Federated Agentless Endpoint Detection Using Behavioral and Characteristic Modeling

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Hansaka Angel Dias Edirisinghe Kodituwakku

December 2021

Copyright © 2021 by Hansaka Angel Dias Edirisinghe Kodituwakku.

All rights reserved.

Acknowledgements

I would like to thank my advisor Dr. Jens Gregor for his excellent supervision and guidance. I would like to thank my parents for their support and belief in me. I would like to thank Dr. Hamparsum Bozdogan for his excellent teaching and guidance. I would like to thank Alex Keller and Eboni Thamavong for their invaluable advice and for sharing their real-world experience. I would like to thank Chase, my four-legged man's best friend for being there day and night unconditionally.

I would like to thank Cisco ASIG for providing computer equipment and software licenses that played a critical part in the development of this work.

Abstract

Due to the lack of endpoint verification and modern security features built into the current data link, network, and transport layers of the TCP/IP stack, spoofing, lateral movement and data exfiltration are difficult to be detected. Often the security aspect of networks gets managed reactively instead of providing proactive protection. Security operations teams struggle to keep up with the ever-increasing number of devices and attacks daily. Modern zero trust security policies demand the verification of each entity in the network without implied trust. Data for policy enforcement and incident response are usually collected at the backbone. They are inadequate to identify endpoints at the edge network. Incident response teams require data that are timestamped and reliably attributed to each individual endpoint, for forensic analysis. With the current state of dissociated data collected from networks using different tools, it is challenging to correlate the necessary data to find origin and propagation of attacks within the network. Critical indicators of compromise may go undetected due to the drawbacks of current data collection systems leaving endpoints vulnerable to attacks. Proliferation of distributed organizations demand distributed and federated security solutions. Without robust data collection systems that are capable of transcending architectural and computational challenges, it is becoming increasingly difficult to provide endpoint protection at scale. This research focuses on reliable agentless endpoint detection and traffic attribution in federated networks using behavioral and characteristic modeling for incident response.

Table of Contents

Chapter 1: Introduction 1

1.1 Motivation 1

1.2 State-of-the-Art 3

1.2.1 Publicly Available Datasets 5

1.2.2 Data Collection Techniques 9

1.2.3 Endpoint Attacks and Protection 14

1.3 Incident Response 24

1.4 Contributions 28

Chapter 2: InDepth Data Collection System 30

2.1 InDepth System Architecture 30

2.2 InDepth Cyber Range 38

2.3 Comparison with Backbone Data Collection 41

Chapter 3: InMesh Endpoint Detection 47

3.1 Endpoint Data Collection 47

3.2 Endpoint Detection Technique 51

3.3 Distance Measurement 53

3.4 Scenarios 61

Chapter 4: Conclusion 65

Bibliography 67

Appendix 76

Vita 82

List of Tables

Table 1.1: Commercially available EDR systems and their capabilities…….22

Table 2.1: Examples of attacks that uses unique features of InDepth……42

Table 3.1: Features used for snapshot distance measurement……..55

List of Figures

Figure 1.1: Benchmark datasets and their number of citations. 8

Figure 1.2: Number of KDD'99 citations for per year. 8

Figure 1.3: Switching and flow data generation at the switch. 10

Figure 1.4: Agentless network and endpoint data collection. 18

Figure 1.5: Perimeter and endpoint security solutions..... 23

Figure 1.6: Forensic study workflow [5] 25

Figure 1.7: Attack visualization before port filtering [5]. 26

Figure 1.8: Attack visualization after port filtering [5]. 26

Figure 1.9: 2019 WRCCDC network topology 27

Figure 2.1: InDepth system architecture. 31

Figure 2.2: Overview of node placement and communication. 33

Figure 2.3: Active and passive feature categorization. 36

Figure 2.4: InDepth cyber range. 40

Figure 2.5: Centralized vs. distributed DCS using 2019 WRCCDC topology. 44

Figure 3.1: Federated logical dataflow of InMesh. 49

Figure 3.2: Information complexity score vs. number of features considered. 54

Figure 3.3: Partial InDepth cyber range used for use-cases. 57

Figure 3.4: Histogram of average d_E of snapshots of the same endpoint. 58

Figure 3.5: Histogram of average d_E of snapshots of two different endpoints..... 58

Figure 3.6: InMesh endpoint detection algorithm. 60

Chapter 1: Introduction

In this chapter, we discuss the motivation for the research, the state of the art of endpoint detection and response systems (EDR), and how this work addresses the drawbacks of the current systems. Then we present the results of the incident response study we conducted on the Western Regional Collegiate Cyber Defense Competition (WRCCDC) and the learning outcomes. In Chapter 2, we discuss the novel data collection mechanism to collect both endpoint and network data. In Chapter 3, we use the collected data to build behavioral models that enable the detection of endpoints at the edge network. Chapter 4 summarizes the learning outcomes and provides closing remarks.

1.1 Motivation

Even though computer networks still use the same Transmission Control Protocol over Internet Protocol (TCP/IP) set of protocols, traffic behaviors and cyber-attacks have significantly evolved since then and continue to do so. Most popular benchmark datasets that are being used today have been introduced more than two decades ago [1] and most networks collect data only from their backbone network and only have perimeter security systems (discussed in Section 1.2.3). This leaves endpoints vulnerable to attacks, and compromises may remain undetected. Endpoint detection and response (EDR) systems are experiencing a rapid growth [2] along with the proliferation of mobile devices and Internet of things (IoT) [3] and the increase of endpoint attacks [4]. EDR systems that use agents may not be applicable in all networks and agentless systems face challenges due to the lack of data collection systems (DCSs) and endpoint modeling techniques.

The motivation for this research stems from analysis of 2019 WRCCDC [5] data. This is an annual college level blue-team, red-team cyber competition to test participant's network defense skills. Packet captures [6] from this competition are publicly available. Packet captures are files containing the contents of complete network transactions which

are also known as PCAP files. We analyzed these packet captures as a forensic study exercise to identify the network attacks and compromised endpoints from the point of view of the participating Stanford University team. Packet captures from cyber competitions provide data that are not synthetic or simulated hence provide extremely valuable information related to the network transactions. The data include real human interactions and up-to-date cyber-attacks that widely used benchmark tests often lack. The ground truth was gathered from the competition rules as well as from the competition participants, regarding the operating systems (OS), services and topology.

Learning outcomes from the analysis include the inadequacy of using IP address of the TCP/IP stack for endpoint detection and data attribution. We also identified architectural limitations of the state-of-the-art of packet capturing and the lack of statistical methods to differentiate endpoints using their behaviors and characteristics that negatively impact incident response. Since backbone packet captures do not capture all network traces from all endpoints in the network, they tend to omit critical data to identify certain indicators of compromise (IOC) [5].

Reliable and complete forensic investigations require a comprehensive set of features collected from modern networks and endpoints [7]. We studied the literature extensively including publicly available datasets, existing DCSs as well as commercial EDR systems. As we discuss next, existing data and tools do not offer the granularity and completeness needed for behavioral and characteristic modeling for federated agentless detection of endpoints.

Contributions include 2019 WRCCDC packet capture forensic analysis, development of the InDepth DCS and cyber range, and development of the InMesh EDR system. The work has been documented in three papers, namely:

- H. A. D. E. Kodituwakku, A. Keller and J. Gregor, "InSight2: A modular visual analysis platform for network situational awareness in large-scale networks," *Electronics,* vol. 9.10, pp. 1747-1749, 2020. [5]
- H. A. D. E. Kodituwakku, A. K. T. Shimeall and J. Gregor, "InDepth: A novel distributed and federated network and endpoint data collection system," *Digital Threats: Research and Practice*. Submitted May 2021. [8]
- H. A. D. E. Kodituwakku, H. Bozdogan, T. Shimeall, and J. Gregor, "InMesh: An agentless system for endpoint detection and response," *Digital Threats: Research and Practice*. Submitted November 2021. [9]

Analysis of the 2019 WRCCDC data provided the understanding and requirements expected for a robust next-generation DCS as well as an endpoint behavioral and characteristic comparison technique to differentiate devices using their traffic activity and system attributes without using software agents. InDepth was developed based on these requirements to collect the necessary data from production networks. InMesh was developed and implemented on InDepth that uses the comparison technique developed for endpoint detection. As a result, endpoints can be tracked more reliably, and their traffic can be attributed more accurately without relying on traditional IP or MAC addresses or software agents.

1.2 State-of-the-Art

Security analysts need data for training and post-breach forensic studies [10]. Cybersecurity researchers require data to develop and test network threat detection models and algorithms [11]. Both groups depend on access to an extensive set of features that cover network traces, endpoint telemetry data, and contextual information [12]. Studies discussed in Section 1.2.1 show that widely used, publicly available datasets contain outdated traffic scenarios, features that do not align with modern networks, statistical anomalies, and simulation artifacts. Due to architectural limitations and

computational challenges, even state-of-the-art data collection systems rely on workarounds that lead to incomplete and disassociated data.

The accuracies of the underlying models and algorithms depend on the quality of the datasets. Lack of comprehensive DCSs that can collect both network and host features leads to incomplete data collected about endpoint activity and characteristics [13] [14]. Data collected using existing mechanisms may miss critical information needed to detect and analyze early stages of an attack such as reconnaissance and lateral movement. Shiravi et al [15] show that total interaction capture is needed including inter- and intra-Virtual Local Area Network (VLAN) transactions. Even state-of-the-art DCSs typically use workarounds to compensate for architectural and computational challenges which leads to incomplete and disassociated data (discussed in Section 1.2.2).

The TCP/IP stack has no built-in endpoint detection capability making it difficult to attribute network transactions to the correct endpoints. Network traces and characteristic logs must be collected and attributed to the correct endpoint for an effective cyber incident response. Agent-based EDR systems use endpoint software agents to uniquely identify the endpoint as well as collect system information such as host name, open ports, Operating System (OS) and network activity. These agent-based systems encounter practical limitations, such as unsupported operating systems, unmanaged devices, resource limitations, read-only memory, and privacy concerns, and can thus not easily be implemented on all devices in a network. Agentless EDR systems, on the other hand, allow ease of deployment and maintenance while also offering increased resilience to attacks not to mention privacy, since they are separate from the endpoint. They can be used standalone or in addition to agent-based EDR systems to cover all endpoints in a network. Current agentless EDR systems rely on addresses from the network packets for data attribution, which may not be reliable [16].

1.2.1 Publicly Available Datasets

According to McHugh [17], widely used benchmark datasets such as DARPA'98 [18], KDD'99 [1], and NSL-KDD [19] are outdated and do not represent modern network traffic. They also lack information on how the data were collected including network topology, functional descriptions of each subnet, firewall rules, endpoint OS, service versions, attack details, and validation of the types of network traffic they are said to represent. Shiravi et al [15] pointed out that it is necessary to move away from static one-time datasets to generating data more dynamically because network behaviors and patterns change, and intrusions evolve.

DARPA’98 and KDD’99 datasets suffer from simulation artifacts and inclusion of statistical anomalies that machine learning models and algorithms adapt themselves to, producing unusually high accuracies that cannot be replicated from real-world data [20]. UGR’16 dataset labels records as benign or malicious [21]. This may lead machine learning models to use the attributes of a single flow to determine whether a network has been compromised. However, in practice, individual flow records do not necessarily represent an attack. Security analysts perform forensic investigations by looking for victim vulnerabilities, attacker motivation, and a chain of actions that lead to the compromise. The source address contained in the flow record may also not be the attacker’s real source identifier since source addresses can be spoofed and traffic can be bounced off of unsuspecting endpoints in the network.

Figure 1.1 provides a histogram of the number of citations for the benchmark datasets included in the survey by Khraisat et al [22] as of June 1, 2021. The number of citations was obtained using Google Scholar. KDD’99 is seen to have more citations than all the other data sets combined. As shown in Figure 1.2, the number of articles that cite KDD’99 has steadily increased every year since its introduction.

DARPA‘98 and its derivatives KDD’99 and NSL-KDD were created at the MIT Lincoln Lab. DARPA‘98 [18] contains synthetic tcpdump data of seven weeks of simulation with various attacks injected. KDD’99 [1] contains four attack types and includes 41 features. NSL-KDD eliminated some (but not all) problems associated with the KDD’99 dataset such as the duplicated records [19]. DARPA’99 [23] contains synthetic tcpdump traffic and endpoint data collected at two sensors, one inside the network and one outside the network. This dataset is not as widely used as the KDD’99 dataset.

Tavallaee et al [19] analyzed KDD’99 and found that 78% of records in the training set and 76% of records in the testing set were duplicated which biases detection algorithms towards the more frequent records. They showed that less frequent attacks may go undetected which could be harmful. Kayacik et al [24] reported that 98% of the training data consists of just three classes, namely, Normal and Neptune or Smurf DoS attacks, which lead to high detection rates and low false positives for those classes. MacHugh et al [17] identified that KDD’99 does not include enough information on how the benign background data was generated. The overall distribution of different attacks was also shown to be biased with denial of service (DoS) attacks at over 71% being most prevalent [19]. Due to their large footprint, the DoS attacks are easily detected using simple statistical methods. Simulation artifacts bias the detection results. Indeed, the extremely high accuracy reported in the literature for the KDD’99 data may be attributed to simulation artifacts and statistical anomalies unique to the attack data [19] [20]. For example, some attacks can be identified by observing the time-to-live values present only in malicious traffic, anomalies in the TCP window size value and specific IP addresses unique to the attackers. Synthetic datasets such as DARPA'98 and its derivatives are not suitable for security analyst training and forensic analysis since they do not represent real-world traffic.

KDD’99 is said to model network traffic for US Air Force bases, but support for this claim has not been published [17]. If so, the data may be specialized to the point of not representing commodity networks. Another potential issue is that the single-threaded

nature of the tcpdump tool renders it susceptible to packet drops [25], yet this aspect has not been studied. The data furthermore contains features that cannot be obtained from real-world networks as they require the contents of a network connection to be observed as cleartext, e.g., the number of login attempts, login status and attempts at privilege escalation. Modern encrypted network protocols such as SSH [26] make it impossible to extract cleartext from network traces. While specialized software agents running on each device can observe login attempts and export the data for decision making, firewall rules and readily available system software can easily prevent these types of attacks without sophisticated algorithms having to be developed. Attacks such as Smurf [27] were furthermore patched at the system level many years ago and do not occur in practice anymore.

Each KDD'99 record is labeled either as normal traffic or as one of four types of attacks [17]. Detailed information about the attacks is missing such as OS and service version of the victim devices, and penetration testing tools and parameters used by the attacker. Labeling a single transaction as an attack is not ideal.

Furthermore, a single packet that causes buffer overflow in the target system may not indicate an attack, and probing should be considered a precursor to a complex attack rather than an attack on its own.

More recent datasets have attempted to improve upon the above-mentioned shortcomings. CAIDA [28] provides anonymized backbone network traces upon request, but the data is not labeled and therefore has limited applicability. UNSW-NB15 [29] contains more up-to-date network scenarios but used a proprietary tool to generate the data artificially.

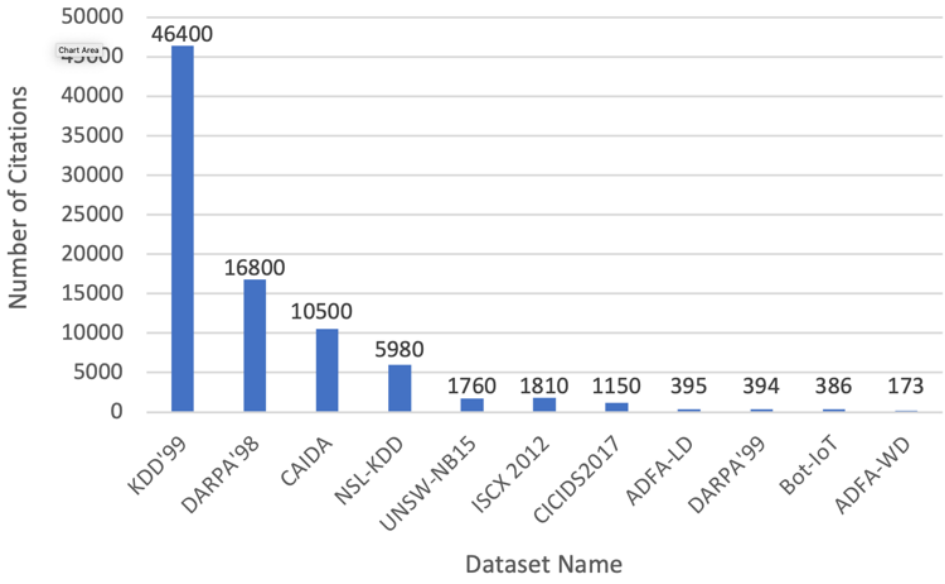


Figure 1.1: Benchmark datasets and their number of citations [8].

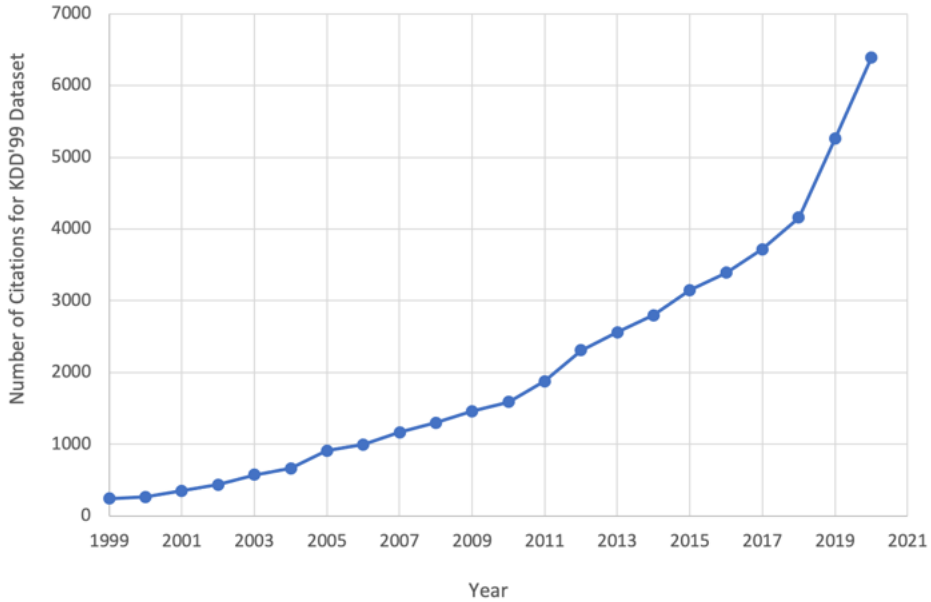


Figure 1.2: Number of KDD’99 citations for per year [8].

