TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Sara Farooghian - 177783IVSB

HANDS-ON-SKILLS LAB ON OPEN-SOURCE INTRUSION DETECTION SYSTEM(IDS)

Bachelor's Thesis

Supervisor: Kristian Kivimägi

Kieren Nicolas Lovell

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Sara Farooghian - 177777IVSB

PRAKTILISTE OSKUSTE TÖÖTUBA - AVATUD LÄHTEKOODIGA SISSETUNGIMISE TURVASÜSTEEMID

bakalaureusetöö

Juhendaja: Kristian Kivimägi
Kieren Nicolas Lovell

Tallinn 2020

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Sara Farooghian

18.05.2020

Abstract

Understanding and building a robust Intrusion Detection System (IDS) is an essential part of the correct application of cybersecurity defence in any organization or personal system. This paper proposes a hands-on-skill lab based on the IDS, which covers all three stages of installation, configuration and intrusion detection of an IDS.

Initially, the research paper reviews the concept of IDS and compares some of the commonly used IDSs called Snort, Suricata and Zeek through a side-by-side feature comparison, literature study and from a sustainability perspective. Besides that, the author reviews malware types, malware analysis techniques and malware statistics reports published during 2019 to determine trending malware.

The lab is designed based on the result of the IDS and malware study. It includes five stages: Initial lab environment configuration, Suricata installation, Suricata configuration, rule creation for simulated DDoS attack, Emotet and Trickbot infection investigation in a Linux-Based virtual environment.

This thesis is written in English and is 35 pages long, including seven chapters, three figures and five tables.

Annotatsioon

Tõhusast sissetungimise süsteemist (edaspidi IDS Intrusion Detection System) arusaamine ja selle ülesehitamine on oluline osa küberkaitsest, mistahes ettevõttes ja organisatsioonis või ka isiklikuks kasutamiseks. Käesoleva lõputöö autor pakub välja praktiliste oskuste töötoa, mis on pühendatud IDS-ile. Seal käsitletakse kõiki kolme etappi - paigaldust, konfigureerimist ja tuvastust.

Töö esimeses osas annab autor ülevaate IDS-i käsitlustest ning võrreldakse kolme kõige enim kasutatavaid IDS-e, mida nimetatakse Snort, Suricata ja Zeek. Autor võrdleb kõigi kolme funktsioone, lisades juurde ka erinevatest allikatest saadud ülevaate ning analüüsib kõigi kolme jätkusuutlikkuse väljavaadet. Lisaks sellele uurib töö autor levinud pahavara tüüpe, pahavara analüüsimise tehnikaid ning statistilisi ülevaateid aastast 2019, et tuua välja viimase aja trende, mis on seotud pahavaraga.

Töö uurimuslik osa ja pakutud töötuba on saadud autori poolt läbiviidud analüüsi käigus. Analüüs hõlmas viite etappi: töötoa algfaasi oleku konfigureerimine, Suricata paigaldamine, Suricata konfigureerimine, DDoS rünnaku korral kehtivate reeglite määratlemine, Emoleti ja Trickboti nimeliste viiruste uurimine Linux-i keskkonnas.

Käesolev töö on kirjutatud inglise keeles ja hõlmab endas 35 lehekülge. Töös on seitse peatükki, kolm joonist ja viit tabelit.

List of abbreviations and terms

API	Application Program Interface
APIDS	Application Protocol-based Intrusion Detection System
ARP	Address Resolution Protocol
BSD	Berkeley Software Distribution
C&C	Command and Control
CFG	Control Flow Graph
CGI	Common Gateway Interface
CIA	Confidentiality and Integrity and Availability
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DLL	Dynamic Link Libraries
DoS	Denial of Service
DPU	Data Protocol Unit
GB	Gigabit
GPL	General Public License
HIDS	Host-Based Intrusion Detection System
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IFT	Information Flow Tracking
IP	Internet Protocol
IPS	Intrusion Prevention System
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization
ISP	Internet Service Provider
KSN	Kaspersky Security Network
LAN	Local-Area Network
MAC	Media Access Control
MTU	Maximum Transmission Unit
NIC	Network Interface Card

NIDS	Network-Based Intrusion Detection System
NSM	Network Security Monitoring
OISF	Open Information Security Foundation
Op-codes	Operation Codes
OS	Operating System
OSI	Open Systems Interconnection
OSS	Open-Source Software
PCAP	Packet Capture
PCRE	Perl Compatible Regular Expressions
PDF	Portable Document Format
PERL	Practical Extraction and Report Language
PIDS	Protocol-based Intrusion Detection System
RAM	Random Access Memory
RST	Reset
SCM	Software Configuration Management
SLTT	State, Local, Tribal and Territorial
SMB	Server Message Block
SMPT	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
SSLBL	SSL Blacklist
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOR	The Onion Router
TOS	Type Of Service
TTL	Time To Live

Table of contents

List of figures

List of tables

1 Introduction

Cybersecurity Technologies I and II courses are for students in the first year master program of cyber security curriculum in Tallinn University of Technology. Each course consists of 32 hours of lectures and 32 hours of practice. The reason that the course is split into two courses is that students may have different knowledge levels about the topics covered in the course. Therefore Cybersecurity Technologies I covers basic concepts and Cybersecurity Technologies II covers advanced concepts. Students must take one of the courses according to their knowledge level.

The course's goal is to provide an overview of adversary models, technologies used by cybercriminals, incident lifecycle and quick overview of tools and tactics used in different incident lifecycle stages based on recent research papers. The topics include operating system (OS) security, malware, capabilities, information flow control, language security, network protocols, hardware security and security in web applications. The practice assignments include labs and projects that teach usage and principles of technologies such as malware analysis, anonymizing principles, firewalls, IDS and IPS configuration.

Cyber-attacks have been overgrowing during the last decade. Cyber-attacks are increasing in number, complexity and even impact. They affect the public and private sectors, individuals and even governments. Therefore, it is essential to have a defence system that can protect the valuable data and systems from these attacks. Intrusion Detection System (IDS) as a layer of defence has a vital role in strengthening the defence system. Studies show that as of today, more than 50% of organizations have IDS implemented as part of their security defence system [1]. Thus, it is important for students to understand the technology of IDSs and how to use them effectively. Thus the IDS is included in the cybersecurity technologies course and the purpose of this work is designing a hands-on-skills lab based on IDS.

1.1 Problem Statement

Hands-on-skills lab can significantly benefit all the different learners. The hands-on approach to learning is the preferred method by many universities. Hands-on labs can improve student's problem-solving, thinking, and analyzing skills. IT professionals need to have these skills to fulfill their daily tasks.

For the same reason, the cybersecurity technologies course relies on practical assignments and labs as part of teaching. However, the current lab assignment for IDS is the "Basic configuration of Suricata" from RangeForce. The RangeForce lab is good and easy to use for students; it covers Suricata installation from the package, basic configuration and a simple simulated attack. The RangeForce lab is sufficient for the "Cybersecurity Technologies I" course, but it does not cover advanced concepts of IDS and, more specifically, Suricata. Therefore a new lab is needed for the "Cybersecurity Technologies II" course, which covers the IDS principle more.

