. Assiri, "A decentralized approach to smart home security: Blockchain with red-tailed hawk-enabled deep learning," *IEEE Access*, vol. 12, pp. 14146–14156, 2024.

[67] A. El-Kosairy, N. Abdelbaki, and H. Aslan, "A survey on cyber threat intelligence sharing based on blockchain," *Adv. Comput. Intell.*, vol. 3, no. 3, Jun. 2023, Art. no. 10.

[68] K.-A. Provatas, I. Tzannetos, and V. Vescoukis, "Standards-based cyber threat intelligence sharing using private blockchains," in *Proc. Ann. Comput. Sci. Inf. Syst.*, Sep. 2023, pp. 649–656.

[69] J. L. Hernández-Ramos, S. N. Matheu, A. Feraudo, G. Baldini, J. B. Bernabe, P. Yadav, A. Skarmeta, and P. Bellavista, "Defining the behavior of IoT devices through the MUD standard: Review, challenges, and research directions," *IEEE Access*, vol. 9, pp. 126265–126285, 2021.

[70] E. Saboori, S. Parsazad, and Y. Sanatkhani, "Automatic firewall rules generator for anomaly detection systems with apriori algorithm," in *Proc. 3rd Int. Conf. Adv. Comput. Theory Engineering(ICACTE)*, vol. 6, Aug. 2010, pp. V657–V660.

[71] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. Int. Conf. Very Large Data Bases*, vol. 1215, 1994, pp. 487–499.

ASIF SIDDIQUI received the Ph.D. degree in cyber defense from Dakota State University, Madison, SD, USA. He is currently an Adjunct Faculty Member of The Beacon College of Computer and Cyber Sciences, Dakota State University. Previously, he was a Lecturer with the Institute of Applied Technology, Abu Dhabi, United Arab Emirates. He is actively engaged in research and brings years of experience in designing and implementing enterprise on-premises and cloud security solutions, including unified threat management (UTM) systems, threat intelligence platforms, and monitoring systems.

BHASKAR P. RIMAL (Senior Member, IEEE) is currently an Assistant Professor with the Department of Computer Science, University of Idaho, Moscow, USA. He was an Assistant Professor with Dakota State University, Madison, SD, USA. He was a Visiting Assistant Professor and an Academic Specialist with the Department of Computer Science, Tennessee Tech University, Cookeville, TN, USA. He was a Visiting Scholar with the Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. He is a Senior Member of ACM. He serves as a Technical Editor for *IEEE Network* and an Associate Editor for IEEE ACCESS.

MARTIN REISSLEIN (Fellow, IEEE) received the Ph.D. degree in systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 1998. He is currently a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University (ASU), Tempe, AZ, USA. He is currently an Associate Editor of IEEE ACCESS and the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. He also serves as an Area Editor for Optical Communication of the *IEEE Communications Surveys & Tutorials* and a Co-Editor-in-Chief for *Optical Switching and Networking*.

DEEPAK GC (Senior Member, IEEE) received the B.Eng. degree in electronics and telecommunication from Pokhara University, Nepal, the M.Eng. degree in computer engineering from Jeonbuk National University, South Korea, and the Ph.D. degree from Lancaster University, U.K., in 2017. He is currently a Senior Lecturer with the School of Computer Science and Mathematics and the Course Leader of Cyber Security and Digital Forensics (B.Sc.) with Kingston University London. Previously, he was a Research Fellow with Liverpool John Moores University, U.K, under EU H2020 Project Wi-5 and Kingston University London under EPSRC GCRF Project DARE. His research interests include radio access technologies, cognitive radio, 5G/6G, physical layer security, the Internet of Things, cyber security, and public safety communications.

YONG WANG received the B.S. and M.S.E. degrees in computer science from Wuhan University, China, in 1995 and 1998, respectively, and the Ph.D. degree in computer science from the University of Nebraska-Lincoln, in 2007. He is currently a Full Professor with Dakota State University. He has published more than 70 peer-reviewed papers in prestigious journals and conferences. His research interests include security and privacy issues in the Internet of Things, mobile devices, cloud computing, cyberinfrastructure, optical networks, wireless networks, and social networks. He received five awards from the National Science Foundation (NSF) and one award from the National Centers for Academic Excellence in Cybersecurity (NCAE-C).