Studies have shown that UNSW-NB15 attributes are more efficient and provide more reasonable accuracies when machine learning models are trained, compared to unusually high accuracies provided by KDD'99 [30]. ISCX 2012 [31] contains 2 million records with 2% of them representing attack, namely, brute-force SSH, infiltration, HTTP DoS, and (Distributed Denial of Service) DDoS. This dataset covers seven days of network activity. CIC-IDS2017 [32] contains synthetic data of 25 user profiles that include HTTP, HTTPS, FTP, SSH, and email protocol usage. This dataset contains 5 days of activity that includes attack data based on brute-force FTP, brute-force SSH, DoS, Heartbleed, web attack, infiltration, iotnet and DDoS. In addition to flow data captured from the main switch, CIC-IDS2017 also includes memory dumps and system calls from all victim machines captured using software agents. ADFA-LD and ADFA-WD have 7 and 13 class labels respectively [33]. Bot-IoT [34] dataset focuses on botnet activity in Internet of Things (IoT) networks.

1.2.2 Data Collection Techniques

Flow data from network packets can be either generated at the networking gear or using dedicated hardware [35]. Modern switches and routers perform switching and routing functions on hardware using an application-specific integrated circuit (ASIC) which is very efficient. As illustrated by Figure 1.3, flow generation on switches and routers is performed by software using RAM and CPU which are limited.

A certain amount of memory proportional to the throughput of the network is required to cache the flow records, and the CPU must process and flush expired flows. Network hardware has limited processing capacity and can easily get overwhelmed in high-volume networks. Most network hardware also uses its processing resources to perform firewalling, Quality of Service (QoS), VPN functions. Further resource sharing can potentially hinder overall performance of the network.

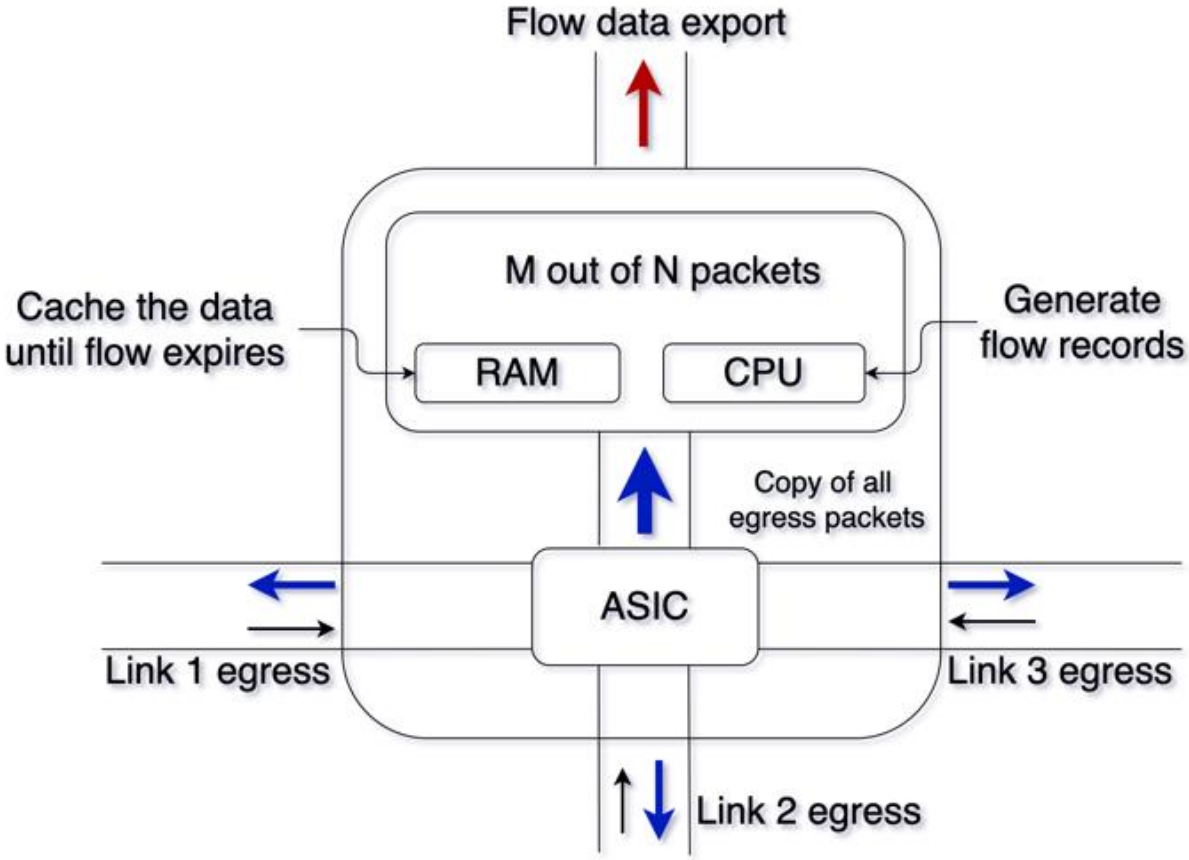


Figure 1.3: Switching and flow data generation at the switch.

Flow data generation using a dedicated external device requires access to raw packets of the network. This can be done either using a mirrored port from a capable switch or a network Terminal Access Point (TAP) device. Advantages of using a network TAP include capturing everything on the wire, including errors where network port mirroring may drop them. Since a network TAP sits on a network link, it does not use switch resources either. However, adding a network TAP on each port of a switch may not be economical.

On the other hand, port mirroring can be configured on one or more ingress or egress ports of an existing device which is more feasible and economical in most situations. For simplicity, we use port mirroring at each ingress port and address packet drop concerns by utilizing multi-threading. More powerful CPUs can thus be deployed for high throughput network segments. Port mirroring can be simple or encapsulated. The latter can be exported outside the subnet to a given IP address which helps central data collection but may not be available from all manufacturers. Simple port mirroring that mirrors traffic creates an exact copy of the traffic seen on all ports that have port mirroring enabled and is widely supported including Software Defined Networks (SDN). We utilize simple port mirroring and use the distributed node structure to finally aggregate all the data at a central location.

Distributed data collection at the subnet level provides numerous benefits that cannot be achieved using central DCS. It captures the data-link layer activity and includes endpoint information such as MAC addresses and VLAN IDs. For example, a distributed system can capture data from the initial stages of a data exfiltration attack as the malicious agent performs scanning for vulnerabilities and lateral movement within the subnet. In contrast, traditional data-collection systems may only capture the final stage of the exfiltration of the data to the command and control (C2) outside of the network. Furthermore, active probing becomes more reliable due to the endpoints being a single hop away, adding valuable endpoint attributes into the data collected. Device active probing harvests endpoint data which may not be possible centrally due to Network Address Translation

(NAT). The data consisting of network activity, endpoint characteristics and contextual information are better generated at the VLAN since it reduces the burden for processing and memory consumption by distributing the computation. It only exports compact meta-data saving the bandwidth compared to exporting packet captures. Due to the proliferation of cloud applications network traffic tend to bias packets going out to the Internet for web and cloud applications as opposed towards local network resources. Backbone DCS are efficient at mitigating threats coming from the Internet since they capture all traffic between local endpoint and internet. But these systems also struggle to process the extremely high throughput data rates at the backbone in modern networks and may miss critical activity happening at the subnet level. For example, critical stages of a compromise such as lateral movement do not involve traffic traversing the backbone.

Several flow standards exist. Developed by Cisco, NetFlow v5 [36] includes five basic flow features, namely, source IP address, destination IP address, source port, destination port, and protocol. Flow generation can be performed randomly instead of capturing every packet to overcome issues with packet drops. For example, the NetFlow exporter can be configured to sample M out of N packets in which case flow records are created just for those samples.

An alternative would be to use sFlow [37], which is specifically developed to generate sampled flow records. Both approaches trade performance for accuracy which can be a problem, especially when identifying slow attacks that span across a long period of time and leave small footprints on the network. NetFlow v9, which uses a proprietary protocol, collects more features from each flow and is the most widely used standard today. Juniper developed jFlow [38] to produce similar features, again using a proprietary protocol. The Internet Engineering Task Force (IETF) developed IPFIX [39] as open flow standard comparable to NetFlow v9. NetFlow and IPFIX are inherently unidirectional, the accuracy of deduplicated records depends on whether the data collection has been accounted for using ingress, egress, or both. Although initially developed by Cisco, NetFlow standard is being implemented by most mainstream vendors. However most white box solutions do

not support this standard with a few exceptions such as Open vSwitch [40]. It can be concluded that generating flow records using dedicated hardware provides the highest level of compatibility and uniformity. In addition, it provides the flexibility of tagging flows with endpoint data and other contextual information within a single process.

In practice, networks employ a Security Information and Event Management System (SIEM) (e.g. Splunk [41]) in addition to a flow collection system such as NetFlow generated on networking hardware. The purpose of a SIEM is to receive, collect, aggregate logs and events from many diverse sources that may send data in many different formats, often vendor-specific, and store the normalized records for use when needed. They can also help identify malicious events by correlating log data. Some of the data a SIEM parses are network firewall, IDS, antivirus, router logs, database logs, web application firewall, etc. Since SIEMs do not specifically collect endpoint characteristics, InDepth complements SIEM's log data and at the same time free up resources of the networking hardware for switching and routing activity.

Data collection is normally performed either outside the firewall, on the firewall, or on each endpoint. Data collected outside a firewall can be sent directly into traditional IDS/IPS solutions. Firewalls that have built-in IDS/IPS functionality benefit from on-firewall data collection. Data collected on each endpoint requires specialized software agents which are more suitable for HIDS applications. To our knowledge, a system does not exist that collects traffic and endpoint features simultaneously and enriches each record with contextual information from each VLAN without the need for software agents.

Techniques exist for collecting data centrally that can be performed using one or more collection points. The challenge with these solutions is that some network transactions may generate duplicate records while others do not, depending on the source and destination addresses.

Single threaded packet capture utilities such as libpcap and PFRING suffer from high packet losses when exposed to high data rates [42]. CPU single core speeds have practical limitations where it starts to provide diminishing returns in terms of power consumption per clock frequency whereas modern multi-core CPUs can have a high number of CPU cores. A multi-threaded implementation of libpcap called DiCAP [43] is aimed at preventing packet losses using parallel processing. Packet loss is more prone to occur in centralized DCS such as backbone data collection. Backbone DCS demands high bandwidth requirements and high CPU and memory requirements because all the network packets routed at the backbone are processed at the backbone router. In contrast, distributed DCS that collect data at the subnet level can utilize low-power IoT devices in place of high-power servers and commodity network hardware.

1.2.3 Endpoint Attacks and Protection

EDR is an emerging aspect of enterprise cybersecurity that provides visibility into the activity of each endpoint in the network to identify emerging threats and active attacks using endpoint modeling. Endpoint models are crucial for shortening response times for incident response and can help eliminate threats before damage is done. For EDR systems to be effective, all network traffic must be captured and attributed to the correct endpoint. This can be challenging, especially when the IP and MAC addresses are used for identification, since these addresses can be spoofed and sometimes are shared among multiple endpoints, especially in private networks.

Agent-based EDR systems attributes endpoint activity by capturing all endpoint activity from the device itself using the software agent. However, this requires dedicated agents to be developed that are compatible with the many OS and application programming interfaces (API) in use. Exacerbating matters, some devices are not capable of accommodating agent software. Internet of things (IoT) devices, for example, are low power devices designed to carry out a limited number of specific, pre-defined tasks. Endpoints that run firmware or an embedded OS, including but not limited to medical

devices, industrial control systems, smart TVs, voice-over-internet-protocol (VoIP) phones, webcams, and printers, do not allow third-party software to be installed which means the manufacturer would have to issue software with the agent embedded. Certain embedded systems furthermore run outdated and discontinued OS versions to remain compatible with other software; agent installation might create undesirable issues including an increase in the attack surface. The number of unmanaged devices in enterprise networks is currently growing by almost 31% a year [44]. In the near future, up to 90% of all devices are predicted to be unmanaged and unsecured [2]. EDR systems are expected to have an annual growth rate of more than 25% during the period of 2020-2025 [45].

Most agentless EDR systems rely on network traffic captured by a backbone DCS for endpoint modeling. They use addresses seen in the data for attribution. An endpoint IP address, which is used for Layer 3 routing, can be assigned and altered within the usable range for the particular subnet manually or by a dynamic host control protocol (DHCP) server. DHCP IP re-lease and re-use facilitates communication for transient devices since there are limited number of IP addresses within the LAN pool with zero configuration. The endpoint IP address can also be modified during network address translation (NAT). In contrast, the media access control (MAC) address was designed to be immutable while being unique to the network adapter of the endpoint and is used for Layer 2 switching. However, they too can be changed, namely, by users with higher system privileges. Frame MAC addresses are replaced automatically when packets are routed from one subnet to another. Source MAC is replaced with that of the forwarding interface of the router and the destination MAC with that of the destination interface of the gateway. For these reasons, use of addresses from the TCP/IP stack may trigger generation of new models for the same endpoint or collision with an existing model for another endpoint when addresses are altered.

When MAC address impersonation occurs, the MAC table (also known as the Content Addressable Memory (CAM) table) of the switch is altered, since it is designed to map

each MAC address to a device interface for switching. This is a Layer 2 to Layer 1 mapping. When the attacker sends packets with the source MAC address of the victim, the switch learns that it should forward the future packets intended for the victim to the attacker, since the MAC/CAM table now lists the MAC address of the attacker instead of the victim. To reduce MAC/CAM table instability in case the victim reclaims the entry, the attacker periodically sends spoofed frames. Successful impersonation renders the victim device unreachable with all packets switched to the attacker. This Layer 2 man-in-the-middle (MITM) attack only involves the LAN switch. It can also be performed at Layer 3 using spoofed gratuitous address resolution protocol (ARP) messages against endpoints. This is called ARP poisoning. The ARP table contains the IP to MAC address mapping which is a Layer 3 to Layer 2 mapping. In both cases subsequent data will be from the attacker rather than the victim.

IP address spoofing has limited legitimate use. One application (albeit legacy) involves certain Internet Service Providers (ISP) in mobile IP networks where roaming mobile devices have to use the IP address of the native ISP for billing purposes which is then spoofed by the ISP to provide Internet connectivity. However, RFC 2344 reverse tunneling eliminates the need for spoofing in this situation [46]. Another application involves virtual private network (VPN) providers who spoof the IP with that of the local IP address. This can also be eliminated using proper VPN routing implementation [47].

Examples of malicious IP address spoofing are numerous, including but not limited to avoiding detection and implication in cyber-crime investigations, masking botnet devices, preventing being blocked by a victim host or network, preventing compromised devices from sending alerts, and bypassing block-lists and devices such as firewalls. Typically, IP address spoofing is followed by an actual attack such as denial of service (DoS), sending spam, and phishing emails. MAC address spoofing is used to bypass weak authentication systems such as MAC address allow-listing and ARP poisoning for MITM attacks. Replies to this traffic do not reach the actual source, since they are spoofed and do not offer bidirectionality. Spoofing is thus mainly used to disrupt communications. A notable

exception is double spoofing [48] which manipulates TCP sequence numbers by exploiting a vulnerability in sequence number generators on some older systems. This can help gather some information about the target without revealing the attacker's actual IP address.

For endpoint detection to be effective when IP or MAC addresses are changed, the EDR system needs to model the endpoint using system attributes such as OS, active services and host keys, and behavior indicators such as average inbound and outbound promiscuity, traffic producer-consumer ratios (PCR) and top talkers. In this dissertation, we use such behavioral models along with temporal and spatial features to develop what we refer to as the InMesh endpoint detection algorithm. Effective and accurate endpoint modeling requires data collection at Layer 2 which includes inter-LAN communication as well as MAC address information from the traffic. To capture this data, a distributed sensor network is required. Figure 1.4 shows data collectors we call 'sensors' operating on the Layer 2 installed on each LAN or VLAN, and a coordinator we call 'core' operating at Layer 7 for coordination and storage. The green arrows indicate flow of sensor data with the red arrows showing network connections. The placement and dataflow are depicted using a simple two-subnet network but can be extrapolated into more complex topologies where the sensor nodes will remain at the LAN level while the core node will move up to the backbone.

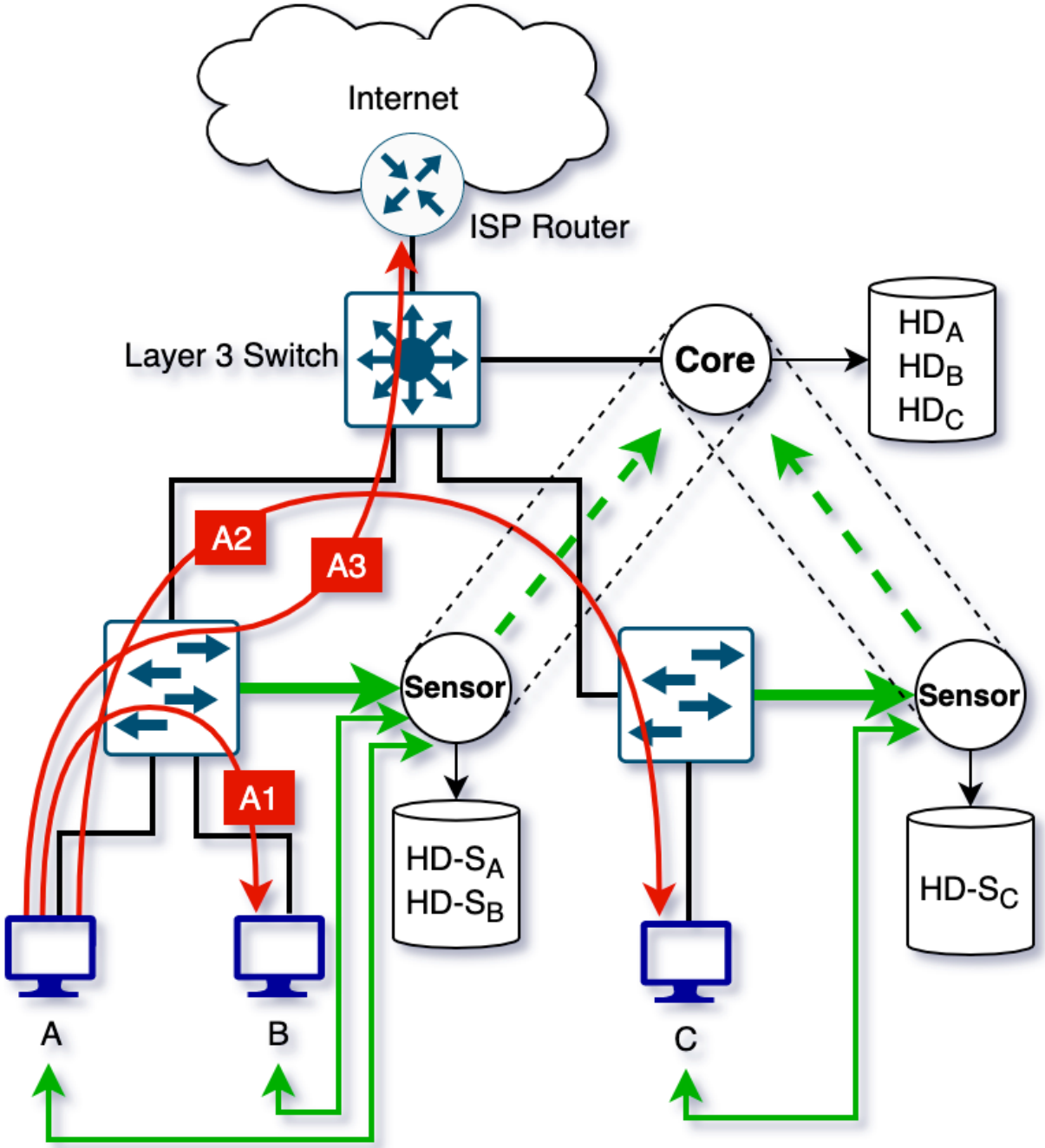


Figure 1.4: Agentless network and endpoint data collection.