1.2 Motivation

The main focus of this work is designing an easy to follow Hands-on-Skills lab on open-source network-based intrusion detection systems. The goal is to design a lab that covers main network IDS technology principles and concepts comprehensively and straightforwardly while providing an in-depth understanding of the IDS technology. Thus, the main objectives for the lab steps are covering:

- The installation process of the IDS from source files;
- Basic configuration options and the effect of each configuration option in overall performance and outputs (logs);
- How the traffic inspection works and how to read logs and alerts through an attack simulation;
- How rules are written and how rules work;
- Investigating an incident scenario based on a malicious traffic sample.

As requirements for the main objectives of the lab, two other questions arise that need to be resolved:

1. What open-source network-based IDS is suitable for this lab?
2. Which malware is good to be used in the incident investigation of the lab?

2 Theory Background

This chapter first covers the basics of network, IDS technology, rules and malware theory background which is important in making the lab and the rest of the paper comprehensive.

2.1 Network Structure

The network infrastructure contains three categories of components:

- Devices;
- Media;
- Services.

An end device is either the source or destination of a message transmitted over the network.

An address identifies each end device on a network; these addresses are the IPs.

Internet works based on IP protocol and data is transmitted in DPU (Data Protocol Unit) in the network.

The data which needs to be transmitted on the internet goes through the seven layers of the OSI model. In the sender device, the data journey starts from the application layer to go all the way down to the physical layer (the lowest layer). After the data passes each layer, some additional fields called headers would be added to the data package. After the data reaches the lowest layer of the OSI model, it will be separated into packets and transmitted to the destination. As the destination receives the packet, the packet again goes through the OSI layer from the lowest layer to the highest layer. Each layer reads their respective fields in the header and passes the packet to the higher layer [2].

The OSI model is covered in more details in the appendix 1.1.

The IP packet structure is shown in figure 1.

<table>
<tr><td>4</td><td>8</td><td>12</td><td>16</td><td>20</td><td>24</td><td>28</td><td>32</td></tr>
<tr><td>Version</td><td>Header Length</td><td colspan="2">Type of Service (TOS)</td><td colspan="4">Total Length</td></tr>
<tr><td colspan="4">Identification</td><td>flags</td><td colspan="3">Fragment offset</td></tr>
<tr><td>Time to Live (TTL)</td><td colspan="3">Protocols</td><td colspan="4">Checksum</td></tr>
<tr><td colspan="8">Source address</td></tr>
<tr><td colspan="8">Destination address</td></tr>
<tr><td colspan="6">options</td><td colspan="2">padding</td></tr>
<tr><td colspan="8">Data</td></tr>
</table>

Figure 1 - IP packet structure [2]

2.2 Intrusion Detection System

Intrusion detection system is a software that automates the process of monitoring the computer devices or network activities and events (logs), analyzing them to detect suspicious or malicious activities [3].

There are various types of IDS, including Host-based IDS (HIDS), Network-based IDS (NIDS), Protocol-based Intrusion Detection System (PIDS), Application Protocol-based Intrusion Detection System (APIDS) and Hybrid Intrusion Detection System.

A network-based IDS is an IDS that monitors and analyzes inbound and outbound traffic to and from all the devices within the network based on the OSI packet content and header data for suspicious behavior or real-time attacks. NIDS is located at crucial points in the network to inspect traffic from all devices within the network.

Host-Based IDS is generally concerned with endpoint security, it runs on all the devices within the network, having direct access to the host, it can monitor all the traffic from and to the host on both external network (Internet) and internal network (Enterprise network). This enables HIDS to be able to detect suspicious and malicious activities missed by the NIDS. HIDS may be able to detect the malicious traffic originating from the host, such as if the host attempts to spread malware to other devices within the network [4].

2.2.1 IDS Detection Techniques

IDS can use different methods for detecting intrusion or malicious activity within the network or in the host. Signature-based detection and anomaly-based detection are the most common detection techniques.

A signature is a pattern that corresponds to a known threat. Signature-based detection is the process of comparing signatures against detected events or logs to identify malicious activities. Signature-based IDSs are great in detecting known attacks. However, they are not very useful in detecting unknown attacks (zero-day attacks), or attacks using evasion techniques [5].

Anomaly-based detection is more complicated compared to signature-based detection. It does not rely on signatures to detect intrusion. Instead, it identifies unknown attacks depending on the similar behavior of other intrusions.

An IDS using anomaly-based detection first develops profiles representing the normal behavior of a network or host (modeling the normal behavior) through monitoring the characteristics of typical activity for a pre-defined period of the time. Anomaly-based IDS can find malicious activities by comparing the definition of the standard profiles against observed detected events or logs. Anomaly-based detection methods are very effective at detecting previously unknown threats [5].

2.2.2 IDS Evasion Techniques

The primary purpose of IDS is to predict the network's state and the devices within the network. This enables IDS to predict how a device responds to different network events like receiving traffic. IDS evasion techniques aim to prevent the IDS from predicting the state of devices and desynchronize the IDS and devices in the network.

The attacker can intrude into the system using the insertion or evasion technique. It is insertion if an attacker creates network traffic that is received and processed, but the host does not receive it. It is evasion if the network traffic is received and processed by the host, but IDS does not receive the traffic (IDS is not aware of the traffic) [6].

IDS evasion techniques are explained in more detail in Appendix 1.2.

2.3 Suricata's Rules and Ruleset

Rules have a very important role in the signature-based IDS like Suricata. The IDS detection engine checks the traffic against the rule and if the traffic matches the rule, take an action as defined in the rule. Rulesets are a group rule that can be added to the IDS as a whole instead of writing each rule one by one.
This section introduces the rule and rulesets available for Suricata briefly.

2.3.1 Rules

A rule consists of the following:

- The action, that determines what happens when the signature matches;
- The header, defining the protocol, IP addresses, ports and direction of the rule;
- The rule options, defining the specifics of the rule [7].

2.3.1.1 Action

Defines the action that is taken if the signature matches, the action can be:

1. **Pass** - stops scanning the packet and skips to the end of all rules (only for the current packet).
2. **Drop** - This only applies in IPS/inline mode, and the packet will not be sent any further. Drawback: The receiver does not receive a message of what is going on, resulting in a time-out (certainly with TCP). Suricata generates an alert for this packet.
3. **Reject** - With reject action both receiver and sender receive a reject message (an RST packet is sent).
4. **Alert** - When a signature that has a rule with Alert action matches, an alert is generated which is visible to system admin. however, the traffic is treated as non-dangerous traffic and passes [7].

2.3.1.2 Header

The header consists of protocol, source and destination IP and port address and direction of the packet:

```
Action Protocol Source_IP Source_port Direction -> Destication_IP
Destination_port
```

The header of the rule is checked against the IP addresses and port addresses of the packet headers and the protocol, and if they match, depending on the rule options (if the rule options also match), the specified action in the rule will be taken [7].

2.3.1.3 Rule Options

Rule options are enclosed in the parenthesis and are separated by semicolon.
There are two types of rule options, the one consisting of keyword and setting, and the ones that only consist of keyword.

```
keyword; setting;
keyword;
```

There are broad categories of rule options that can be used in the rule. Some of the commonly used rule options are: meta-settings keywords, protocol based keywords like HTTP, IP, TCP and SSH, file keywords, flow keywords and more [7].
The rule options are covered in more details in the appendix 1.3.

2.3.2 Rulesets

It is possible to add the rules for inspection and alerting manually; however, it is recommended to use rule management tools for managing the rules in Suricata.
There are several tools available for rule management, such as Scirius from StamusNetworks which is a web interface that handles the rules file and updates associated files [8], PulledPork is a PERL based tool for Suricata and Snort rule management that can determine the installed IDS version and automatically download

the latest rules. Pulledpork only works with the Emerging Threat Open and Pro version [9].

Suricata-update is python-based rule management for Suricata from OISF for downloading and managing rules. It is the official rule management tool for Suricata [10].

There are several commercial and non-commercial rulesets from different vendors available for Suricata.

- Emerging threats: A non-commercial which is a great anti-malware ruleset;
- Trafficid: identifying and classifying traffic;
- ssl-fp-blacklists: A project of abuse.ch to detect malicious SSL connections, by identifying and blacklisting SSL certificates used by botnet C&C servers. SSLBL identifies JA3 fingerprints that help to detect & block malware botnet C&C communication on the TCP layer [11].

2.4 Malware

Malware, also known as malicious content, is a software or firmware that enters the information system with intentions of performing an unauthorized action or process that negatively affect Confidentiality, Integrity, or Availability (CIA) of an information system [12]. Malware can be written with the goal of mass infection, as previously seen with "Wanna Cry" or they might be written to infect particular targets as previously seen with Stuxnet.

2.4.1 Malware Types

Malware has different types. Some of the common types of malware are explained in the following sections.

2.4.1.1 Virus

A malicious computer code or program that intends to copy itself and infect a host without an authorized user's permission by attaching itself to a legitimate program to execute its code. The virus can potentially damage the infected host by disrupting the system or damaging data [13].

2.4.1.2 Worm

A worm can be defined as a malicious computer code or algorithm that can replicate and propagate and use the network to spread itself. Worms can destructively consume the infected system or network resources [14].

2.4.1.3 Spyware

Spyware is unwanted software that gains access to the user's device by the goal of stealing internet usage data and sensitive data like credit card or bank account information, personal information, login credentials. The stolen data later relays to advertisers, data firms, or external users.

Some types of spyware can install additional software or change some of the device's settings.

Spywares are one of the most common cyber threats; devices can easily get affected, while it is difficult to identify them.

There are four main types of spywares: Adware, Trojan horse, Tracking cookies, and Keyloggers [15].

2.4.1.4 Rootkits

A set of malicious computer programs or tools used by an attacker to get root access to a computer system and maintain the access and attacker's activities hidden from the authorized system user [15].

2.4.1.5 Malicious Cryptomining

Malicious cryptomining is also known as crypto-jacking or drive-by mining. Cryptominers are usually delivered via a trojan to the victim system and it allows attackers to use the victim device’s resources to mine cryptocurrencies like Bitcoin and Ethereum. Cryptominers do not have the same destructive effect as other malware types like Ransomware, but they consume victim system resources for the attacker's benefit [16].

2.4.1.6 Ransomware

The goal of ransomware is gaining access to a user's device, taking control of the device by encrypting the device partially or fully with a key typically known only by the attacker, to demand something to give the access back. The demand is usually money transfer through cryptocurrency, as they are not traceable. There is usually a time limit for users to pay the demand. Otherwise, the file will be deleted permanently.

Ransomware uses similar methods as other types of malware, like using network hiding tools like TOR. Ransomware is relatively simple for attackers, the payouts are high, and it is not easy to mitigate.

The most effective method against ransomware is a regular backup of the system.

Ransomware is not a new malware, ransomware has been around for almost three decades by now, but it started increasing by the emergence and growth of cryptocurrency as it makes the money transfer untraceable [17].

2.4.2 Malware Analysis Techniques

The purpose of malware analysis is to understand the malware better, to strengthen the defense against malware.

Malware can be of two types, obfuscated and non-obfuscated ones. The obfuscated malware is more sophisticated to be analyzed as the malware's author hides the malicious executable file using different methods.

There are three main approaches for malware analysis: static analysis, dynamic analysis, hybrid analysis. The main difference between static and dynamic analysis is that static analysis is done without executing the malware and dynamic analysis is performed by executing the malware and investigating the functionality of the malware [18].

2.4.2.1 Static Analysis

Static analysis examines the executable file without executing the malware to determine whether it is malicious or not. This consists of investigating the functions and libraries that are used by the executable file. Finding the linked libraries and functions are among the most useful methods to gather information about the malware.

Static analysis can be used to extract a detection pattern for the malware, and this detection pattern can be extracted based on API calls, string signature, control flow graph (CFG) and opcode (operation codes) frequency [19].

API stands for Application Programming Interface, it is a set of definitions and protocols for building and integrating applications. The API calls can reveal the behavior of programs and therefore can be used in malware detection [20].

Malware's strings can reveal the attacker's goal and intentions by carrying critical semantic information. Therefore they can be good indicators for malicious or suspicious programs [21].

A Control Flow Graph (CFG) is a directed graph-based representation of a program's code, where the code blocks are presented by nodes and control flow paths by edges. CFG can be used in the analysis of a PE file and extract the program structure, which can be used in malware detection [22].

Op-codes (Operation codes) are numeric codes that represent the instructions that show the actual operation performed by the CPU to execute or run a program [23]. Testing opcode frequency or calculating the similarity between opcode sequences can be used for extracting the pattern for malware analysis and detection.

Besides these, other less complicated features can also be used in static malware analysis like file size and function length, network features and executables file hashes.

2.4.2.2 Dynamic Analysis

The dynamic analysis may also be called behavioral analysis. In the dynamic analysis, the executable file is executed in a safe environment [24]. The executable should be executed in an environment without an anti-virtual machine and anti-emulator techniques as some of the malware can detect these environments and not show malicious activity.

Compared to static analysis, the dynamic analysis is more effective and can detect known and unknown malware. Additionally, obfuscated and polymorphic malware can not evade dynamic analysis.

Function call monitoring, function parameter analysis, and information flow tracking are typical dynamic analysis techniques [19].

A function call is the line of the code that calls the functions. Functions can provide crucial information about the overall behavior of the program. Function call monitoring tracks all the functions using hook functions. Hook functions capture the functions calls, implements the analysis procedure, performs tasks like logging target program execution, observes intermediary function calls and analyzes various inputs and outputs [25].

Function parameter analysis can also be used in the static analysis by estimating the set of possible values for the function. dynamic parameter analysis monitors the actual values passing to the function, as the function is being called, and monitors the values that the function returns when it finishes. Analyzing the function's parameters and grouping the functions can provide a detailed insight into the program's behavior.

Information Flow Tracking (IFT), also referred to as Taint Analysis, monitors the programs and investigates how the program processes the data. IFT focuses on monitoring the propagation of data (Labeled or trained) while the program is executed [26].

3 Related Works

There are various sources that provide virtual labs and assignments on Intrusion Detection Systems (IDS).

Cisco has a series of 3 labs on the firewall and IDS based on Snort, which starts by covering the virtual lab environment's preparation in the first lab, introduction to the firewall rules, and IDS signature in the second lab. The third lab allows students to perform and investigate an SQL injection attack [27]. Although the lab is good as it covers alerts, rules and incident investigation, it does not cover the installation phase of the tools nor the configuration. The first lab is mostly setting up the virtual environment, which is mainly related to network knowledge than the IDS.

There are also other platforms such as Cybrary and Linux Academy hand-on labs scenario, or virtual environment, which mostly cover similar situations of introduction to logs and alerts, rules and signatures, and incident investigation. However, they often do not include the initial installation and configuration.