Generally, attackers start with reconnaissance of the network topology and its endpoints. This step provides crucial information about the layout of the network, and, in turn, which intermediate devices must be compromised in order to reach the target endpoint. Reconnaissance and compromise are an iterative and incremental process. In Figure 1.4 the attacker's endpoint performs host discovery and vulnerability scanning on the same LAN using the A1 connection and other LANs using A2 connection. In this example, network data collection from the router or the Layer 3 switch does not reveal the A1 connection. More complex networks may have deeper hierarchies where the A2 connection would also not be detected.

Once vulnerable endpoints are found, they are exploited using payloads for known vulnerabilities, zero-day vectors, and brute-forcing login credentials. Credentials can also be stolen using social engineering, phishing attacks and MITM attacks. Endpoints can be compromised leveraging disgruntled employees, phishing attacks and physical access. Once an endpoint is compromised, privileges can be escalated to root or administrator exploiting various vulnerabilities, then preventative software such as antivirus, anti-malware and EDR agents can be disabled, malicious programs can be installed such as advanced persistent threats (APTs), a command-and-control (C2) channel can be established to communicate with the attacker, usually via a temporary virtual private server (VPS) hosted on the Internet on a predetermined domain name using domain name resolution (DNS) tunneling. In Figure 1.4, the attacker can use the A3 connection to establish a control channel with C2 on the internet. Perimeter protection can only detect an attack at this stage which may already be too late. More advanced threats can be fully autonomous, keeping silent until the target endpoint has been reached and data exfiltration begins. Transmission is typically to an attacker owned VPS via DNS tunneling, email, Bittorrent protocol, file transfer protocol (FTP) or secure shell (SSH), upload to a cloud service or paste site such as Pastebin or Github, or to physical media [49]. Attackers can compromise a significant number of devices in a network before current perimeter protection systems are able to identify them. Since perimeter protection systems look for known indicators and signatures, such compromises maybe missed entirely [50].

Agentless EDR systems have advantages, such as faster deployment and easier maintenance, because data is collected outside the endpoint. They also offer increased privacy, since there is no software running with file system and process level access that reports user activity. Most importantly, since they run on separate security hardened devices they cannot be accidentally or deliberately disabled or uninstalled during an endpoint compromise. Agentless EDR systems do not monitor system processes and files for malicious activities. But antivirus, anti-malware and data loss prevention systems (DLP) are dedicated software to monitor endpoint files and system resources. Antivirus aims to detect known, more established threats, such as Trojans, viruses, and worms. Anti-malware provide protection from new threats, such as polymorphic malware and malware delivered by zero-day exploits. DLP consists of a set of tools and processes to prevents data leaks, misuse and loss. In 2021, more than 95% of all web traffic is encrypted compared to less than 50% in 2014 [51]. Modern malware also uses traffic encryption to hide the contents. While it is possible to decrypt SSL/TLS on the endpoint itself using agents, once setup with proper root certificates on the endpoint OS, encrypted traffic can be decrypted externally, eliminating the need for agents.

Figure 1.5 shows common security solutions in a notional network, their physical location as well as the network stack layer they operate on. Perimeter security systems, such as intrusion detection and prevention systems (IDS/IPS), firewalls, and threat intelligence gateways monitor and secure traffic to and from the local network and the Internet. Threat intelligence gateways, firewalls and IDS/IPS aim to secure the network from threats from the Internet while antivirus, anti-malware, DLP, web application firewalls (WAF) and email security solutions aim to secure the particular service or the endpoint. Network segmentation, such as use of a demilitarized zone (DMZ), keeps internal devices of the network compartmentalized from resources that needs to be directly exposed to the Internet which have a higher probability of getting compromised. Devices within the DMZ are managed devices and can thus accommodate an agent based EDR system. It can be seen that the growing number of unmanaged devices, and the transient and mobile,

bring-your-own-devices on the enterprise network are not as protected as the other devices and the perimeter which can be protected using an agentless EDR system.

Most cybersecurity research uses either network data or endpoint data but rarely both together. Examples of research that uses network flow data include [52], [53], research that uses system resources such as CPU, RAM and processes include [54], [55], [56], research that uses file system information include [57], [58], [59] and research that uses logs include [60]. Some research focuses on mobile devices, e.g., [61] and [62]. To the best of our knowledge, a system has not been reported in the literature for agentless EDR that combines network traffic statistics as well as endpoint telemetry together.

A very limited number of datasets include both network traces and endpoint telemetry that may be used for EDR system development. One example is CIC-IDS2017 [32] which contains 5 days of traffic and endpoint data from simulated hosts that spans the protocols HTTP, HTTPS, FTP, SSH, and email. Attacks include brute-force FTP, brute-force SSH, DoS, web attack, infiltration, IoT and DDoS, as well as the relatively recent Heartbleed vulnerability. The dataset contains more than 80 network flow features collected from the main switch along with endpoint data such as memory dumps and system-calls from 25 hosts including Ubuntu, Windows and Mac. A major drawback is that the endpoints are synthetic. Since the endpoint related data are captured from within the host itself, it may be more suitable for agent based EDR systems.

Table 0.1 summarizes commercially available EDR systems and their capabilities [63]. Only Microsoft, Infocyte, and Cybereason vendors have agentless EDR products. We also compare them with regard to mobile device coverage and whether the appliance is hosted on the client’s premises or on the vendor’s cloud. On-premises installations can provide more flexibility and cloud-based systems can provide more up-to-date threat analysis.

Table 0.1: Commercially available EDR systems and their capabilities [63].

Name	Mobile	On-prem	Cloud based	Agentless
Microsoft Defender	No	No	No	Yes
Infocyte	No	No	No	Yes
Cybereason	No	No	No	Yes
Bitdefender	No	Yes	Yes	No
BlackBerry Cylance	Yes	No	Yes	No
Broadcom (Symantec)	No	No	Yes	No
Check Point	No	Yes	Yes	No
Cisco	No	No	Yes	No
CrowdStrike	No	No	Yes	No
ESET	No	No	Yes	No
FireEye	No	No	Yes	No
Fortinet	No	No	Yes	No
F-Secure	No	No	Yes	No
Kaspersky	No	No	Yes	No
McAfee	No	Yes	Yes	No
Panda Security	No	No	Yes	No
SentinelOne	No	No	Yes	No
Sophos	No	No	Yes	No
Trend Micro	No	No	Yes	No
VMWare Carbon Black	No	No	Yes	No
Red Hat Openshift	No	No	Yes	No

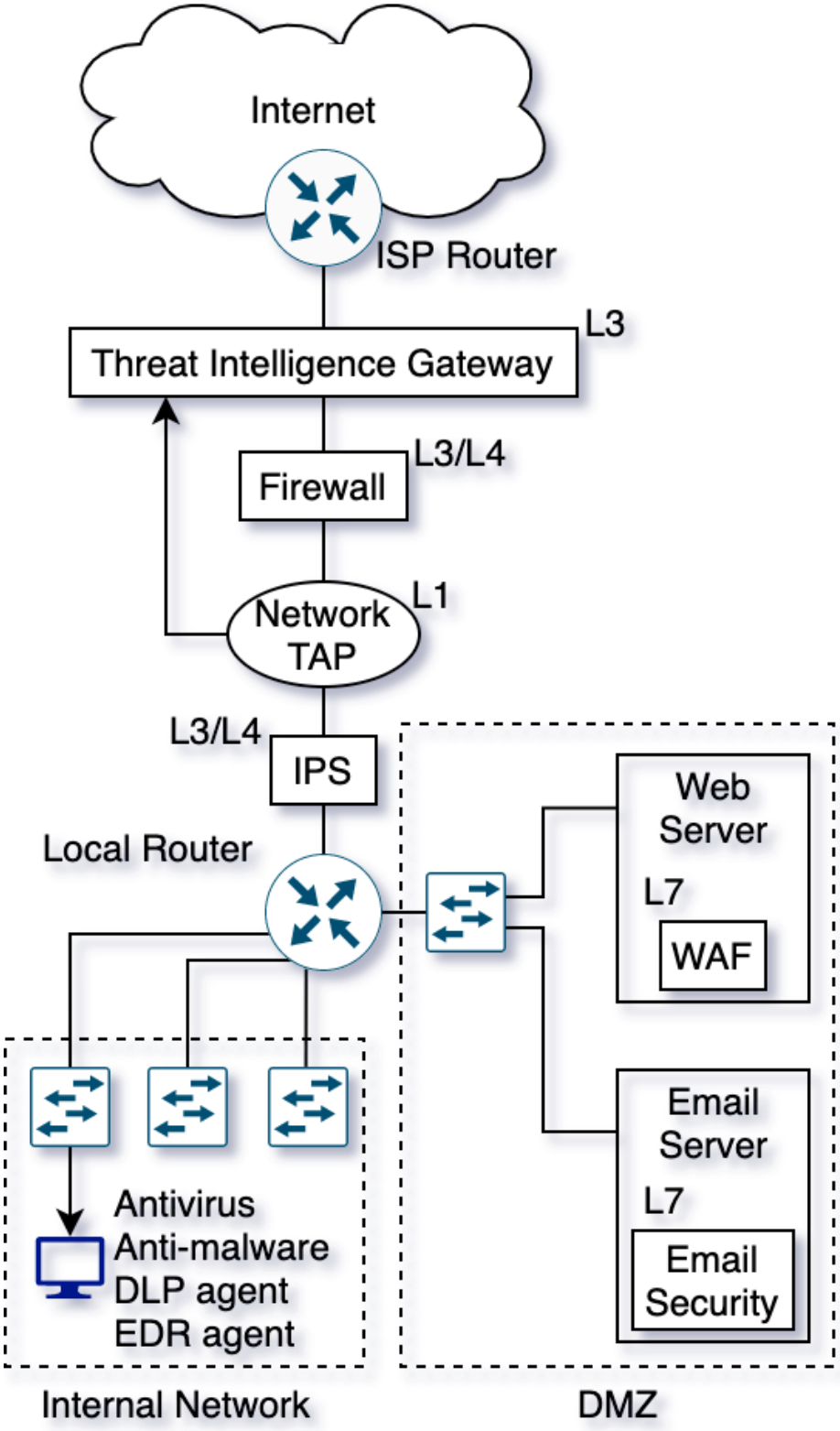


Figure 1.5: Perimeter and endpoint security solutions.

1.3 Incident Response

In this section we discuss the network incident response workflow of a security analyst using the network incident response exercise conducted using the 2019 WRCCDC data [5]. Figure 1.6 shows a typical forensic study workflow of a security analyst investigating a security incident that we used for the analysis. In this case the security incident was an endpoint compromise. First the critical services running on the endpoints are identified using the ground truth data and using passive packet analysis. Then each service was focused on using destination port number filter. Anomalous traffic was identified using the visualizations. The source of the anomalous traffic is identified. Using the ground truth IP addresses blue and red team are separately identified. Using the reverse shell port and the red-team IP address combination traffic was investigated to identify potentially compromised blue team endpoints. Further investigation was carried out to verify the compromises. Final goal of this exercise was to detect the reverse shell that originated from the victim machine to the attacker which has shown in Figure 1.7 and Figure 1.8.

The ground truth about the data was available prior to the analysis. The 2019 WRCCDC competition topology is shown in Figure 1.9. It consisted of eight blue teams whose objective was to defend network services such as web and email servers. The blue teams operated on subnets 10.47.x.0/24, where x denotes team numbers 1 through 8. A red team operating on a wide network range of 10.128.0.0/9 attacked the blue team servers using various techniques. A service check engine operating at 10.0.0.111 (and various other IPs) periodically checked the status of the blue team services to see if attacks had successfully disabled their critical network applications.

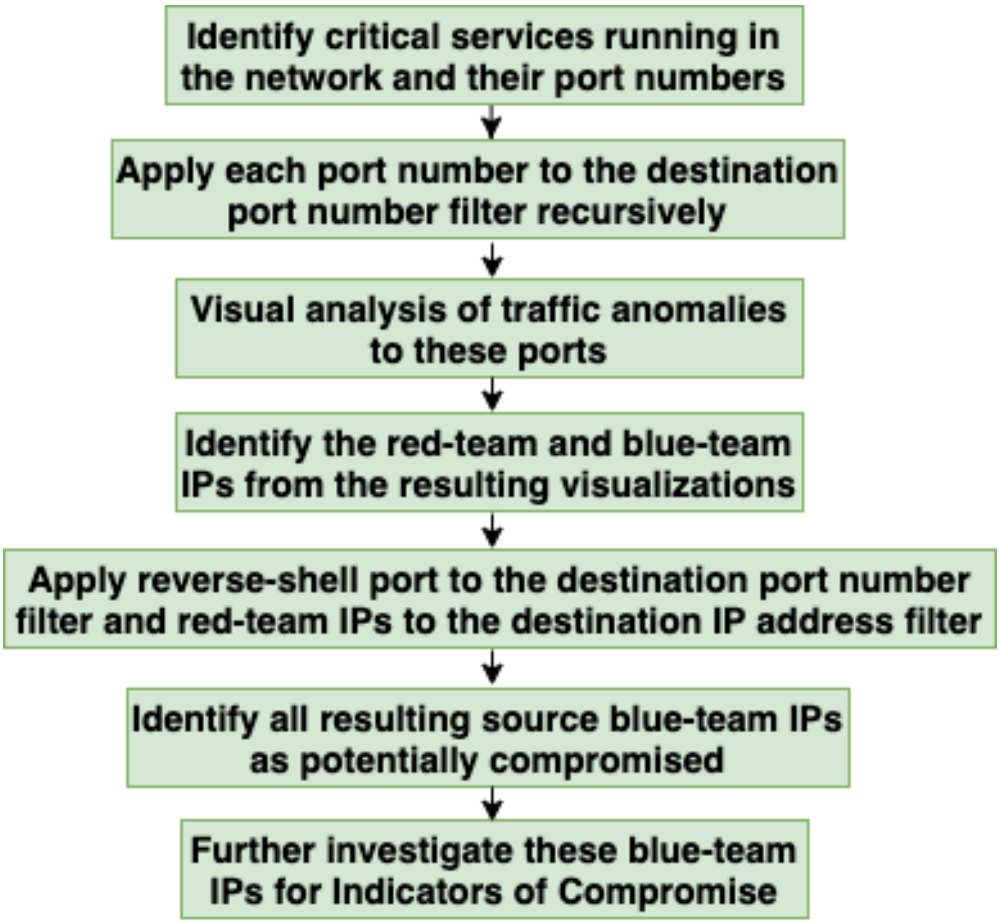


Figure 1.6: Forensic study workflow [5].

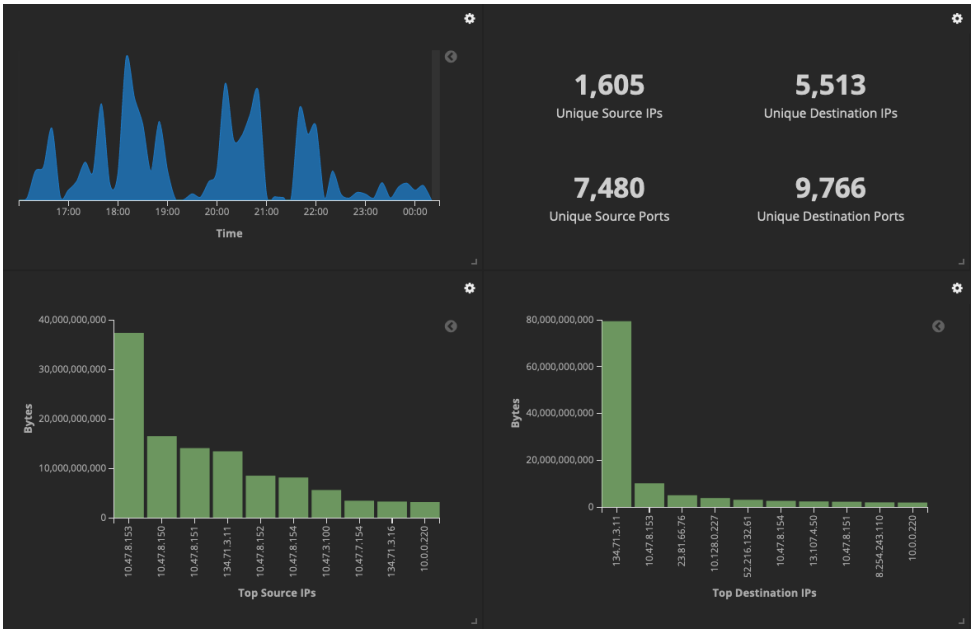


Figure 1.7: Attack visualization before port filtering [5].

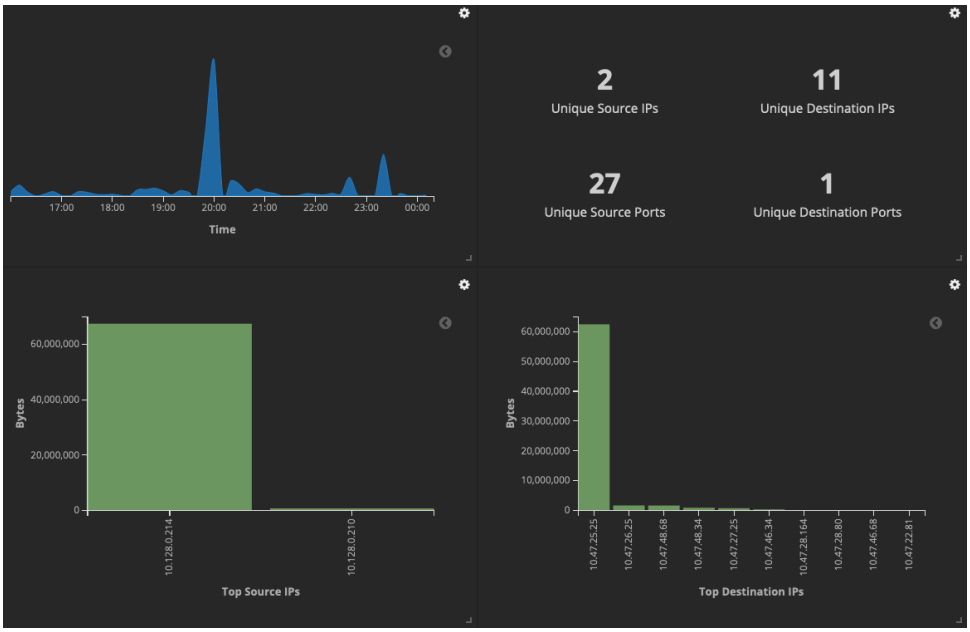


Figure 1.8: Attack visualization after port filtering [5].