There are also some smaller security exercises that cover incident investigation scenarios, usually based on malicious traffic samples, such as the Malware-Traffic-Analysis website [28], Network Forensics Puzzle Contest [29]and Lincoln Laboratory of Massachusetts Institute of Technology datasets [30]. These also cannot be considered a complete lab as they are not specifically designed based on IDS and they are more of individual security practices.

Wayne State University has its dedicated firewall and IDS lab, which is designed based on Snort [31]. The lab starts with preparing the virtual machine, followed by Snort installation from the source files. Afterward, the students are introduced to Snort configuration and rule file locations and write a new rule and add it to the Snort's rule file. Then trigger an alert for the newly added rule by sending an ICMP packet to the device running the IDS (by pinging the device from a Windows host). The lab finishes by answering the set of questions related to the lab. While the lab introduces the IDS technology well and is comprehensive, it also does not cover more advanced

configuration and incident investigation exercises. Besides, it does not provide a rich knowledge of theory through lab steps.

Like Wayne State University, Cyber Security Education Consortium has an IDS lab based on Security Onion and Snorby. In the same way, the lab starts with setting up and configuring the virtual environment, followed by Security Onion installation and configuration, and simple rule management is covered step by step. The last part of the lab is replaying sample traffic (Pcap file) and investigating the alerts [32]. Even though the lab has great step by step instructions, it still does provide a great theory knowledge, and it is more focused on what to do rather than why doing a specific action or configurations.

The approach in this paper is partially similar to Wayne State University, by covering the initial IDS installation from the source files, configuration, rule management and alerts. However, to ensure that all the objectives of this work are satisfied, the lab designed through this work would have more steps, including steps on advanced configuration and incident investigation.

4 Methodology

The lab design process is divided into 7 steps:

- Determine the IDS that the lab is designed based on what satisfies the defined criteria in the problem statement section;
- Define the lab sections and high-level overview of the labs;
- Determine the malware to be used in the incident investigation scenario;
- Determine the traffic sample for the incident investigation scenario;
- Define the installation and configuration of the IDS following the official documentation and website;
- Design the incident investigation scenario using the sample traffic and define the questions for the incident investigation scenario;
- Test the lab.

To determine the backbone IDS for the lab, a group of popular open-source network-based IDS were chosen. First, each IDS's features were compared based on their respective documentation, secondly reviewed literature that has studied and compared the IDS, and finally the sustainability of the IDS was considered.

The general flow of the lab and steps take into consideration the installation, configuration, logs, alerts, attack detection, and investigation aiming to cover all the major aspects of the IDS.

The lab aims to also emphasize on how the infection with the malicious program happens in the incident investigation step, and malicious program behavior, as the CyberSecurity Technology course also covers malware analysis. Therefore the malware that satisfies the defined criteria in the problem statement of this paper is determined by studying and reviewing the reports and statistics published by security organizations and companies. The main criteria as already defined for the sample traffic is the number of generated alerts and the total size of the file.

The installation and configuration steps are defined based on the official documentation and each command and step purpose is explained in detail or the link for optionally more reading is provided in the lab.

The incident investigation scenario is focused on the reading and understanding of the alerts and correlating the alerts to the traffic and investigating the malware delivery and analysis.

In the end, the lab was tested with a group of students with a different level of background knowledge on Cybersecurity and IDS. Each student provides the consumed time for completing the lab, positive points, negative points and recommendations.

5 Analysis

This section's goal is to cover the pre-objectives of the lab design, which is determining the IDS, and the malware for the incident investigation.

5.1 IDS Comparison and Evaluation

This section aims to compare three popular open-source NIDS to determine the IDS to be used in the lab. It first reviews and compares the 3 IDSs based on their features according to their official documentation. Then review the research papers and literature, and finally compare them from a sustainability perspective.

5.1.1 General Comparison

This section reviews Snort, Suricata, Zeek and provides a side-by-side comparison of the three IDSs.

5.1.1.1 Snort

Snort is an open-source network intrusion detection system (NIDS), that can perform real-time traffic inspection and packet logging on IP networks, as well as protocol analysis and content searching/matching.

Thus Snort can detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and OS fingerprinting attempts [33] [34].

The original founder of Snort, Sourcefire, was founded in 2001 by Martin Roesch. Cisco acquired Sourcefire on October 7, 2013. Snort's mission is to combine open source roots with proprietary innovation to deliver the most effective and comprehensive real-time network defense solutions [35].

Talos is a group of leading-edge network security experts and the largest group dedicated to discovering, assessing, and responding to the latest threats. Talos is

supported by the Snort community and writes the official Snort ruleset, the Snort Subscriber Rule Set [36].

Snort can be configured to run in three modes:

- Sniffer mode: Reads the packets of the network, displays the packets in a continuous stream on the console screen;
- Packet Logger mode: logs the packets;
- NIDS mode: Performs detection and analysis of network traffic. (most complex and configurable mode).

In the NIDS mode, Snort reads the configuration from the snort.conf file and applies the rules configured in the "snort.conf" file to each packet to decide if, based on the rule type in the rule file, an action should be taken. One drawback of Snort is that it is not an application-aware IDS [37].

5.1.1.2 Suricata

Suricata is a free and open-source IDS that supports multi-threaded processing and is capable of real-time intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM) and offline pcap processing [38].

Suricata covers more protocols of the application layer, and it supports hashing and file extraction and Lua scripting, which can be used to modify outputs and even create complex and detailed signature detection logic [38].

The Suricata project and code is owned and supported by the Open Information Security Foundation (OISF) [38].

Suricata has several “building blocks”, which are threads, thread-modules, and queues. Suricata is multi-threaded, which means that multiple threads are active at once.

A thread-module is a part of the functionality. Suricata has four thread modules:

Packet acquisition: Reads packets from the network

Decode and stream application layer: Decodes the packets and inspects the application

Detection: Compares signatures and can be run in multiple threads.

Outputs: Processes all the alarms

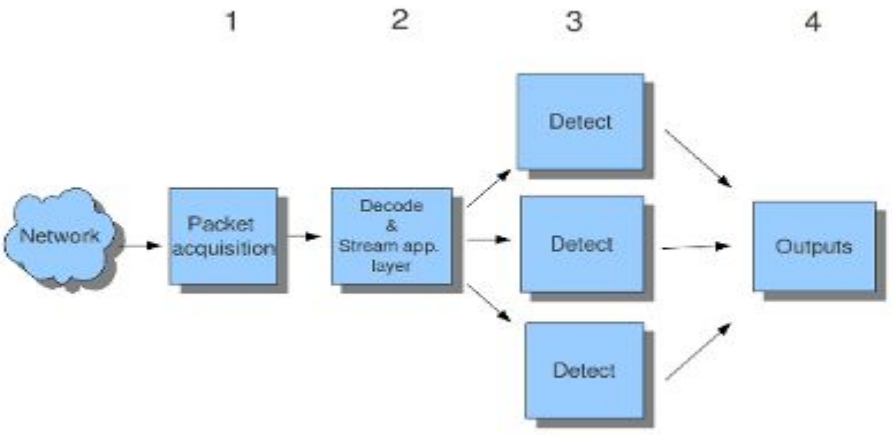

Figure 2. Suricata structure [39]

Packets will be processed only by one thread at the time. However, a thread can process more than one packet at the same time. A thread can have one or more thread-modules, but only one thread module can be active at the time. Runmode is the way threads, modules, and queues are arranged together [39].

Suricata has three run-modes:

workers: Is the recommended runmode as balancing happens in the hardware or driver, which means packets are managed by NIC/driver to make sure they are properly balanced over Suricata's processing threads.

autofp: Is good for pcap file processing and in case of certain IPS setups. In autofp, there are one or more capture threads that capture the packet and do the packet decoding. After this, the packet is passed on to the flow worker threads.

single: Is suitable for use in development. Single runmode is similar to worker runmode with the difference that it works with only the single packet processing thread [40].

5.1.1.3 Zeek (Bro)

Zeek, formerly known as Bro, primarily works as a security monitor tool that can perform an in-depth inspection of the traffic and look for suspicious activity. However, Zeek is also able to perform a wide range of traffic analysis tasks, including non-security tasks analysis, for example, performance measurements and help with

troubleshooting [41]. Zeek also logs all the network's activity in high-level terms. Zeek is aware of the application layer, and therefore the application layer transcripts are also included in the log files. Zeek logs are organized in a tab-separated manner, which enables post-processing with external tools. Zeek provides built-in functionality for a range of analysis and detection tasks, including file extraction from HTTP sessions, interfacing to external registries for malware detection, identifying popular web applications, SSL brute-forcing, SSL certificate validation [41].

Zeek is a traffic analysis platform that is fully customizable and extensible, it supports the traditional signature-based detection methodology, but its strong scripting language can enable other detection approaches like semantic misuse detection, anomaly detection, and behavioral analysis on high-speed and high-volume networks. Architecturally Zeek has two main components, event engine (or core) and policy script interpreter. Event engine reduces the incoming packets into more high-level events. Moreover, these high-level events reflect network activity in policy-neutral terms [42].

However, Zeek is single-threaded, to overcome this limitation, Zeek introduces the clusters. Clusters help to spread the load on several cores. Tap components split the traffic packets, make a copy of the packet for the inspection, frontend splits traffic into many streams or flows to be sent to the worker's components for the protocol analysis. The manager is the component that ensures the centralized log, events, or data [43].

5.1.1.4 General Comparison Summary

Table 1 shows the side-by-side comparison of Snort, Suricata, and Zeek.

As it is also shown in the table, all three IDS have very similar features. They all support real-time inspection, IPv6, capture accelerators, and offline analysis. However, Suricata and Zeek are capable of handling considerably more traffic than Snort.

Besides, Suricata supports multi-thread inception, which can have a great effect on its performance. Overall, Suricata seems to be a better option for a small to medium organization as it supports both IDS and IPS, high network throughput, and multi-thread inspection.

Features	Snort	Suricata	Zeek
Real Time	Yes	Yes	Yes
Throughput	1GB	10GB+	10GB+
IPS capability	Yes	Yes	No
Threats	Single-Threat	Multiple-Threat	Single-Threat
Installation	Simple, also available from packages	Simple, also available from packages	Simple, also available from packages
IPv6 Support	Yes	Yes	Yes
Capture accelerators	Yes (PF-Ring)	Yes (PF-Ring)	Yes (PF-Ring)
offline analysis	Yes	Yes	Yes
License	GNU GPL v.2	GNU GPL v.2	BSD

Table 1. Feature comparison for IDS tools

5.1.2 Literature Review

M. Hänninen has provided an overall evaluation of three open-source IDSs, Snort, Suricata, and Zeek, to find the best solution among them based on the evaluation criteria, which were:

- Security performance out of the box, without a significant work effort;
- Credibility;
- Integration options.

One evaluation criteria is performance out of the box, most of the configuration has been kept in the default state during all the tests. The study has four sub-areas:

1. How well do the IDS detect the variety of network attacks using built-in scripts;
2. How much effort does the configuration of the IDS require;
3. How well is the IDS maintained and developed;
4. What kind of alert outputs are supported.

The study is conducted in a lab environment consisting of three target servers, two clients, all in the same network with NIDS solutions listening traffic in promiscuous

mode. Listening to the traffic in promiscuous mode provides an identical test setting for all three IDS solutions in all the performed tests. Scoring is on a scale of 1-3. Three points are given to the best solution, two to the second-best and one point to the last one. Adding the score of all the tests, the best solution based on defined criteria is Suricata, and the author's conclusion is to a small and medium enterprise, Suricata seems the best option [44].

J. White, T. Fitsimmons and J. Matthews study both default and "out of the box" performance of different versions of Snort and Suricata, investigate alternative configuration and performance improvement and examine them based on scaled resources like CPU cores while focusing on performance and scalability. The study results show that even though both Snort and Suricata may present substantial problems with scalability, Suricata outperforms Snort even in a single core where it was that Snort should perform better. Suricata also shows lower memory usage and CPU utilization.

The research also resulted in some changes to Suricata, as OISF has been very open to communication [45].

M. Pihelgas studies the advantages and disadvantages of the three popular IDSs Snort, Suricata, and Zeek. It compares the IDS performance using dropped packets, memory usage, and CPU usage metrics in 3 approaches:

- Default OS and IDS configuration;
- Optimized IDS configuration;
- Modified or replaced packet capture module.

The results show that all 3 IDS are able to handle 100Mbit/s network traffic by default configuration, however, with optimized configuration, both Zeek and Suricata were able to handle 1000Mbit/s. Zeek had way less memory usage from a memory usage perspective than Suricata; however, from the CPU usage perspective, Suricata has the most efficient performance [46].

G. Khalil studies and compares Suricata, Snort, and Zeek through existing literature and concludes that Suricata and Zeek can be used in networks with 10 GB throughput or more, while Snort is suitable with 1GB throughput. It also suggests that Snort is most suitable for commercial organizations as it has community support and a wide user base. Suricata is the best fit for organizations with high throughputs like ISPs, and Zeek is

excellent for research purposes due to its support for high throughput and powerful scripting support [47].

Table 2 shows the evaluation factors from each research reviewed above in more detail. Overall Suricata represents the best result as all the studied research papers also showed. The green cells in table 2 show the best result in each row.

Research	Evaluation Factors	Snort	Suricata	Zeek
Markku Hänninen 2019	Network attack monitoring	1	3	2
	Configuration	3	3	1
	Software maintenance and development	1	3	3
	Alert Outputs	3	3	1
J. White, T. Fitsimmons, andJ. Matthews	Scalability (PPS)	252,896	258,912	-
M. Pihelgas	Throughput (Optimized IDS configuration) (Mbits/s)	100	100	100
	Throughput (Optimized IDS configuration) (Mbits/s)	-	1000	1000
	CPU	~12.5	Lowest	~12.5
	Memory (Mib)*	2125	6500	2700
G. Khalil	Throughput (GB)	1	10	10
	Best for	commercial organization	ISPs	Research purposes

Table 2. Evaluation of IDS tools

*The average memory usage of all the experiments

5.1.3 Sustainability of Open-source Tools

Open-source software (OSS) is software with its source code released under a license in which the copyright holder allows users to study, change and distribute the software to anyone and for any purpose [48].

Open-source software may be developed in a collaborative public manner; it is a prominent example of open collaboration. While this makes open source projects appealing and exciting, the sustainability of the open-source tool is an important point to be considered before starting to use them.

The open-source guide defines a project as open source when "anybody can view, use, modify, and distribute" the project for any purpose [49]. As this definition suggests, the developer's contributions are a significant factor in open-source project continuity. Weinstock, C & Hissam's study also confirms that a potential developer pool and dedicated developer community is an essential factor for successful open-source software projects [50].