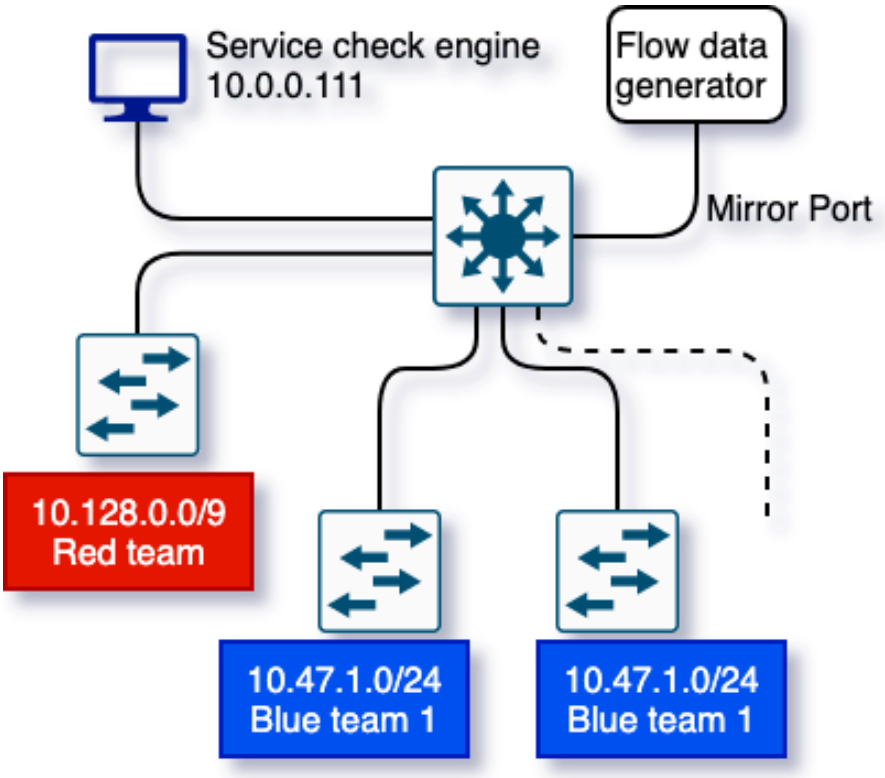


Figure 1.9: 2019 WRCCDC network topology.

.

Based on the available ground truth the following discoveries were made using the forensic study workflow. The attacking red team had changed their source IP addresses occasionally to avoid being categorically blocked by the blue team. This made it challenging to aggregate their actions and attribute them to the correct source endpoint. With the data collected at the gateway router no endpoint information other than tcpdump data of the network packets crossing the gateway was logged. No contextual information about the endpoint's characteristics (both red and blue teams) was available either. The competition rules indicated that defending blue teams were given one hour to get familiar with and update their services that they were defending to make them more resilient to the potential attacks. The versions of the services at the time of compromise were thus unknown.

The traffic was collected at the backbone of the network. It was challenging to track a single endpoint in the network using its IP address and model its behavior since IP addresses changed periodically to prevent getting block listed. It was difficult to collect and analyze all the traffic belonging to the endpoint under scrutiny in such scenarios. Even armed with some prior information which is not usually available for real world incident response situations, the analysis was slow and required substantial manual work which is prone to error. The simple topology made our task easier since it divided the red and blue teams into separate subnets. It is usually not the case in practice where IP addresses of attackers and victim endpoints are mixed within the same subnet making an analyst’s task much more challenging, calling for more robust and complete data collection mechanisms.

1.4 Contributions

There are several key contributions of this work. Novel InDepth DCS, which consists of a distributed set of sensor nodes, simultaneously collects network traffic and endpoint behavior. To the best of our knowledge, it is the only system that can perform total interaction capture from existing networks without needing any changes to the underlying hardware due to its unique architecture. The InMesh EDR system uses the rich data

provided by InDepth for correct attribution to endpoints using statistical endpoint modeling. It consists of a distance measurement technique and a decision algorithm to uniquely identify endpoints even when they are spoofing and impersonating their IP and MAC address. Together they perform endpoint detection and data attribution for incident response. We achieve these goals using WRCCDC data without simulating endpoint behavior on a cyber range that we developed for collecting experimental cybersecurity datasets which closely represent modern commodity networks.

Chapter 2: InDepth Data Collection System

Increasingly, modern networks and devices are being implemented in software such as SDN and VMware ESXi. This allows network administrators to allocate resources based on demand, dynamically and instantaneously without any downtime. Network infrastructure and virtual servers can have a flexible amount of processing power, bandwidth, and memory. Current and future networks require a data collection framework that equally applies to both hardware and software implemented environments. They need to scale well based on the throughput of the network seamlessly. With reference to Figure 2.1, the InDepth system architecture consists of a distributed sensor-node-based DCS that generates network flows at sensor nodes passively and collects endpoint characteristics actively. The resulting data records are enriched by contextual information before being sent via a secure channel to a core node for indexing and storage.

2.1 InDepth System Architecture

The sensor node receives mirrored packets from the switch from all the ingress ports of its VLAN. Since simple port mirroring provides exact replicas of all packets seen on the selected ports (except for certain error messages depending on the manufacturer) this can be performed on any modern switch hardware or virtual networking device. Simple port mirroring also does not involve encapsulation thus does not require further processing at the receiving end. Received network packets are converted to network flows in parallel using separate CPU threads. This parallel processing can be scaled up to utilize all available CPU threads.

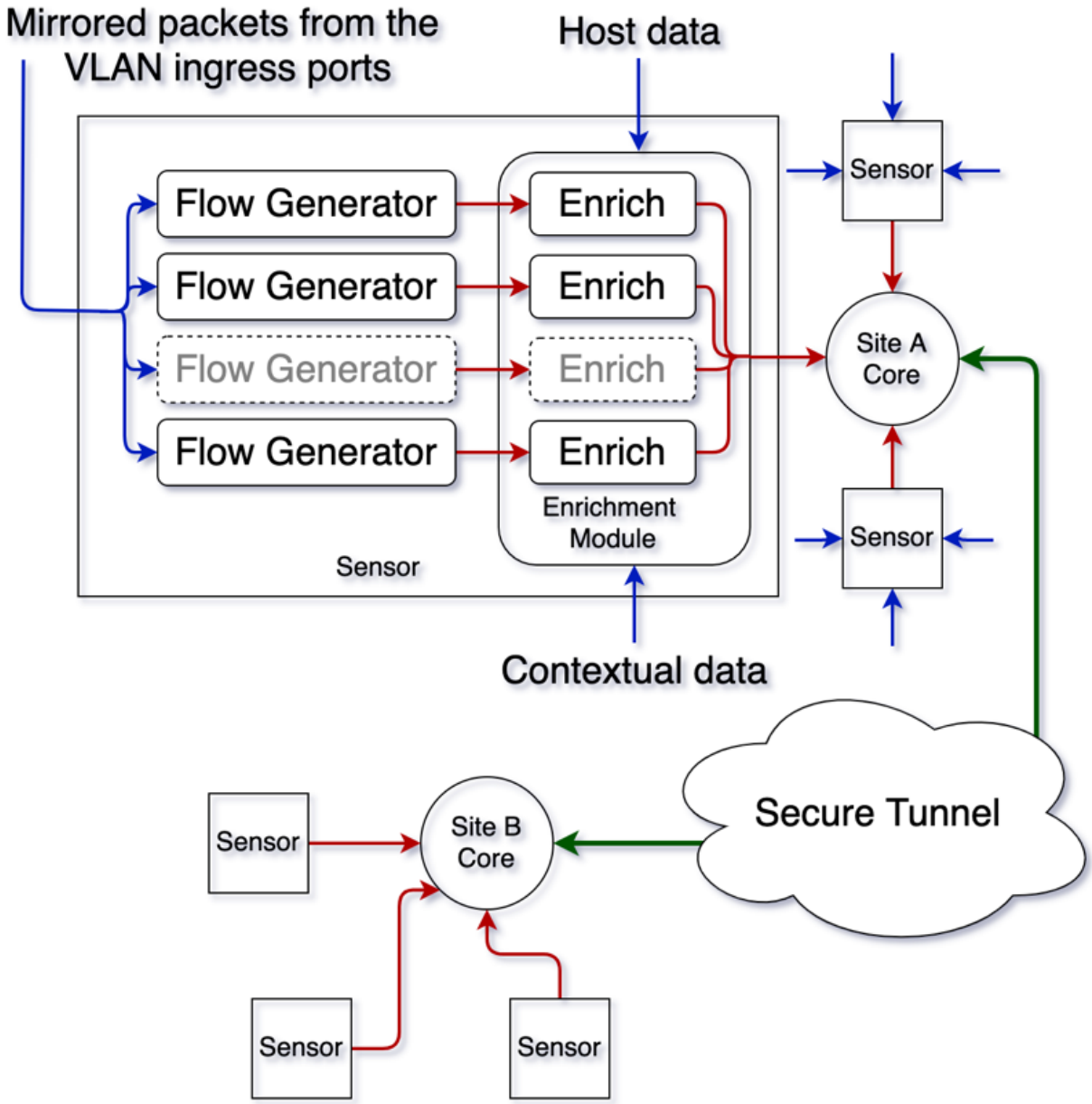


Figure 2.1: InDepth system architecture.

Endpoint data actively probed by a separate thread are collected and buffered for each device on the network. During the next step the flow records are enriched with both endpoint data and contextual data. Once the data records are created, they are exported to the core using an encrypted connection. The core aggregates the records from all the sensor nodes in the system.

Figure 2.2 shows placement of the core and sensor nodes on the model cyber range. Details about the cyber range including network policies and endpoint OS and services are described in Section 1.2. Each VLAN is allocated a sensor node equipped with appropriate system resources. Since all processing is performed on commodity devices, sensor nodes can be upgraded as needed to keep up with increasing traffic.

Open-source software was used when possible combined with commonly used industry standard software. For example, Argus [64] was used for flow data generation, nmap [65] for active endpoint probing, Elasticsearch [66] for storage and indexing, and WireGuard [67] to establish the encrypted connection between sensor nodes and the core node. These are all open-source tools and protocols. Cisco VIRL [68] was used to build the virtual network due to its ease of use and integrated features. However, it could be replaced by open-source SDN tools such as Open vSwitch [69] or Open Network Operating System (ONOS) [70] for a fully open-source stack. Different flow generators can likewise be used in place of Argus, e.g., NetFlow v9, and different probing tools such as ZMap [71] can replace nmap if needed. In other words, InDepth is a framework that can be implemented using different tools. Proprietary and open-source operating systems are used as endpoints with services running on them.

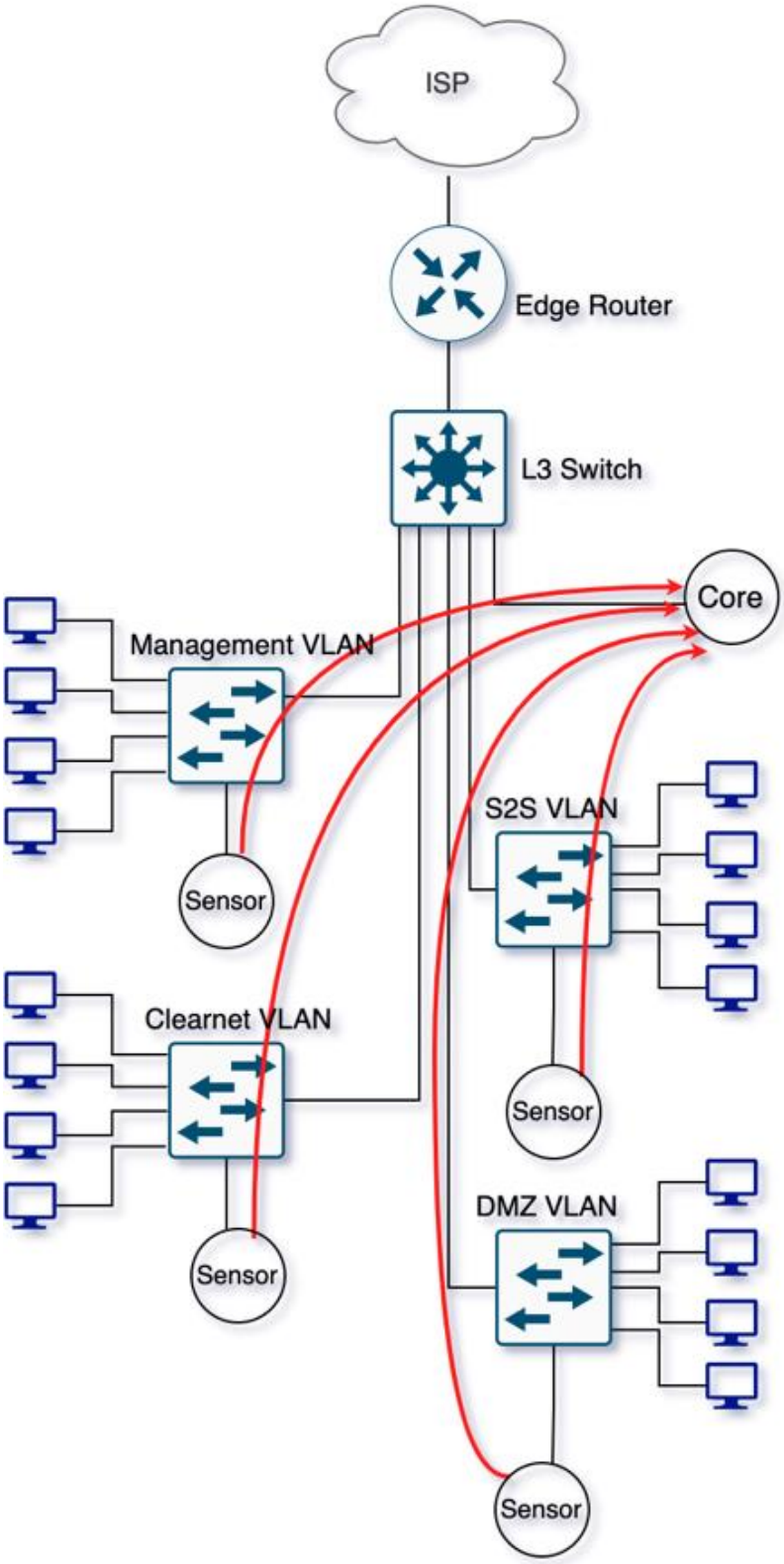


Figure 2.2: Overview of node placement and communication.

Modern networks are protected using multiple security solutions such as network firewalls, web application firewalls (WAF), next generation firewalls (NGFW), antivirus, IDS/IPS, network detection and response systems (NDR), SIEM, VPN, wireless security, email security, and endpoint security tools. They operate at different points in the network topology. For example, classic firewalls operate on the backbone router enforcing a pre-programmed set of rules deciding whether to allow or drop connections. WAF are reverse proxies that protect the application layer of the network stack by mediating the connections between the user and the web application by examining the HTTP/S connections.

NGFW are forward proxies that can detect web, email and other services and protects the user by enforcing user-based policies by performing URL filtering, file scanning for malware etc. IDS/IPS look out for known signatures and either flag the malicious activities or block them. They operate on a vast number of protocols and are used to protect Layer 2 and 4 (in some cases up to Layer 7). NDR systems such as InSight2 [5], provides real-time statistical analysis and insights about the network operations using threat intelligence feeds at the Layer 3. SIEMs collect log data from various resources. The other tools are typically designed to protect that specific service.

InDepth operates primarily at Layer 2 collecting flow data and endpoint information and augmenting them with contextual information but uses Layer 7 (the application layer) for the communication between a sensor node and the core node. Its purpose is complete interaction capture and archival. It does not replace existing security solutions. Information collected by InDepth can be used to better understand, correlate and attribute collected data to the correct endpoint.

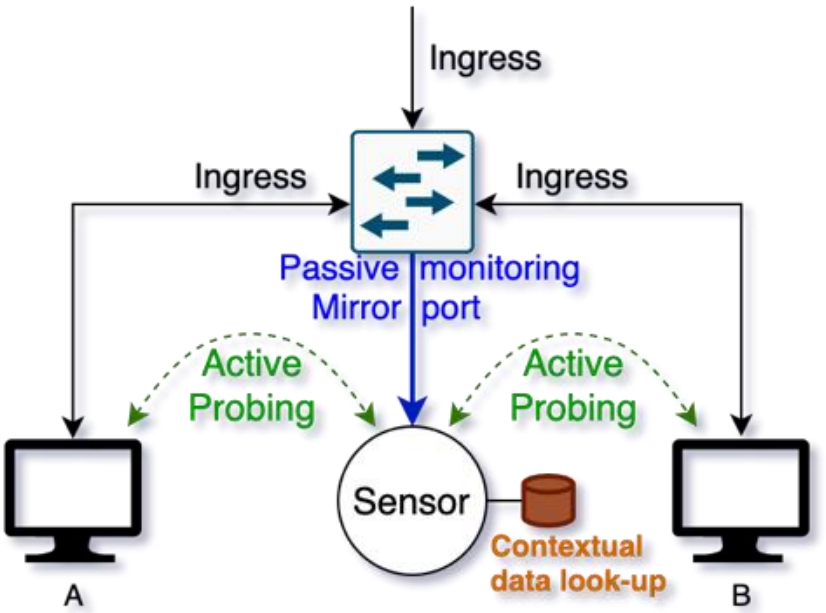
Furthermore, datasets can be created using the data collected by InDepth during cybersecurity exercises or real-world security incidents.
Figure 2.3 lists selected traffic and endpoint features collected both by passive and active monitoring at the sensor node. The features depicted in blue corresponds to the features

extracted from the network packets, the features depicted in green corresponds to system features harvested using active probing and the and features depicted in brown corresponds to the features enriched using the contextual data look-up table.

First the presence of an endpoint is detected when it uses the network to send packets or by actively scanning the subnet for endpoint discovery. Discovered endpoints are monitored for their network activity and system characteristics.