J. Gamalielsson and B. Lundell study the long-term sustainability of the open-source software and software communities involving a fork. By considering developer's commitment as the main factor in the long-term sustainability of an open-source project, they investigate the LibreOffice project, together with the OpenOffice.org project (LibreOffice project is a fork of OpenOffice.org) and Apache OpenOffice projects, and it presents the results from an analysis of first-hand experiences from contributors in the LibreOffice community. It provides insights concerning challenges related to the long-term sustainability of open-source software communities. The insights are based on the analysis of:

- Characterization of the three projects consisting of the history and governance of the projects, the release history, and commits to the SCM and contributing committers over time;
- Developer's commitment by considering developer's commitment level in the different projects under different governance regimes;

- Retention of committers by considering the recruitment of committers over time, the retirement of committers over time, the distribution of commits for committers contributing to different combinations of projects, and the temporal commitment patterns between projects for committers.

The analysis suggests various factors that may cause developers to lose interest in the projects based on reasons like vendor dominance, copyright assignment, lack of influence, lack of fun, and bureaucracy in the project and factors that keep the developers committed to the project as freedom and proper licensing, and personal and emotional attachments. It also shows that it is the most successful project in terms of indicating long-term sustainability due to its success in recruiting and retaining new contributors to its community and establishing sustainable communities [51].

In a more informal approach, S. Wilson suggests code activity, releases, user community, longevity, and ecosystem as the effective factors on sustainability that are investigatable using a web search [52].

Table 3 shows the data extracted from the git for Snort, Suricata, and Zeek. All three IDSs' communities show good contributions and commitment to the projects. Therefore none of these projects are showing signs of non-sustainability.

Metrics	Snort	Suricata	Zeek
Lines of code	323K	392K	148K
Current contributors	13	37	47
Last commit	3 days ago (as of 26.04.2020)	3 days ago (as of 26.04.2020)	5 months ago (as of 26.04.2020)
Number of commits	4245	10947	9970
Users on open hub	87	16	9

Table 3. IDS comparison based on git data

5.2 Malware Statistics

This section reviews malware statistics reports published for 2019 and provides an insight into how malware has been growing and changing.

5.2.1 Proofpoint Q1 2019

Proofpoint publishes quarterly reports that highlight not just the threats but useful takes-aways and methodologies from them based on its database of analyzed emails, social media posts, and malware samples. The goal of the report is to provide actionable knowledge for readers. In the first quarter of 2019, ransomware has been absent in the email-based threats as Emotet has 82% of malicious content. For the web-based threat, Coinhove crypto mining has increased in January and almost zeroed in March. The majority of fraudulent domains had an SSL certificate, which leads to a false sense of security to end users encountering these domains online and in email attacks [53].

5.2.2 Kaspersky lab 2019

In the 2019 report from Kaspersky, banking malware, crypto-ransomware, miners, web-based attacks, and local threat categories are studied. Banking malware, crypto-ransomware, miners statistics are based on data from users of Kaspersky Security Network (KSN), a distributed antivirus network that works with various anti-malware protection components and agreed to provide their information from between November 2018 to October 2019. For each category, the number of users being attacked, the geography of the attacks, and the top malware signature is studied [54].
Web-based attack statistics are derived from web antivirus components that protect users from attempts to download malicious objects from a malicious/infected website or server. It shows geographic information on the countries which the attacks originated from, that had the highest risk of infection, and trending malicious programs.
After the first scan of the system, local threat categories include objects detected on user computers by Kaspersky's file antivirus. These derive from the result of analysis of the statistical data based on antivirus scans of files on the hard drive, and the results of scanning various removable data storages attached [54].

5.2.3 M-Trends 2020

FireEye M-Trends yearly reports are based on the FireEye Mandiant investigations of targeted attack activity conducted between October 1, 2018, and September 30, 2019. Besides attack statistics on the trending malware, the report also provides an analysis of the source of the detected attacks, dwell time, targeted industries, threat techniques, and further related case studies.

Overall, it looks like attackers are growing more adept at working on hybrid environments, which are a combination of cloud architectures and on-premise services, while the attack techniques and tactics do not change dramatically.

Analysis of the dwell time of 56 for 2019 compared to the 78-day global median dwell time for 2018, shows we are getting better in detecting attacks. Back in 2011, the variable of dwell time was 416 days. These trends could be due to organizations developing their detection programs, leading to faster detection, which can also be due to changes in attacker behaviors. Mandiant experts have seen a continued rise in disruptive attacks (such as ransomware and cryptocurrency miners), which often have shorter dwell times than other types.

FireEye often examines the malware family using a disassembler, or decompile tools through a process called reverse engineering, however, the fact of belonging to the same malware family does not necessarily mean that malware within the same family have identical codes.

Based on malware statistics, 41% of the malware families seen this year were never seen before. This also shows that 59% of attacks are using techniques that are known to us.

Moreover, 70% of the samples identified belonged to one of the five most frequently seen families, which are based on open source tools with active development [55].

5.2.4 CheckPoint 2020

CheckPoint's yearly research reviews the previous year’s cyber incidents and gathers key insights about the cyber threat landscape. It first reviews the timeline of major cyber events of 2019 and projection of 2020 and upcoming events like the Olympics of Japan and evolving technologies like 5G. It offers recommendations for security best practices by focusing on prevention rather than detection and remediation.

A critical takeaway from the timeline of 2019 can be that no organization, regardless of their size and mission, would be an exception in the criminal's eyes, and the attacks can be spread throughout an entire nation with highly destructive impact.

Statistics from 2019 also show that 28% of all organizations worldwide were subject to botnet infection. Malware statistics based on the CheckPoint ThreatCloud Cyber Threat Map between January and December 2019 show an overall scope of each malware category and trending malware families globally and by region.

Most categories of trending malware families on the global and regional levels are very close to each other, and there are a few cases where regional lists differ.

Even though the numbers and statistics give us a contentful picture of the threat landscape, it is important to note that numbers and diagrams are not the whole story, as the ransomware category has a relatively small share between malware categories, but they have a severe impact on businesses and nations [56].

5.2.5 Static Reports Summary

The statistics reports are based on data collected by different institutes using different methods, as a result the trending malwares list differs in each report. However there are malware families that are reported by all like Trickbot, Emotet, Dridex, Ramnit and Trickster. Emotet is one of the malware that has been trending for a few years and one interesting point about Emotet is that it is often used to deliver other types of malware as an example Trickbot and IcedID. Thus, the lab incident scenario will be based on Emotet and Trickbot infection, in order to introduce how these two malware work. The summary statistics from the reports are presented in table 4 in appendix 2.

6. Lab Environment

The lab environment is Ubuntu 18.04, a Linux distribution OS installed in a virtual environment. The use of a virtual environment is not compulsory, and it can be a student's preferred one; however, this lab's instructions are based on the Oracle Virtual Box.

In case the virtual environment used by the student is anything except the Oracle Virtual Box, the main known significant difference currently that may occur is the name of the network interface being different from the one mentioned in the lab. Students can clarify the name of the network interface using ifconfig.

The network interface runs in the promiscuous mode to introduce the promiscuous mode during the lab, as it is necessary for the IDS's real-world deployment, as promiscuous mode enables the IDS to be able to capture all the packet transmitted in the network the IDS is protecting. If the device is not running in promiscuous mode, it only captures the packets which have the IDS's IP address, but it will not receive the packets that are being transmitted by other devices within the network.