Features used for Layer 2 and 3 routing from the source to the destination are extracted from the packet headers such as MAC and IP addresses. Protocol, type of service (TOS), time to live (TTL) value and TCP options are also extracted from the packet header. Packets belonging to a single network transaction are aggregated into a flow record using the five tuples, namely, source and destination IP and port, and the protocol. Once packets are binned into flows, total number of bytes and packets for that flow are generated. Statistical features such as mean and maximum and minimum packet sizes are further generated using packet size information. Jitter is calculated using the TCP sequence numbers and packet arrival timestamps. Producer consumer ratio refers to the ratio between sent and received number of bytes. We extract the necessary traffic features including these using the Argus tool.

System characteristics are harvested by sending especially crafted packets towards a target and observing its replies. Even though every OS TCP/IP stack is compliant with the IETF RFCs they have minor differences in their implementations due to ambiguities and interpretation differences. These differences are used to identify the OS, kernel, and service information. Target endpoint's OS fingerprint is obtained by sending 1 – 16 TCP, UDP and ICMP packets and listening to the responses. Open ports are discovered by sending packets to the ports in the well-known ports range, namely, 1 – 1023. Their service types and versions are fingerprinted using similar techniques to OS fingerprinting.



Traffic features	Endpoint features
Flow duration	IP addresses
Protocol	MAC addresses
Type of service (TOS)	Port numbers
Time to live (TTL)	VLAN ID
Hop count	
Number of packets	Host OS
Number of bytes	Running services
Producer consumer ratio	Ports open
Load	Host-keys
Retransmissions	Device type
Jitter	Kernel version
TCP options	
Packet mean size	Reverse DNS name
Packet max size	Malicious tags
Packet min size	Geo-location
Packet rate	Physical location
Application bytes	Serial number

Figure 2.3: Active and passive feature categorization.

Certain services such as web servers publish information that can be read by connecting to the port. If the endpoint is running SSH it publishes its key fingerprint and the public key which can also be read by connecting to the port. We use Nmap tool to obtain these features.

Contextual information such as IOC, known botnet IP addresses, Internet Service Provider (ISP) IP ranges and physical/geo location can be further infused into the records. This information is time sensitive and contains data specific to the organization. They are indexed into a database for linear time lookups. Some examples of this information may include chassis serial number and reverse DNS.

For total interaction capture [15], network transactions must be collected as close to the endpoint as possible. In this paper, we use the InDepth [8] data collection system (DCS) which collects data on the wire between the endpoint and the first hop at each subnet using a distributed network of sensors. To the best of our knowledge, this is the only system capable of collecting network traffic and endpoint telemetry simultaneously from each local area network (LAN) while augmenting the data with contextual information. This data is used to create endpoint models called host descriptors (HD).

InDepth uses MAC address to identify the endpoint, under the assumption that MAC addresses do not change. The original MAC address is preserved since traffic is captured just after a packet leaves the source endpoint and before it is forwarded to the next subnet by the switch. InDepth is a distributed DCS that consists of a sensor node per LAN and a core node per network. A network consisting of n LANs will have n+1 nodes in total. Each sensor node is connected to a mirror port that replicates an exact copy of the traffic for that LAN for passive monitoring of the traffic as well as active probing of the endpoints to harvest system characteristics. See [8] for more details.

Even though MAC addresses are less volatile than IP addresses, they can still be spoofed. InMesh attributes endpoint activity without relying on MAC or IP addresses, ensuring correct data attribution even when an address does not have a one-to-one endpoint mapping. The goal of InMesh is to develop a unique identifier using behavioral and spatial endpoint data providing reliable information for a security incident response. 'Spatial' information refers to the VLAN where the endpoint is located.

The system enables record keeping of the different addresses used by the endpoint and their timestamps which is critical for attributing log data collected by a security information and event management (SIEM) system which can be dissociated due to address volatility. Combining the InMesh EDR with the InDepth DCS provides unified insights into endpoint activities even when they are mobile among different subnets or sites of a network. Models thereof can be used to detect behavior changes indicating a compromise, reconnaissance, or data exfiltration.

2.2 InDepth Cyber Range

We developed a modern model network topology with multiple types of devices and network applications known as a cyber range. As shown in Figure 2.4, traffic from four VLANs is routed using a Layer 3 managed switch. The default gateway is connected to the edge router which routes the Internet traffic to the ISP. This hierarchical topology can be considered a reference network from which other networks can be derived. Each subnet serves a specific purpose. We use the 802.1Q standard for tagging. For simplicity, the subnets consist of a Classless Inter-Domain Routing (CIDR) of 24 which provides 254 usable IP addresses for endpoints. The VLANs have their ingress ports mirrored to a single port that the sensor node is connected to. Port bundling protocols such as LACP can be used to increase the bandwidth when single port bandwidth is not adequate.

The Clearnet VLAN contains the user devices but has no VPN functionality or special firewall rules. It is untagged and uses CIDR 192.168.1.0/24. The management VLAN

consists of devices used by management staff which are separated from all other subnets using firewall rules. It uses CIDR 192.168.10.0/24 with an 802.1Q tag of 10. The site-to-site VLAN connects this network to an off-site network over the Internet using an encrypted connection and is isolated from other networks. It uses CIDR 192.168.20.0/24 with an 802.1Q tag of 20.

These three network segments contain devices with a variety of up-to-date OS including Microsoft Windows, Apple MacOS, Ubuntu Desktop and Kali Linux. The latter host can be used for launching attacks against the other hosts since it has built in tools to launch attacks. The DMZ VLAN contains critical services in a demilitarized zone (DMZ) including web, email and database services.

These services typically have public IP addresses accessible from the Internet as well as from the other subnets. However, for demonstration purposes we used private IP addresses from CIDR 192.168.30.0/24 with an 802.1Q tag of 30 since this cyber range is not actually connected to an ISP and also to prevent any association with potential conflict of such actual public IP addresses.

We installed latest versions of the services for experimentation purposes. They can be replaced by earlier, vulnerable services to generate threat signatures such as the Open Web Application Security Project (OWASP) tools [72] and Metasploitable [73]. The InDepth VLAN uses CIDR 192.168.40.0/24 with an 802.1Q tag of 40 and hosts the Core node which aggregates the data collected at the Sensor nodes and is isolated from the other segments. The Core node uses a single connection since it is the only device in that subnet. Figure 2.4 lists services and their versions running on the DMZ.

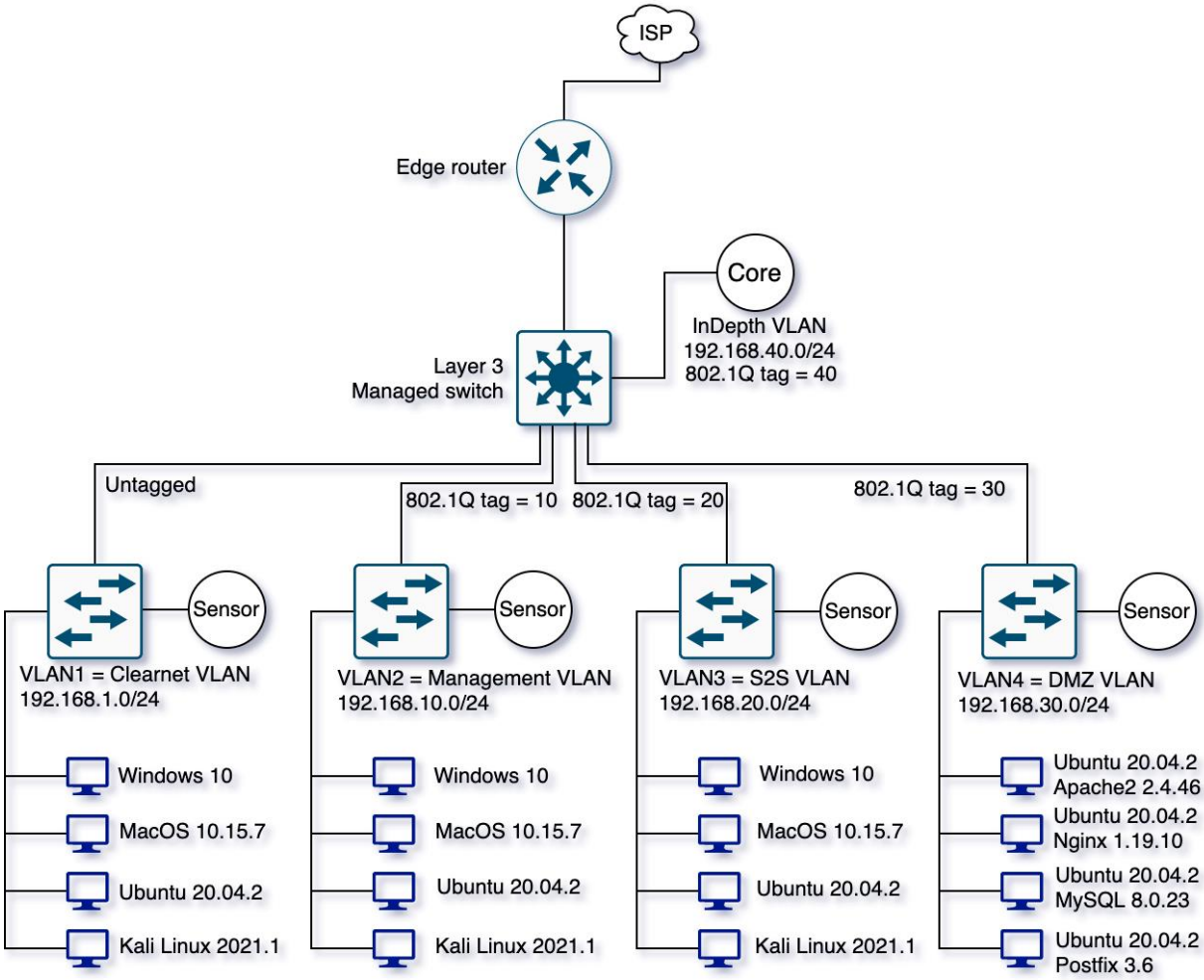


Figure 2.4: InDepth cyber range.

2.3 Comparison with Backbone Data Collection

We re-created parts of the network described by the competition rules consisting of one red team subnet and one blue team subnet. We collected traffic PCAP dumps the same way the competition gathered using the Layer 3 routing device at the backbone that connects the red and blue teams as well as using InDepth with a sensor node per LAN and a central node for aggregation. Then the following experiments were conducted in relation to Figure 2.5 where R represents the red team endpoint, and B1 and B2 represent two blue team endpoints in the same subnet as R and in a different subnet respectively.

As summarized by Table 2.1, we conducted three experiments that compares InDepth distributed data collection against standard backbone DCS capabilities. We note that the attack scenarios may not occur all by themselves but are usually a step in the attack chain. The results show that in all three scenarios InDepth collected more complete and more actionable data.

Experiment 1: This experiment consisted in aggregating and attributing all traffic originating from an endpoint when its IP address is changed by the owner or reassigned by a DHCP server. This use-case utilizes the endpoint characteristics that InDepth collects in addition to the network features.

Table 2.1: Examples of attacks that uses unique features of InDepth.

Experiment	InDepth Feature	Notes
1. Aggregation and attribution	Endpoint characteristics and contextual information collected in addition to conventional network flow information.	Attacker changes source address occasionally to avoid being blocked by victim. Or lease renewal for a different IP address by the DHCP server.
2. Reconnaissance and lateral movement	Inter and intra VLAN peer-to-peer traffic capture.	Attacker uses techniques to move deeper into network after gaining initial access.
3. IP address spoofing	Flow visibility at both source and destination for traffic within the network reveals the attacker location.	Attacker not interested in replies from victim(s), or the spoofed source IP address to that of the victim. Used in DoS or flood attacks.

In the 2019 WRCCDC dataset, we observed that the red team attacker IP addresses occasionally changed to avoid being permanently blocked by the blue team to protect its services from being further attacked. This is not frequent enough to be regarded as traditional spoofing since the red team still needed to listen to the replying packets to perform these attacks, unlike DoS attacks where the attacker is not interested in the replies. The red team used the Metasploit framework to send the attack payload data to compromise the target blue team device before waiting for the response. A successful compromise sent a reverse shell to the attacker device which was used to subsequently access the compromised device. We replicated this behavior in our replica network. As expected, when mixed with other traffic, detecting this IP address change and tracing the traffic back to the original real source device was challenging. From a flow perspective, it appeared as if the attack was being conducted by multiple devices.

During security investigations, all the network activity and attributes belonging to the endpoint under investigation need to be manually compiled and analyzed. This process is time-consuming and prone to error when data-link layer information and endpoint characteristics are not available for the analyst from the centralized DCS. Currently in practice, attribution is achieved using packet marking, ICMP traceback messaging, reactive tracing, hop-by-hop tracing, IPsec authentication logs, and traffic pattern matching.

These techniques require special network packets to be crafted, modifying the firmware of the routers, manual tracing, logging into the routers, comparing routing tables and advertised source IP addresses, and scanning through logs to detect discrepancies in IPsec security associations generated by Internet Key Exchange (IKE). The process, which requires a significant amount of time and skill and is usually carried out by a team of security experts, may only be feasible when large-scale data breaches are detected and the network is completely shut down, to stop ongoing damage caused by the attack.

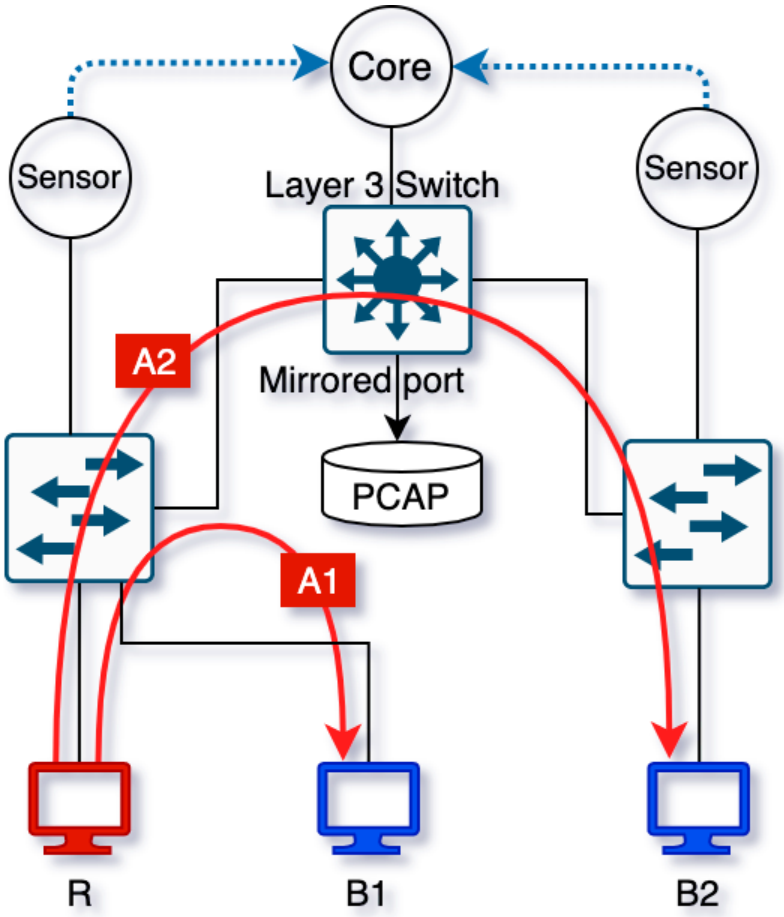


Figure 2.5: Centralized vs. distributed DCS using 2019 WRCCDC topology.

InDepth collects data-link layer information such as MAC address at the originating subnet, endpoint characteristics including OS, services, open ports, host-keys, device type and kernel version, etc. and other extensible amount of contextual information making the attribution of network activity to the right endpoint more reliable. MAC address spoofing is outside the scope of this work and may require endpoint modeling to be detected.

Experiment 2: This experiment focused on reconnaissance and lateral movement. A compromised device on the blue subnet initiated network connections on other unsuspecting devices in the network. This use-case utilizes the peer-to-peer traffic captured by InDepth both within a VLAN and between VLANs. During the A1 attack R initiated a connection to B1 which is on the same subnet as R, simulating an exploit payload for a known vulnerability. The same procedure was then carried out for B2 on a different subnet during the A2 attack. As seen in Figure 2.5 backbone data collection did not observe A1 attack connections, missing the footprint of the reconnaissance and lateral movement on the network. InDepth, on the other hand, was able to see the A1 attack due to endpoint characteristics such as OS and device type of R included in the collected data. The A2 attack was detected by both data collection mechanisms.

The device initially compromised, here B1, is usually a low priority device on the network leveraged by the attacker to get inside the network. The average break-out time, which is the time for an attacker to move laterally into other systems after gaining access to one system, is just under 2 hours [74]. Detecting an attack and acting on it within this time period is critical as lateral movement if successful can cause serious damage. Network administrators typically use a combination of Endpoint Detection and Response (EDR) tools to identify these attacks. Real-time network and endpoint information collected by InDepth may be warranted for detection in a timely manner.

Experiment 3: This experiment studied spoofed source IP addresses. The visibility provided by InDepth by capturing the flow at both the source and the destination allowed us to observe the attacker activity even though the source IP was spoofed.

This technique is used in DoS and certain flood attacks. One of the most common DoS attacks seen in practice is the buffer overflow attack. Under this attack a server receives more traffic than it is built to handle causing the system to be unavailable for legitimate request. The attacker randomizes the source IP address since it is not interested in the replies from the victim. There are two major types of flood attacks. ICMP type flood attack sends spoofed packets that ping every computer in the network and due to misconfigurations in the network these packets get amplified ultimately rendering the network unusable. SYN flood initiates TCP handshake by sending the first packet but never completes the handshake which exhausts server resources rendering the service unavailable for legitimate users. While large scale DoS attacks occur in practice involving many compromised devices through the Internet often using botnets, these attacks can also occur from within the network to disable critical local services.

These types of attacks use spoofed source IP. With centralized DCSs it is almost impossible to detect the LAN segment the traffic is generated from. Since InDepth captures the connection both at its origin and destination network flows belonging to the attacker can still be identified. If the compromised endpoint is a part of a botnet, it can be used to launch dos attacks to the public Internet damaging the trust in the organization. This information can be used to remove the compromised device from the network.

Chapter 3: InMesh Endpoint Detection

Reliable endpoint detection is crucial for a security analyst when conducting a forensic investigation. For example, flow data from archives and logs from a SIEM belonging to a particular host during a time period can be extracted using reliable address to timestamp mapping provided by InMesh. Behavioral changes can likewise be monitored over time to establish endpoint baseline behavior and detect unusual or malicious activities. It also reduces the burden of the analyst to manually compile the addresses used by the endpoints which may be incomplete.

3.1 Endpoint Data Collection

Endpoint discovery refers to the knowledge that some endpoint connected to the LAN while endpoint detection refers to identifying which endpoint it is. In other words, discovery is the sensing the presence of an endpoint and detection is differentiation of it from other endpoints. First, endpoints need to be discovered on the network before performing detection. Discovery is performed in two ways by the sensor nodes. Passive detection waits for the first packet from the newly connected endpoint and active probing periodically probes the subnet for any endpoints that has not yet sent any packets. Passive detection occurs when an endpoint automatically broadcasts a gratuitous ARP message to announce or update its IP and MAC address mapping to the entire subnet. Gratuitous ARP is an ARP response initiated by the endpoint itself rather than a reply to an ARP request. Active probing uses ARP packets for IPv4 and neighbor discovery protocol (NDP) for IPv6 endpoint discovery. The probing frequency depends on the size of the classless inter-domain routing (CIDR) IP address range. Discovery of a single endpoint takes a minimum of 250 ms, but in practice it depends on the network conditions, device power profile and Wi-Fi signal strength if connected wirelessly. In case a device connects to the network but neither sends packets nor responds to probing, it cannot pose a threat to other devices and will not be tracked.

Once an endpoint is discovered on the network, the sensor node probes it to harvest endpoint features and passively observes it for sample time (tS) for its traffic activity. Then an HD snapshot containing traffic statistics, endpoint features, and source sensor node ID (which provides spatial awareness) is created, timestamped and sent to the core node via a secure tunnel using a hub-and-spoke VPN. Figure 3.1 depicts how data are collected from the endpoints via the sensor nodes and sent to the core where endpoint detection can take place.

Federation shown in Figure 3.1 allows endpoint models to be shared with other sites which may be geographically distributed and connected over the Internet using a secure channel. It expands the visibility beyond the network perimeter providing more insights into its behavior during roaming. Endpoint models are shared between remote deployments of the organization using a secure channel between core nodes of different federated sites. Endpoint activity at each site is available during an incident response. It also reduces the time needed to create a new endpoint model each time a device connects to a federated site for the first time, since it is only needed to be created once.

The Layer 2 MAC addresses of the source and destination, Layer 3 IP addresses of the source and destination, and Layer 4 protocol make up the primary five-tuple used to identify a network flow. Some traffic features are directly collected using packet header information such as TCP base sequence numbers, TCP window advertisement, type of service (TOS) value, time to live (TTL) value and timestamps. Other statistics are generated after packets are binned into flows such as the number of packets, bytes, flow duration, packet loss, packet re-transmissions, packet rate (packets per second), interpacket arrival time and runtime. The destination endpoint requests resending of the lost packets when it identifies those packets are missing using the TCP sequence numbers. The source re-transmits the missing packets, and we capture the number of retransmissions per flow. Jitter is calculated when packets arrive out of order using the same TCP sequence numbers. High amount of jitter can have a negative impact for VoIP, video conferencing and streaming.

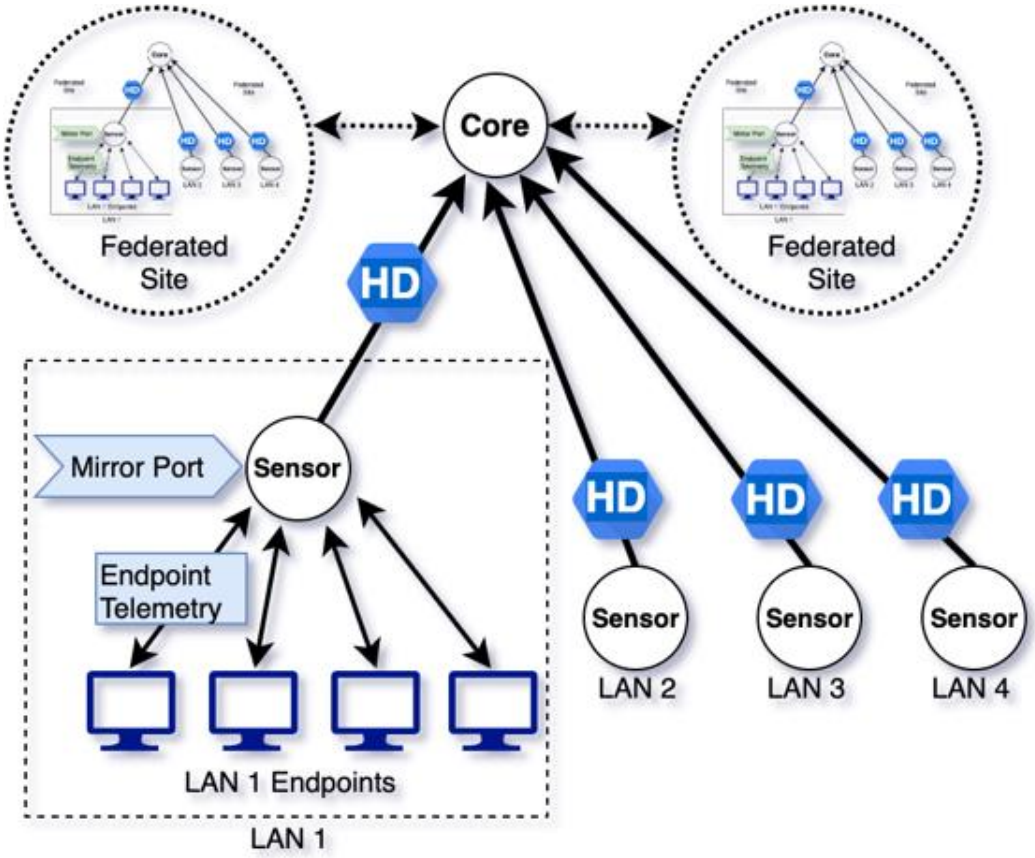


Figure 3.1: Federated logical dataflow of InMesh.

Furthermore, each of the traffic features can be further divided into source and destination such as number of packets, bytes, application-level bytes, re-transmissions, packet rate, interpacket arrival time, jitter, TCP window advertisement, mean maximum and minimum size of the packets. Percentage values for features such as packet loss and re-transmissions are also calculated. These features are collected by observing the traffic from the mirror port.

Endpoint features are collected using especially crafted packets and observing replies to them. OS detection requires at least one open port and one closed port. An open port usually corresponds to an active service running on that port that responds to incoming messages. In some cases, ports can be filtered. Filtered ports do not respond to probes since a firewall or a network obstacle is blocking the communication. Closed ports do not have services listening for packets. Port status changes when services are installed and start responding to incoming requests. If the endpoint has a public IP address, reverse DNS information is collected. Device type, common platform enumeration (CPE), OS details such as the kernel version, hop distance from the sensor, and traceroute information are also collected. The number of endpoint features may increase with an increase in the number of open ports. For each open port, the following additional information is collected:

1. Port number
2. Service name
3. Service version
4. Service data

Service data includes host-keys, HTTP header, and banner information.

In total, 126 fixed network features are collected. The number of endpoint features is given by 10 + 4m where m is the number of open ports. So, for an endpoint that has one open port, 14 features are collected. In addition, ingress and egress promiscuity is

calculated for the endpoint as a whole and per open port. These features are included in the snapshot sent to the core node. Each snapshot defines the position of the endpoint behavior observed within the t_S time in the high-dimensional Gaussian point cloud.

3.2 Endpoint Detection Technique

The goal of endpoint detection is to identify endpoints even when their addresses have changed or when they have physically moved among different subnets. So, we can attribute the activities performed by each endpoint accurately for incident response. As an agentless EDR system, InMesh performs endpoint detection at the core node using the snapshot features. System attributes such as OS and port information etc., provide useful information about the system characteristics of the endpoint. Statistics about the traffic activity show the behavior of the endpoint on the network. However, not all features collected are suitable for endpoint detection. In this subsection we discuss the feature reduction methodology we used to select the core features that we use for endpoint detection. We first select the network features that define the endpoint behavior by removing irrelevant features. We analyze the endpoint behavior in its Gaussian point cloud using these features. Then we compare three statistical feature reduction techniques to reduce the number of features within the original feature space. Finally, we perform the same analysis using the reduced number of features and show the effectiveness of feature reduction. We also discuss the statistical distance measurement technique and present the endpoint detection algorithm using a decision tree, that we use on the final features for endpoint detection.

Data is traditionally transformed. Features that vary are often standardized by subtracting the mean and dividing by the standard deviation. Static features are normalized using the minimum and maximum values. However, this requires recalculation each time a new snapshot is added, which is not scalable. Additionally, endpoint behaviors tend to drift over time which could be problematic if the transformation is based on historical values.

Instead, we apply a logarithmic transform to each feature when a new record is received which helps downplay large differences in the data.

Some quantitative features are derived using others such as total bytes, which is the sum of source and destination bytes, and total packets, which is the sum of source and destination packets. Qualitative features that are useful to define characteristics of a single flow, such as the protocol and TCP options, are not useful outside of the context of that flow. We do not consider these features for endpoint detection. We also do not use MAC and IP addresses which are collected as a part of the network features since we do not rely on them. Furthermore, we omit features from the statistical distance calculation that are not common to all networks, such as autonomous system (AS) number, which is used for routing packets between Internet service providers (ISP), and multiprotocol label switching (MPLS) identifier, which is considered the Layer 2.5 that provides faster switching in wide area networks (WAN). This reduces the original feature space to 48 network features that describe the endpoint behavior.

Feature reduction techniques that map composite features to a lower dimensional space, such as principal component analysis (PCA), are detrimental for security analysts who need to understand the physical meaning of the features when investigating an incident [7]. An alternative is to use information entropy to eliminate features in the original high dimensional space [75]. Here, we compare two such criteria defined as follows [76], [77]:

$$\mathrm{AIC(k)} = -2 \log \mathrm{L}\left(\hat{\theta}_{\mathrm{k}}\right) + 2\,\mathrm{k} \tag{3.1}$$

$$\mathrm{CAIC(k)} = -2 \log \mathrm{L}\left(\hat{\theta}_{\mathrm{k}}\right) + (1 + \log \mathrm{N})\,\mathrm{k} \tag{3.2}$$

where k is the number of features considered, θ_{k} is the corresponding feature vector, $\mathrm{L}\left(\hat{\theta}_k\right)$ denotes the value of the associated maximum likelihood estimate, and n is the number of underlying samples. CAIC has a stronger penalty for overparameterized models than AIC. We refer to the literature for more details.

Figure 3.2 illustrates the result of applying these criteria to the 48 network features. AIC yields poor discrimination. CAIC on the other hand, indicates that a minimum is achieved for 9 features. Flow data contains features that describe individual network flows as well as features that describe the behavior of the particular endpoint. Using information entropy-based feature selection, we are able to discard many features as noise while retaining a significant amount of endpoint related information. Indeed, the 9 features selected by CAIC account for 99% of all the information.

From the 14 endpoint features, 5 features are selected that are available for all endpoints and are not reflected in other features. Reverse DNS was omitted since local domain names are not always available. Traceroute was omitted since intermediate hops are reflected in the hop count. Other omitted features include device type, common platform enumeration (CPE), OS details, kernel version since in certain situations an exact match cannot be found.

Table 3.1 lists the network and endpoint features used for snapshot distance measurement. The network features are numerical while the endpoint features are categorical.

3.3 Distance Measurement

We used the InDepth cyber range [8] to generate data for the feature reduction and observe the effect of the feature reduction. The cyber range consists of four subnets with different firewall rules found in a typical network including the Clearnet subnet which usually contains user devices. The endpoints were set up to match the 2020 Western Regional Collegiate Cyber Defense Competition (WRCCDC) [78]. See Figure 3.3. We applied the ‘tcpreplay’ tool to replay packet capture (PCAP) files from each endpoint.

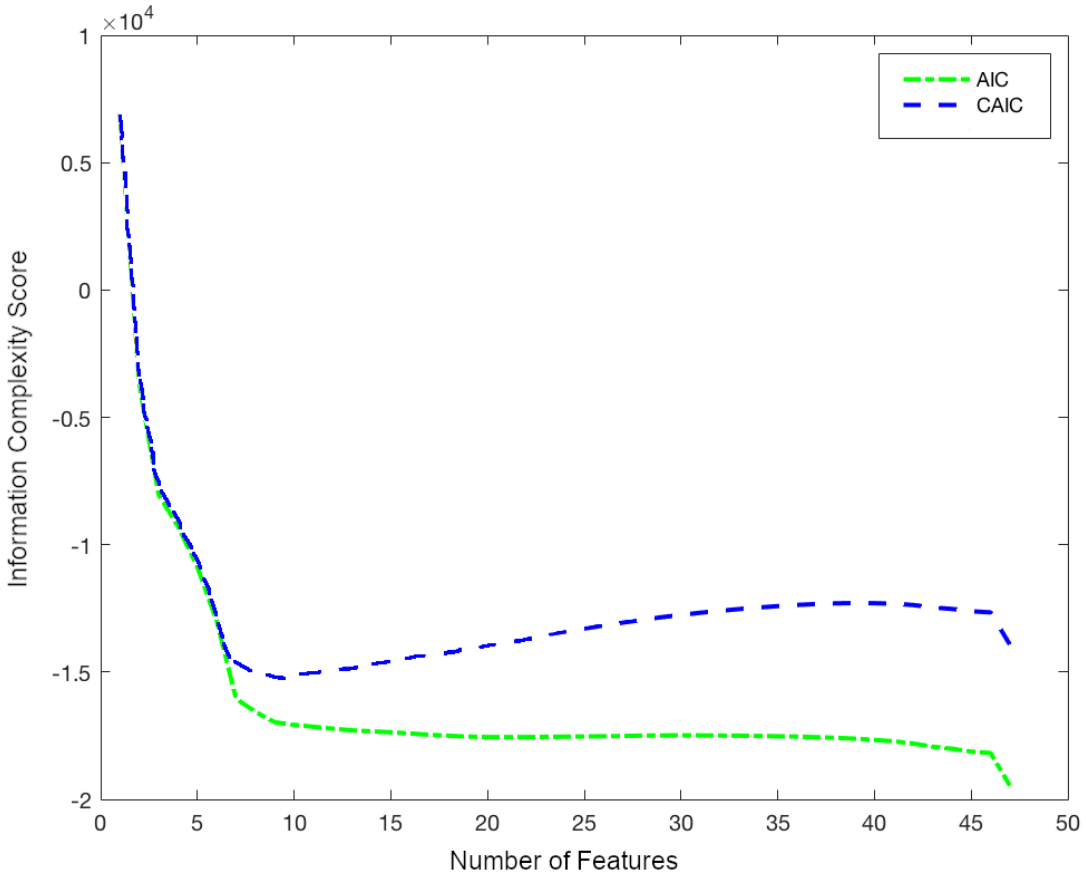


Figure 3.2: Information complexity score vs. number of features considered.

Table 3.1: Features used for snapshot distance measurement.

Network Features	Endpoint Features
Average flow duration	Operating system
Average TCP window size	Device type
Average packets per flow	Hop distance
Average packet rate	Number of open ports, N
Average PCR	Open port 0 (4-tuple)
Average time to live (TTL)	Open port 1 (4-tuple)
Average hop count	…
Ingress promiscuity	Open port N-1 (4-tuple)
Egress promiscuity	

A snapshot for each endpoint was generated and sent to the core node every t_S time interval. We collected 138 consecutive snapshots with t_S = 60 seconds. Lower values produced more volatile snapshots since they contained less behavioral information. Also, probing was found to take up to 60 seconds especially for battery powered mobile devices.

We determine if two snapshots are from the same endpoint as follows. First, the Hamming distance (d_H) is calculated for the categorical endpoint features. With only open ports considered, a Hamming distance greater than 1 is taken to imply that the endpoints are different. When this is not the case, the Euclidean distance (d_E) is calculated for the numerical network features using data averaged across all observed flows for the t_S time interval considered. Figure 3.4 and 3.5 shows histograms of average Euclidean distance observations for the same endpoint and the two different endpoints, respectively. The distance between two samples from the same endpoint follows a one-sided Gaussian curve with the majority of the distances being close to 0. The distance between two samples from different endpoints resembles a Gaussian curve with a slight left skew. To establish a threshold that distinguishes snapshots from the same endpoint from different endpoints, we chose a value of 4.32 corresponding to five standard deviations.

We can now introduce the complete InMesh endpoint detection algorithm. With reference to Figure 3.6, the core node decides if a device is new, already existing in the network, physically moving throughout subnets, or associated with spoofing or impersonation using spatial and behavioral information. Here, ‘spoofing’ means use of an inactive address while ‘impersonation’ refers to the address belonging to another device in the network. Addresses refer to either MAC or IP addresses. As mentioned earlier, ‘spatial’ refers to the VLAN where the endpoint is located.

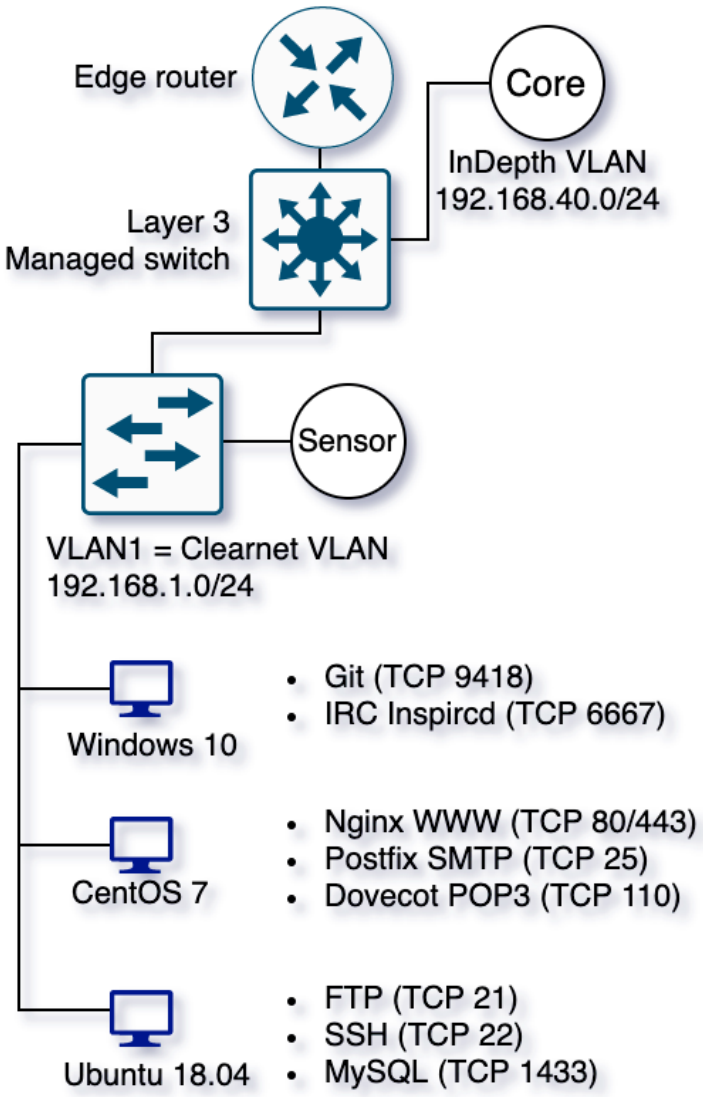


Figure 3.3: Partial InDepth cyber range used for use-cases.