The minimum hardware requirement for the Ubuntu installation in the virtual environment: RAM: 4GB, Hard Disk: 25GB, CPU: 2GHz dual-core processor.

As the study in the chapter "5.1 IDS comparison and evaluation" suggested, the lab is designed based on Suricata.

The malware that is used in the incident investigation scenario has been chosen according to "5.2 Malware Statistics". In the lab, the student gets a traffic sample that is infected with Emotet and Trickbot and investigates the incident.

The completed lab file is available here, And the lab structure is explained in detail in the following section.

6.1 Lab Structure

The lab is structured in 5 sections that cover each step of the preparation of the virtual environment, details for the installation of IDS from the source, step-by-step configuration, writing a rule for the simulated DDoS attack followed by predetermined questions, investigation of a traffic sample infected with a malware.
Through the lab, each step is explained in detail or an external link is given for further study to ensure that theory concepts are also covered.

6.1.1 Install Ubuntu in the VB and Verify Connectivity

This step covers the installation and setup of the virtual environment to perform the lab on.
This step is optional as any Suricata-compatible Unix operating system can be used, and it is recommended to set up a virtual machine using the methods described as a means to perform the lab in a neutral, tested environment from scratch so that:

- It does not cause any damage to the system being used, as a virtual machine initialized just for executing this lab can easily be wiped/deleted, and all changes occur only inside the virtual system;
- It is the exact same environment that was used to create and test the lab, thus greatly diminishing the chances of any result deviation from the expected result defined in the lab instructions;
- Has no software interference or significant system environment differences that can interfere with the installation and usage of Suricata, as the Ubuntu system is initialized from scratch and vanilla installation is used (as in, all the software included in the system are the default ones shipped with the system that are known not to cause any interference).

This step also outlines the installation of some pre-required tools that will be needed throughout the lab and verifying that this information can be acquired, namely the network configuration with ifconfig command.

6.1.2 Install Suricata and Suricata-Update

This step regards the installation of Suricata and programs required to install and run Suricata and Suricata IDS itself, by downloading to extraction and compiling from source, that is, building Suricata into the system from the compilation of the publicly available source files.

Furthermore, the Suricata-update is installed, which is the official Suricata rule management and basic rule management with the Suricata update.

The rulesets enabled in this step are Emerging threats, Trafficid, and ssl-fp-blacklists.

6.1.3 Configure Suricata with Basic Settings and Run Suricata

This step details the initial configuration and setting up of Suricata for it to run in the system as a background process.

Initially, the IP address of the machine needs to be determined by the student.

The configuration in this step is defining the $HOME_NET with the IP address of the previously determined device, Output types, and the network interface that Suricata listens to.

Outputs types: Fast.log, HTTP logs, DNS logs, eve.json.

Not all of these outputs are needed for this lab, and configuring these logs is to introduce the outputs.

Also, testing the configuration for a check against errors and setting up the necessary procedure to run Suricata as a background app, by enabling the "systemctl" file provided with Suricata source to the Ubuntu system and editing it to be allowed to run.

Running Suricata for the first time and verifying it is running as intended.

6.1.4 Simulate DDoS Attack and Write a Rule

The goal of this step is to cover how rules are written, how the rule matches the traffic, and alerts are generated. First, using hping, DDoS traffic is generated. The student should write the rule that matches the traffic, add this rule to the rule folder and run Suricata and view the generated alert based on the generated traffic.

The rule is not given in the lab, and the rule needs to be written based on the guides and tips provided in the instruction.

6.1.5 Replay the pcap File in Suricata and Wireshark and Analyze the Alerts

This step has a more advanced scenario than the last step.
The main task for this lab is to replay sample traffic, which is infected with Emotet and Trickbot in Suricata, and investigate the traffic.
Several predetermined questions are associated with the infected traffic that need to be answered. These questions include the source of the infection, the infected host, the malicious activities.
Besides, a simple malware analysis task is also included in this part, by extracting the executables file and studying their associated hash.

6.2 Lab Testing and Modification

The lab has been tested, ran, and re-ran by three colleagues that volunteered to test it and execute it from beginning to end and asked to assess a few points regarding the execution of the lab from a third-view referential, such as:

- The approximate time required to complete the lab (not including answering the questions) ;
- Possible problems that may have been encountered during the execution;
- Overall feedback of positive and negative aspects of it and recommendations for improvements.

	Defina	Nathan	Romaine
Duration (Mins)	120	100	110
Error encountered	[27968] 30/4/2020 -- 21:57:21 - (tm-threads.c:2087) <Error> (TmThreadCheckThreadState) -- [ERRCODE: SC_ERR_FATAL(171)] - thread W#01-eth0 failed	Suricata did not start with 'sudo systemctl restart suricata'	'make' failed
User error?/Fixed?	User error	instruction changed to include the fix	more likely user error. The fix is included in the file now
Recommendation	- More explanation about the log location - dns output - Network interface	- Explaining the configuration of the systemctl startup file in case the system does not recognize the parameters provided in the default file from suricata	- clarify if students can choose the their preferred virtual environment

Table 4. Lab test

7 Conclusion

This paper proposed a comprehensive and easy to follow virtual lab for the Cybersecurity Technologies II course, which introduces students to network-based IDS and simple malware analysis.

It first provided a review of the three popular open-source network-based IDS comparisons to determine the backbone IDS. The comparison work is done by side-by-side comparison, literature review and from a sustainability perspective. On the side-by-side comparison Suricata, being multi-thread and supporting 10GB of throughput, is ahead of Snort and Zeek for this work. Same as the side-by-side comparison, the literature review also suggests Suricata as the best option for this work.

Furthermore, this work also reviews the malware statistics reports of trending malware from 2019 to 2020 which shows Emotet and Trickbot among the most spread malware in 2019 and even previous years, and common scenario of Emotet used for delivering other malware including Trickbot to the target device.

The lab, which is a virtual Ubuntu 18.0, covers all steps from setting up the environment, installing and configuring Suricata and testing it against a simulated DDoS attack and investigating an incident scenario based on sample traffic infected with Emotet and Trickbot. Each part of the lab includes a step-by-step and detailed structure and explanation of the reason to make sure the student gains in-depth knowledge while completing the lab. Besides, the external link and resources have been suggested for the lab to support students that may not have background knowledge about the topic or may be interested in more in-depth knowledge. The questions included in the lab challenge the student's knowledge at the end of the lab and can be used to evaluate the student's takeaways by completing the lab.

Three volunteers tested the lab and based on the feedback provided, it takes 100 mins to 120 mins to complete the lab, the lab steps are clear and comprehensive. As it was also stated on this paper, the signature-based detection method used by Suricata is not very effective on the zero-day attacks, thus the possible future work can focus on IDS that uses anomaly-based detection.

References

[1] CyberEdge Group, “2019 Cyberthreat Defense Report,” 2019, [Online]. Available: https://www.imperva.com/resources/reports/CyberEdge-2019-CDR-Report-v1.1.pdf.

[2] J. R. Vacca, *Computer and Information Security Handbook*, vol. 1200. 2013.

[3] K. Scarfone and P. Mell, “Guide to Intrusion Detection and Prevention Systems (IDPS),” *NIST Special Publication 800-94*, 02.2007, [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf.