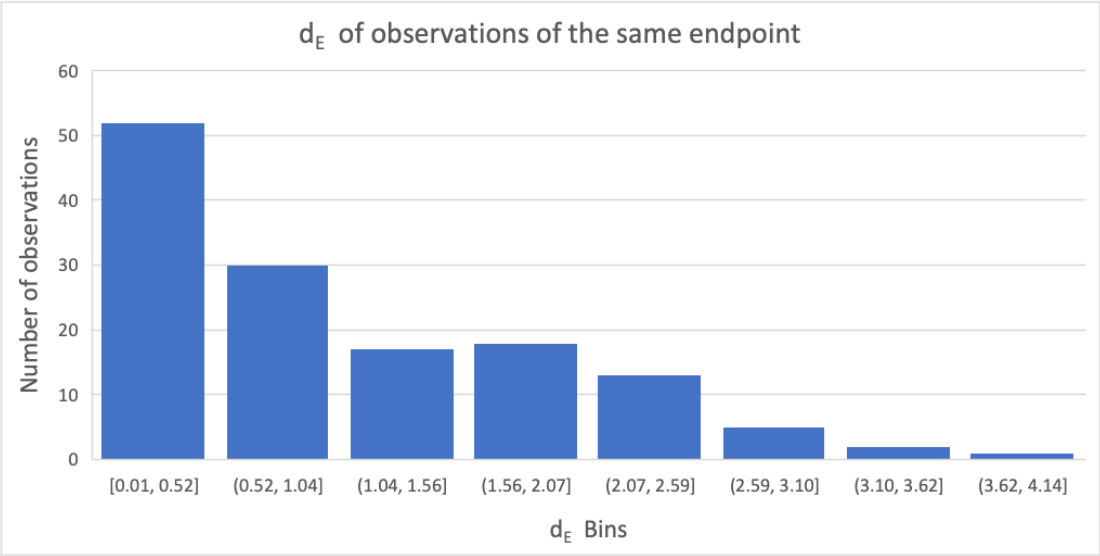


Figure 3.4: Histogram of average d_E of snapshots of the same endpoint.

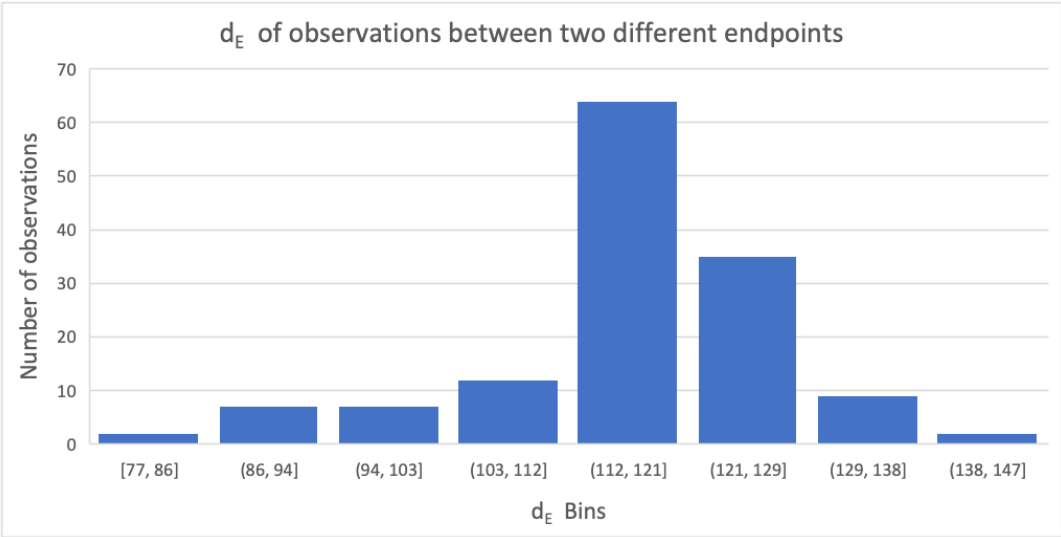


Figure 3.5: Histogram of average d_E of snapshots of two different endpoints.

If the addresses of the received snapshot match with those of the latest snapshot of an existing HD and they are determined to be from the same endpoint per the technique discussed above, the snapshot is determined to be from a known endpoint. The snapshot is appended to the HD of the known endpoint. If the addresses match, but they are determined not to be from the same endpoint, impersonation has occurred. In this case, the snapshot is appended to the HD of the original endpoint and a flag indicating impersonation is raised.

When a snapshot is received that does not have an existing snapshot for from the same endpoint with the same addresses, it is cross-checked with the last seen snapshots of existing HDs. If one exists from the same LAN (i.e., reported by the same sensor node) the endpoint is considered to be subject to spoofing. The snapshot is appended to the HD of the existing endpoint and a flag indicating spoofing is raised. If the snapshots are reported by two different sensor nodes from a different LANs, physical movement is detected. The snapshot is appended to the HD of the actual endpoint with its new IP, MAC address and LAN ID along with it.

If the addresses are different and no snapshots from that endpoint exist, the new snapshot represents a new device joining the network. A new HD is created, and subsequent snapshots are attributed to it.

If the time between the consecutive snapshots is within tS, the decision has high confidence because the two samples have high temporal resolution. Snapshots may be missing in certain situations increasing the time between the consecutive snapshots. This can happen when endpoints are unreachable due to various reasons such as disconnections, poor Wi-Fi signal, network congestion or power saving features. The longer the time between two snapshots the higher the probability of an event that can occur that can appear as spoofing. One such case is DHCP re-leasing which can happen if a device has been disconnected from the network for more than 24 hours, which is the standard DHCP lease time.

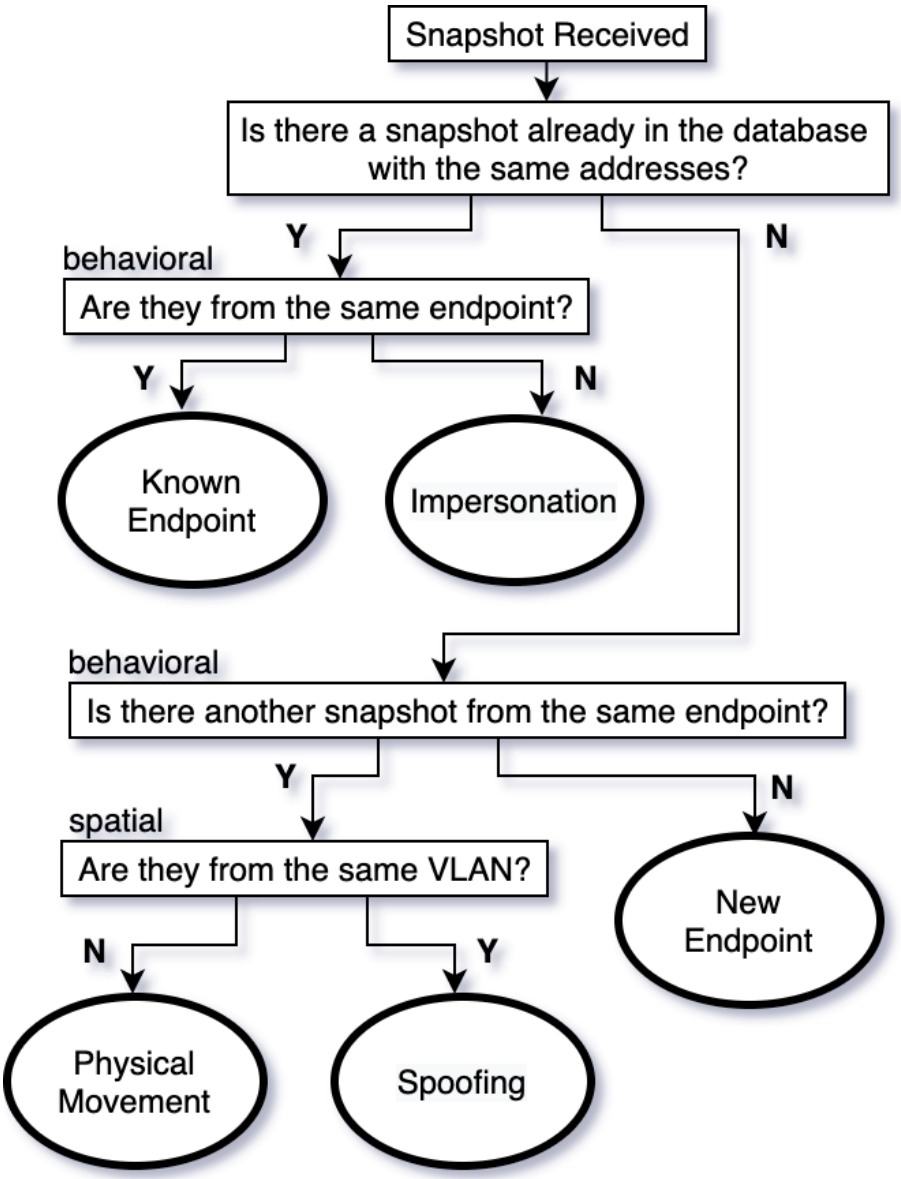


Figure 3.6: InMesh endpoint detection algorithm.

If the client did not renew its IP configuration during the DHCP lease time, then the client loses the IP configuration data and begins the DHCP lease negotiation process again which can result in a different IP address which can appear as spoofing.

3.4 Scenarios

Five scenarios were conducted to validate endpoint detection and response. Ettercap was used for ARP poisoning, Nmap for reconnaissance, and dnscat2 for DNS tunnelling [7]. Scenarios 1 through 3 relate to endpoint detection using the InMesh endpoint detection algorithm. Scenarios 4 and 5 simulate security analyst workflow [7] for an incidence response in connection with reconnaissance and data exfiltration.

Scenario 1:
In this scenario, endpoints within a LAN spoofed their IP and MAC addresses. We used the Kali Linux endpoint to spoof its own IP and MAC address. When the IP address is spoofed, it must be within the correct CIDR. Otherwise, traffic will not be routed properly to and from the endpoint. We used the Clearnet LAN, but the scenario will be identical on any other LAN since the event is self-contained within the LAN and all LANs are identical except for routing policies between the LANs.

Address spoofing is instantaneous. InMesh endpoint detection algorithm identifies spoofing when two consecutive snapshots from the same endpoint have different addresses. When we spoof the IP or MAC addresses of the Kali Linux endpoint, snapshot received before spoofing and after had the same snapshot distance but different addresses.

Scenario 2:
In this scenario, the attacker’s endpoint took over the identity of another endpoint in the network. We used the Kali Linux endpoint to impersonate the Windows and Ubuntu Linux endpoints in turn by changing its IP or MAC address to that of those. IP address

impersonation is usually associated with MITM attacks and identity masking. When an attacker wants to take over the IP address of a victim it sends an ARP packet containing the attacker's MAC address and the victims's IP address to the target device known as ARP poisoning. This can be targeted on all endpoints in the LAN or to a specific target. Using the gateway's IP address, we were able to capture victim's replayed packet capture traffic exiting the subnet. In this case we also forwarded the incoming packets from the victim to the gateway using root privileges since by default the Linux kernel drops packets that are intended for other addresses.

To capture bidirectional communication, the endpoints on either side must be modified by the attacker. Poisoning the ARP tables of Windows and Ubuntu hosts appropriately, we were able to capture bidirectional communication between the two endpoints successfully performing an MITM attack. This attack can be used to snoop on traffic between two or more devices. Similarly, critical systems can be put offline by re-directing traffic to a non-existing IP address. It can also be used to perform malicious activities as one of the existing devices.

We were able to detect impersonation when the same addresses were used by two different endpoints when they are within the same LAN.

Scenario 3:

In this scenario, endpoints were disconnected from one subnet and connected to another subnet simulating physical movement. Using the InMesh endpoint detection algorithm, we identified the device uniquely when it connects to a different subnet. InMesh endpoint detection algorithm uses behavioral and spatial information to identify physical movement between LANs. When two snapshots from an endpoint are seen on different subnets, it is deemed a physical movement event.

Mobile devices in corporate networks are increasing in number and are more difficult to track compared to stationary devices since they connect, disconnect, and move between

different subnets. Some captive portals log the MAC address when the device is authenticated into the network and track device using its MAC address. This may not be practical in all situations as seen in Scenario 1 where MAC addresses can be spoofed.

Scenario 4:

In this scenario, endpoints engaged in reconnaissance attacks including vulnerability scanning to identify vulnerable OS and service versions to determine which attack payloads has to be sent to compromise the device. Even though ideally all devices should run the most up to date operating system and services, some devices often fail to be properly updated which leaves them vulnerable to attack. We studied this scenario using the same technique used to harvest endpoint system characteristics to protect them. That is, using the Kali Linux endpoint as the attacker, we initiated a port scan on the other endpoints. Specifically, we scanned ports 0–1023 since this is a well-known port range.

Reconnaissance attacks change the egress promiscuity of the attacker when it sends probing packets to different IP addresses or to different ports of a single IP address. The producer consumer ratio (PCR) toward these endpoints also changed due to the attack payloads sent. Note that it is possible to perform slow scans that may not significantly change the behavior of the attacker's endpoint. Stealth options for the Nmap tool were not used.

Scenario 5:

In this scenario, we simulated post exploitation data exfiltration. We transferred local data to an external endpoint which acted as the C2 server using the dnscat2 tool. Typically, attackers use unsuspecting protocols such as DNS for covert channels to send data back to the C2. Data exfiltration using DNS involves breaking down the data into smaller segments, encryption and embedding into DNS queries. Web traffic is fundamental for the operation of a network which almost never gets blocked at the protocol level. Web application firewalls monitor the HTTP and HTTPS traffic. It leaves the DNS traffic

overlooked, since is essential to resolve the IP address of a domain name to reach the website and is used by almost all the users in the network.

Data exfiltration creates a signature that alters the top talkers and their traffic composition. Per protocol statistics such as average number of bytes and packets for DNS included in the snapshot revealed the altered behavior since most legitimate DNS queries are small in nature.

Chapter 4: Conclusion

In this dissertation we discussed the inherent inadequacies of the TCP/IP stack for reliable endpoint detection. Endpoints can easily manipulate their network addresses to bypass security measures, hide the source of an attack and evade detection. Logs collected from the network may not be properly attributed to the correct endpoint due to the transient nature of the IP addresses. We provided a comprehensive review of the literature related to intrusion detection data sets and data collection mechanisms. We found that publications focus on a small number of datasets that do not describe modern networks and network attacks well. We found that data collection mechanisms and tools used in practice in many cases produce data that are disassociated. To our knowledge there does not exist a dataset nor a data collection system that includes both host and network data, which is required in order to build behavioral models that allow endpoints to be detected without relying on MAC and IP addresses.

We also conducted a simulated incident response exercise using the 2019 WRCCDC cyber competition data set. Key findings of this activity include lack of data collected from the peer-to-peer communication within a LAN segment and lack of a reliable mechanism to attribute they collected data to the correct endpoint, especially when ground truth is not provided.

We developed a novel federated distributed data collection system, that can collect network traces as well as endpoint characteristics called InDepth. We also developed a cyber range as a reference network from which other networks can be derived from. By deploying InDepth on the cyber range, we conducted three experiments that reflect the attacks observed in the WRCCDC data. We showed that InDepth can be used to collect data to identify these attacks where backbone data collection mechanisms failed to observe.

We adopted the developed cyber range to create a network similar to the one used for the 2020 WRCCDC cyber competition. We replayed the data from the competition that is publicly available. Using InDepth, we created a dataset that includes the network traces, endpoint characteristics and contextual information. After rigorous experimentation using entropy based statistical methods, we selected a final set of features that we used to build endpoint models called host descriptor (HD) that can be used to uniquely detect endpoint without relying on Mac or IP addresses. Finally, we developed the InMesh endpoint detection system that keeps track of endpoints during situations namely, spoofing, impersonation, and physical movement where backbone data collection systems failed. InMesh allows tracking endpoints and their behavior regardless of their address or the location. We discussed its capabilities using five different scenarios.

Cyber incident responders can use data from InMesh to identify attack origin and compromised devices during an audit since InDepth captures peer-to-peer network traffic within a subnet, which may be missed in backbone data collection systems. Furthermore, with more reliable data attribution provided by InMesh can identify attacker true identity even when spoofing is involved.

To summarize, in this dissertation we have addressed one of the key challenges in modern network security, namely, endpoint detection. The combination of InDepth and InMesh provides a mechanism for existing computer networks to collect data that is reliably attributed. We presented its application using several realistic scenarios. The system can be deployed on production networks and cyber ranges for better incident response and dataset creation, respectively.

Bibliography

[1] "KDD’99 Dataset," 1999. [Online]. Available: http://kdd.ics.uci.edu/ databases/ kddcup99/ kddcup99.html. [Accessed 4 August 2021].

[2] "IBM XForce Documentation," [Online]. Available: https://www.ibm.com /downloads/ cas/ GBPQEPY1 . [Accessed 4 August 2021].

[3] K. D. Alferidah and N. Z. Jhanjhi, "Cybersecurity Impact over Bigdata and IoT Growth," *IEEE International Conference on Computational Intelligence (ICCI),* pp. 103-108, 2020.

[4] G. Xiong, J. Tong, Y. Xu, H. Yu and Y. Zhao, "A survey of network attacks based on protocol vulnerabilities," *Asia-Pacific Web Conference,* pp. 214-256, 2014.

[5] H. A. D. E. Kodituwakku, A. Keller and J. Gregor, "InSight2: A modular visual analysis platform for network situational awareness in large-scale networks.," *Electronics,* vol. 9.10, pp. 1747-1749, 2020.

[6] "Packet Capture File Format," 1995. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc1761. [Accessed 10 10 2021].

[7] N. Hoque, M. Bhuyan, R. Baishya, D. Bhattacharyya and J. Kalita, "Network attacks: Taxonomy, tools and systems," *Journal of Network and Computer Applications,* vol. 40, pp. 307-324, 2014.

[8] H. A. D. E. Kodituwakku, A. K. T. Shimeall and J. Gregor, "InDepth: A Novel Distributed and Federated Network and Endpoint Data Collection System," *Digital Threats: Research and Practice,* p. Submitted May 2021, 2021.

[9] H. A. D. E. Kodituwakku, H. Bozdogan, T. Shimeall and J. Gregor, "InMesh: A Novel Agentless System for Endpoint Detection and Response," *Digital Threats: Research and Practice,* p. Submitted August 2021, 2021.

[10] K. Mepham, P. Louvieris and N. C. G Ghinea, "Dynamic cyber-incident response.," *2014 6th International Conference On Cyber Conflict (CyCon 2014),* pp. 121-136, 2014.

[11] I. H, A. S. M. Kayes, S. Badsha, H. Alqahtani, P. Watters and A. Ng, "Cybersecurity data science: an overview from machine learning perspective.," *Journal of Big data,* pp. 1-29, 2020.

[12] F. L. Sikos, Kim-Kwang and R. Choo, "Sikos, Leslie F., and Kim-Kwang Raymond Choo, eds.," *Data science in cybersecurity and cyberthreat intelligence,* 2020.

[13] F. L. Sikos, "Packet analysis for network forensics: A comprehensive survey," *Forensic Science International: Digital Investigation ,* p. 200892, 2020.

[14] G. Ibrahim, V. Prenosil, J. Svoboda and M. Hammoudeh., "A survey on network security monitoring systems.," *IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW),* pp. 77-82, 2016.

[15] A. Shiravi, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection.," *Computers & Security ,* pp. 357-354, 2012.

[16] S. Fabian, J. Wallerich and A. Feldmann, "Packet capture in 10-gigabit ethernet environments using contemporary commodity hardware.," *Springer, Berlin, Heidelberg,* pp. 207-217, 2007.

[17] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory.," *ACM Transactions on Information and System Security (TISSEC),* vol. 3.4, pp. 262-294, 2000.

[18] "DARPA'98 Dataset," 1998. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset. [Accessed 4 August 2021].

[19] M. Tavallaee, E. Bagheri, W. Lu and A. Ghorbani., "A detailed analysis of the KDD CUP 99 data set," *2009 IEEE Symposium on Computational Intelligence for Security and Defense applications.,* pp. 1-6, 2009.

[20] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection.," *International Workshop on Recent Advances in Intrusion Detection. Springer, Berlin, Heidelberg,* pp. 220-237, 2003.

[21] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro and R. Therón, "UGR ‘16: A new dataset for the evaluation of cyclostationarity-based network IDSs," *Computers & Security,* pp. 411-424, 2018.

[22] A. Khraisat, "Survey of intrusion detection systems: Techniques, datasets and challenges.," *Cybersecurity,* vol. 2.1, pp. 1-22, 2019.

[23] C. Thomas, V. Sharma and N. Balakrishnan, "Usefulness of DARPA dataset for intrusion detection system evaluation.," *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security,* vol. 6973, p. 69730G, 2008.

[24] H. Kayacik, A. Günes, N. Zincir-Heywood, M. I. and Heywood., "Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets.," *Third Annual Conference on Privacy, Security and Trust,* vol. 94, 2005.

[25] R. Javier, D. Sala and A. I. Ali, "Maximizing packet loss monitoring accuracy for reliable trace collections.," *16th IEEE workshop on local and metropolitan area networks.,* pp. 61-66, 2008.

[26] B. Florian, "Multi-ciphersuite security of the Secure Shell (SSH) protocol.," *ACM SIGSAC Conference on Computer and Communications Security,* pp. 369-381, 2014.

[27] W. Cao and X. Bao, "Research on the Detection and Defense Method of the Smurf-Attack," *International Conference on Soft Computing Techniques and Engineering Application (pp. 315-324). Springer,* pp. 314-324, 2014.

[28] "CAIDA dataset," [Online]. Available: https://www.caida.org/ catalog/datasets /overview/. [Accessed 4 August 2021].

[29] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems.," *Military Communications and Information Systems Conference (MilCIS). IEEE,* 2015.

[30] N. Moustafa and J. Slay, "The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems.," in *4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security* , 2015.

[31] "ISCXIDS2012 Dataset," [Online]. Available: https://www.unb.ca/cic/datasets/ids.html. [Accessed 4 August 2021].

[32] "CIC-IDS2017 Dataset," [Online]. Available: https://www.unb.ca/ cic/datasets/ids-2017.html. [Accessed 4 August 2021].

[33] G. Creech and J. Hu., "Generation of a new IDS test dataset: Time to retire the KDD collection.," in *2013 IEEE Wireless Communications and Networking Conference (WCNC). IEEE*, 2013.

[34] N. Koroniotis, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset.," *Future Generation Computer Systems,* pp. 779-796, 2019.

[35] Z. Rongbo, T. Xu and H. Hou, "An enhanced Netflow data collection system.," *IEEE Second International Conference on Instrumentation,* pp. 508-511, 2012.

[36] C. Benoit, "Cisco systems netflow services export version 9".

[37] W. Mea, B. Li and Z. Li, "sFlow: Towards resource-efficient and agile service federation in service overlay networks," *IEEE 24th International Conference on Distributed Computing Systems,* pp. 628-635, 2004.

[38] C. Ítalo, F. Silveira, R. Oliveira, R. Teixeira and C. Diot, "Uncovering artifacts of flow measurement tools," *International Conference on Passive and Active Network Measurement,* pp. 187-196, 2009.

[39] C. Benoit, "Ipfix protocol specification.," 2005.

[40] P. Ben, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme and J. Gross, "The design and implementation of open vswitch," *Symposium on Networked Systems Design and Implementation.,* pp. 117-130, 2015.

[41] ""Splunk"," [Online]. Available: https://www.splunk.com. [Accessed 4 August 2021].

[42] P. Orosz and T. Skopko, "Software-based packet capturing with high precision timestamping for Linux.," in *2010 Fifth International Conference on Systems and Networks Communications. IEEE*, 2010.

[43] C. Morariu and B. Stiller, "DiCAP: Distributed Packet Capturing architecture for high-speed network links.," in *2008 33rd IEEE Conference on Local Computer Networks (LCN). IEEE*, 2008.

[44] "Armis Publication," [Online]. Available: https://www.exabeam.com/wp-content/uploads/2020/06/EXA_Solution-Brief_Armis.pdf. [Accessed 4 August 2021].

[45] "Endpoint Detection and Response Market - Growth, Trends, Forecasts (2020 - 2025)," [Online]. Available: https://www. reportlinker.com /p06000968/Endpoint-Detection-and-Response-Market-Growth-Trends-Forecasts.html?utm_source=GNW. [Accessed 4 August 2021].

[46] "RFC2344," [Online]. Available: https://www.ietf.org/rfc/rfc2344.txt. [Accessed 4 August 2021].

[47] "RFC4364," [Online]. Available: https://datatracker.ietf.org/ doc/html/rfc4364.

[48] "Security of IPv6 Routing Header and Home Address Options, The Double Spoofing Problem," [Online]. Available: https://datatracker.ietf.org/doc/html/draft-savola-ipv6-rh-ha-security-00#section-2.2. [Accessed 4 August 2021].

[49] A. Giani, V. Berk and G. Cybenko, "Data exfiltration and covert channels. In Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense," *International Society for Optics and Photonics,* vol. 6201, p. 620103, 2006.

[50] C. Ho, Y. Lai, I. Chen, F. Wang and W. Tai, "Statistical analysis of false positives and false negatives from real traffic with intrusion detection/prevention systems," *IEEE Communications Magazine,* vol. 50(3), pp. 146-154, 2021.

[51] "Google Transparancy Report," [Online]. Available: https: //transparencyreport. google.com/https/ overview?hl=en. [Accessed 4 August 2021].

[52] A. Gezer, G. Warner, C. Wilson and P. Shrestha, "A flow-based approach for Trickbot banking trojan detection," *Computers & Security,* vol. 84, pp. 179-192, 2019.

[53] E. Viegas, A. Santin, A. Bessani and N. Neves, "BigFlow: Realtime and reliable anomaly-based intrusion detection for high-speed networks," *Future Generation Computer Systems,* vol. 93, pp. 473-485, 2019.

[54] R. Luh, H. Janicke and S. Schrittwieser, "AIDIS: Detecting and classifying anomalous behavior in ubiquitous kernel processes," *Computers & Security,* vol. 84, pp. 120-147, 2019.

[55] P. Burnap, R. French, F. Turner and K. Jones, "Malware classification using self organising feature maps and machine activity data," *Computers & Security,* vol. 73, pp. 399-410, 2018.

[56] A. Cohen and N. Nissim, "Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory," *Expert Systems With Applications,* vol. 102, pp. 158-178, 2018.

[57] M. Rhode, P. Burnap and K. Jones, "Early-stage malware prediction using recurrent neural networks," *Computers & Security,* vol. 77, pp. 578-594, 2018.

[58] A. Cohen, N. Nissim, L. Rokach and Y. Elovici, "SFEM: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods," *Expert Systems With Applications,* vol. 63, pp. 324-343, 2016.

[59] H. Zhou, "Malware Detection with Neural Network Using Combined Features," in *China cyber security annual conference*, 2018.

[60] K. Berlin, D. Slater and S. Joshua, "Malicious Behavior Detection Using Windows Audit Logs," *8th ACM Workshop on Artificial Intelligence and Security - AISec,* pp. 35-44, 2015.

[61] J. Abawajy, S. Huda, S. Sharmeen and M. Mehedi, "Identifying cyber threats to mobile-IoT applications in edge computing paradigm," *Future Generation Computer Systems,* vol. 89, pp. 525-538.

[62] R. P. J. M. B and J. Bickford, "Detecting Malicious Activity on Smartphones Using Sensor Measurements," *Network and System Security,* pp. 475-487, 2015.

[63] " Magic Quadrant for Endpoint Protection Platforms," [Online]. Available: https:// www.gartner.com/doc/ reprints?id= 1-25BMPL2O&ct=210225&st=sb. [Accessed 4 August 2021].

[64] "Argus source code," [Online]. Available: https://www.qosient.com/argus/ downloads .shtml. [Accessed 4 August 2021].

[65] "Network Mapper," [Online]. Available: https ://nmap.org/. [Accessed 4 August 2021].

[66] "Elasticsearch source code," [Online]. Available: https://github.com/elastic/elasticsearch. [Accessed 4 August 2021].

[67] "Wireguard repository," [Online]. Available: https://www. wireguard.com/ repositories/. [Accessed 4 August 2021].

[68] "Cisco Virtual Internet Routing Lab," [Online]. Available: http://get.virl.info/. [Accessed 4 August 2021].

[69] "Open vSwitch," [Online]. Available: https://www.openvswitch.org/. [Accessed 4 August 2021].

[70] "Open Network Operating System," [Online]. Available: https:// opennetworking.org/ onos/. [Accessed 4 August 2021].

[71] "ZMap: The Internet Scanner source code," [Online]. Available: https://github.com/zmap/zmap. [Accessed 4 August 2021].

[72] D. Sagar, "Studying open source vulnerability scanners for vulnerabilities in web applications," *IIOAB JOURNAL,* vol. 9.2, pp. 43-49, 2018.

[73] "Metasploitable 2 Exploitability Guide Documentation," [Online]. Available: https:// docs.rapid7. com/ metasploit/ metasploitable-2-exploitability-guide/. [Accessed 4 August 2021].

[74] "Crowdstrike Global Threat Report," [Online]. Available: https:// www. crowdstrike. com/ resources/reports/ 2020-crowdstrike- global- threat-report,. [Accessed 4 August 2021].

[75] H. Bozdogan, "Novel dimension reduction techniques for high-dimensional data using information complexity," in *Optimization Challenges in Complex, Networked and Risky Systems*, 2016.

[76] H. Bozdogan, "Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions," *Psychometrika,* vol. 52(3), pp. 345-370, 1987.

[77] H. Bozdogan, "ICOMP: A New Model-Selection Criteria," *Classification and Related Methods of Data Analysis,* pp. 599-608, 1988.

[78] "Western Regional Collegiate Cyber Defense Competition 2020," [Online]. Available: https:// archive.wrccdc.org/ pcaps/2020/primary-site/feed1/ . [Accessed 4 August 2021].

[79] H. Bozdogan, "Mixture-Model Cluster Analysis Using Model Selection Criteria and a New Informational Measure of Complexity.," in *First US/Japan Conference on the Frontiers of Statistical Modeling*, 2016.

Appendix

Table A1: Complete list of network features collected [64]

Field Name	Description
srcid	source identifier for Argus
stime	starting time of the record
ltime	ending time of the record
flgs	flags of the Flow State
seq	sequence number of Argus flow record
smac, dmac	MAC address Source or destination node
soui, doui	OUI part of the source or destination MAC address
saddr, daddr	IP address of the source or destination
proto	Protocol
sport, dport	Source or destination port number
stos, dtos	type of service byte value of the source or destination
sdsb, ddsb	diff serve light value of source or destination
sco, dco	country code of source or destination
sttl, dttl	Source time to live: source to destination or destination time to live: destination to source value
sipid, dipid	IP identifier of source or destination
smpls, dmpls	MPLS identifier of source or destination
spkts, dpkts	Packet account from source to destination or destination to source

Table A1: Continued

sbytes, dbytes	transaction bytes from source to destination or destination to source
sappbytes, dappbytes	application bytes from source to destination or destination to source
sload, dload	load from Source or destination in bits per second
sloss, dloss	packets retransmitted or dropped from Source or destination
sgap, dgap	bytes missing in the flow Stream from Source or destination
dir	transaction Direction
sintpkt, dintpkt	inter packet arrival time from Source or destination
sintdist, dintdist	Time-based distribution of arrival time between two packets by Source or destination
sintpktact, dintpktact	active arrival time between two packets from Source or destination
sintdistact, dintdistact	Time between two packets arriving from Source or destination
sintpktidl, dintpktidl	idle time between two packets from Source or destination
sintdistidl, dintdistidl	Distribution of the idle time of 2 packets from Source or destination
sjit, djit	Jitter observed at source or destination
sjitact, djitact	active jitter observed at source or destination
sjitidle, djitidle	Idle jitter observed at source or destination
state	state of the transaction
suser, duser	user data seen at the source or destination
swin, dwin	TCP window length advertised by Source or destination

Table A1: Continued

svlan, dvlan	virtual local area network (VLAN identification number at source or destination
svid, dvid	VLAN Identification number observed at source or destination
svpri, dvpri	Private VLAN Identification number observed at source or destination
srng, erng	Center time range by start or ending time
stcpb, dtcpb	base sequence number of TCP Source or destination
tcprtt	round trip time of the connection
synack	time to set up connection between SYN and SYN_ACK
ackdat	time to set up connection between SYN_ACK and ACK
tcpopt	Connection options observed or the lack of it during connection initiation
inode	intermediate node IP address of ICMP event
offset	offset reported in the TCP header
spktsz, dpktsz	histogram of the distribution of packet sizes from Source or destination
smaxsz, dmaxsz	maximum packet size of the packets Santa by Source or destination
sminsz, dminsz	minimum packet size of the packets Santa by Source or destination
dur	Flow duration
rate, srate, drate	packet rate in packets per second
trans	total record count of the incoming Argus stream
runtime	total sum of duration of the records observed in the input

Table A1: Continued

mean	mean of duration of the records observed in the input
stddev	standard deviationof duration of the records observed in the input
sum	sum of duration of the records observed in the input
min	minimum of duration of the records observed in the input
max	maximum of duration of the records observed in the input
pkts	number of packets seen in the transaction
bytes	number of bytes seen in the transaction
appbytes	number of application bytes seen in the transaction
load	network load observed in the incoming Argus flow in bits per second
loss	number of packet retransmissions or dropped packets
ploss	percentage of number of packet retransmissions or dropped packets
sploss, dploss	number of packet retransmissions or dropped packets by Source or destination
abr	ratio between source application bytes and destination application bytes

Table A2: Complete list of endpoint features collected

Field Name		Description
dDNS		Reverse DNS name lookup for an IP address
OS		Operating system (Windows, Linux etc.)
Device type		Whether general purpose, router, bridge, firewall, load balancer, phone, printer, router, switch, WAP etc.
Common platform enumeration (CPE)		A URL that encodes seven ordered fields such as cpe:/<part>:<vendor>:<product>:<version>:<update>:<edition>:<language>
OS details (raw)		OS details including minor version and build if available
Kernel version		Linux and other OS kernel version
Hop distance		Number of network hops between the endpoint and the sensor
Traceroute		Traceroute results from the sensor to the endpoint
Number of open ports		Number of open ports
Traffic per port		Bytes transmitted per port
Port info	Port number	Port number from 1-1023
	Service name	Name of the running service e.g. Nginx
	Service version	Version of the service
	Service data	host-keys, HTTP header, banner information

Table A3: Intermediate 48 network features

Field Name	Description
load	network load observed in the incoming Argus flow in bits per second
sload, dload	load from Source or destination in bits per second
rate, srate, drate	packet rate in packets per second
sintpkt, dintpkt	inter packet arrival time from source or destination
sintpktact	active arrival time between two packets from source or destination
sintdistidl, dintdistidl	Distribution of the idle time of 2 packets from source or destination
sjit, djit	Jitter observed at source or destination
sjitact	active jitter observed at source or destination
swin, dwin	TCP window length advertised by source or destination
tcprtt	round trip time of the connection
synack	time to set up connection between SYN and SYN_ACK
ackdat	time to set up connection between SYN_ACK and ACK
sMeanPktSz	maximum packet size of the packets size by source or destination
dMeanPktSz	maximum packet size of the packets size by source or destination
smaxsz, dmaxsz	maximum packet size of the packets size by source or destination
smaxsz, dmaxsz	maximum packet size of the packets size by source or destination
sminsz, dminsz	minimum packet size of the packets size by source or destination

Vita

Hansaka Angel Dias Edirisinghe Kodituwakku is born in Sri Lanka. He graduated from the University of Moratuwa with a BSc Hons. in Electronics and Telecommunication Engineering. During this program he built robots for competitions, electronic devices for differently abled persons as class projects and participated in hackathons. He also contributed to the GLORIAD project to secure funding for $1M, from the National Science Foundation (NSF) during the internship in his junior year, which later fully funded his MS. He completed his MS in Computer Engineering with a concentration in computer networks at the University of Tennessee, Knoxville in 2017 (in 1.5 years since start). Upon graduation he continued to work full-time for another two years on the InSight2 project, which was conceived as a part of his MS under the same NSF grant. His work is actively being used at the Stanford University and Queens University for real-time network detection and response. During this time, he also actively engaged in grant submissions and community cybersecurity workshops for high-school students interested in STEM field. He started his PhD in Fall 2019 in Computer Engineering with a concentration in cybersecurity. He developed InDepth hybrid data collection system and InMesh endpoint detection and response system which are presented in this dissertation. He worked closely with Stanford University and Carnegie Melon University to ensure the research has direct applications in the industry while producing novel contributions to the academic body of knowledge. The inventions are being filed for a patent as of this writing. He also presented at various conferences, published journal articles, and talked at Carnegie Melon University, Software Engineering Institute Webcast (available on YouTube). He defended his dissertation in Summer 2021 (in 2 years since start) for a panel consisting of The University of Tennessee and Carnegie Melon University professors. Immediately after defense, he started working at the Labs division at Centripetal Networks Inc. as their first hire. His intension is to revolutionize the cybersecurity arena by building cutting edge tools, techniques, and models to solve real world problems.