[4] M. Rouse, “Intrusion detection system (ids),” *TechTarget*, 02.2020. https://searchsecurity.techtarget.com/definition/intrusion-detection-system.

[5] K. Scarfone and P. Mell, *Computer and Information Security Handbook*. Springer, Berlin, Heidelberg, 2010.

[6] P. Gibbs, “Intrusion Detection Evasion Techniques and Case Studies,” *SANS Institute*, Jan. 2017, [Online]. Available: https://www.sans.org/reading-room/whitepapers/detection/intrusion-detection-evasion-techniques-case-studies-37527.

[7] “4.1. Rules Format — Suricata 4.1.0-dev documentation.” https://suricata.readthedocs.io/en/suricata-4.1.4/rules/intro.html (accessed May 14, 2020).

[8] S. Networks, *scirius*. Github.

[9] M. Shirk and PulledPork Team, *pulledpork*. Github.

[10] OISF, “7. Rule Management — Suricata 5.0.2 documentation.” Accessed: May 15, 2020. [Online]. Available: https://suricata.readthedocs.io/en/suricata-5.0.2/rule-management/index.html.

[11] OISF, *suricata-intel-index*. Github.

[12] “malware - Glossary | CSRC,” *Computer Security Resource Center*. https://csrc.nist.gov/glossary/term/malware (accessed May 10, 2020).

[13] “virus - Glossary | CSRC,” *Computer Security Resource Center*. https://csrc.nist.gov/glossary/term/virus (accessed May 10, 2020).

[14] “worm - Glossary | CSRC,” *Computer Security Resource Center*. https://csrc.nist.gov/glossary/term/worm (accessed May 10, 2020).

[15] “rootkit - Glossary | CSRC,” *Computer Security Resource Center*. https://csrc.nist.gov/glossary/term/rootkit (accessed May 10, 2020).

[16] “What is Malware?,” *Malwarebytes*. https://www.malwarebytes.com/malware/ (accessed May 10, 2020).

[17] Industrial Control Systems Emergency Response Team (ICS-CERT) Advanced Analytical Laboratory (AAL), “Malware Trends,” *National Cybersecurity and communications Integration Center*, 10.2016, [Online]. Available: https://www.us-cert.gov/sites/default/files/documents/NCCIC_ICS-CERT_AAL_Malware

_Trends_Paper_S508C.pdf.

[18] R. Sihwail, K. Omar, and K. A. Z. Ariffin, “A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis,” 01.2018, [Online]. Available: https://www.researchgate.net/publication/328760930_A_survey_on_malware_analysis_techniques_static_dynamic_hybrid_and_memory_analysis.

[19] E. Gandotra, D. Bansal, and S. Sofat, “Malware Analysis and Classification: A Survey,” *Journal of Information Security*, vol. 5, no. 2, pp. 56–726, Apr. 2014, Accessed: May 10, 2020. [Online].

[20] “What is an API?,” *Red Hat*. https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces (accessed May 10, 2020).

[21] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, “A Survey on Malware Detection Using Data Mining Techniques,” *ACM Computing Surveys*, vol. 50, no. 3, pp. 1–40, Jun. 2017.

[22] Z. Li, “Control Flow Graph Based Attacks,” 09.2014, [Online]. Available: https://www.diva-portal.org/smash/get/diva2:762870/FULLTEXT01.pdf.

[23] W. B. Ribbens, *Understanding Automotive Electronics*, vol. 710. 2017.

[24] G. A. N. Awad and N. B. I. Ithnin, “Survey on Representation Techniques for Malware Detection System,” vol. 14, no. 11), pp. 1049–1069, Nov. 2017, Accessed: May 16, 2020. [Online].

[25] W. Aman, “A FRAMEWORK FOR ANALYSIS AND COMPARISON OF DYNAMIC MALWARE ANALYSIS TOOLS,” *International Journal of Network Security & Its Applications*, 09.2014, [Online]. Available: https://arxiv.org/pdf/1410.2131.pdf.

[26] M. Egele, T. Schole, E. Kirda, and C. Kruegel, “A Survey on Automated Dynamic Malware Analysis Techniques and Tools,” 2010, [Online]. Available: https://publications.sba-research.org/publications/malware_survey.pdf.

[27] “Cisco Networking Academy Builds IT Skills & Education For Future Careers,” *Network Academy*. http://www.netacad.com (accessed May 03, 2020).

[28] “Malware-Traffic-Analysis.net - Traffic Analysis Exercises,” *Malware Traffic Analysis*. http://malware-traffic-analysis.net/training-exercises.html (accessed May 03, 2020).

[29] “Puzzles! – Network Forensics Puzzle Contest,” *Forensics Contest*. https://forensicscontest.com/puzzles (accessed May 03, 2020).

[30] “2000 DARPA Intrusion Detection Scenario Specific Datasets.” https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-scenario-specific-datasets (accessed May 03, 2020).

[31] F. Zhang, “Lab 8: Firewall & Intrusion Detection Systems,” *Wayne State University*, [Online]. Available: http://webpages.eng.wayne.edu/~fy8421/19sp-csc5290/labs/lab8-instruction.pdf.

[32] R. Hamilton, “Lab 11 Configure an Intrusion Detection System (IDS) for a Control System,” *Cyber Security Education Consortium*, Jul. 2014, [Online]. Available: https://www.nationalcyberwatch.org/ncw-content/uploads/2016/03/ControlSystemsSecurityLab11.pdf.

[33] “What is Snort?,” *Snort*. https://www.snort.org/faq/what-is-snort (accessed May 09, 2020).

[34] Cisco, “SNOR Users Manual 2.9.16.” Apr. 08, 2020, [Online]. Available:

https://www.snort.org/.

[35] “What is the relationship between Snort and Cisco?,” *Snort*. https://www.snort.org/faq/what-is-the-relationship-between-snort-and-cisco (accessed May 09, 2020).

[36] “What is the role of Talos?,” *Snort*. https://www.snort.org/faq/what-is-the-role-of-talos (accessed May 09, 2020).

[37] Cisco, “SNORT Users Manual,” Snort. Accessed: Sep. 05, 2020. [Online]. Available: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/.

[38] OISF, “Suricata User Guide — Suricata 5.0.2-dev documentation.” Accessed: May 09, 2020. [Online]. Available: https://suricata.readthedocs.io/en/suricata-5.0.2/.

[39] OISF, “8.1. Suricata.yaml — Suricata 5.0.2-dev documentation.” Accessed: May 09, 2020. [Online]. Available: https://suricata.readthedocs.io/en/suricata-5.0.2/configuration/suricata-yaml.html?highlight=packet%20acquisition.

[40] OISF, “7.1. Runmodes — Suricata 5.0.2-dev documentation.” Accessed: May 09, 2020. [Online]. Available: https://suricata.readthedocs.io/en/suricata-5.0.2/performance/runmodes.html.

[41] “Introduction — Zeek User Manual v3.1.3.” https://docs.zeek.org/en/current/intro/index.html (accessed May 09, 2020).

[42] “Introduction — Zeek User Manual v3.1.3.” Accessed: May 09, 2020. [Online]. Available: https://docs.zeek.org/en/current/intro/index.html.

[43] “Cluster Architecture — Zeek User Manual v3.1.3.” Accessed: May 09, 2020. [Online]. Available: https://docs.zeek.org/en/current/cluster/index.html.

[44] M. Hänninen, “Open source intrusion detection systems evaluation for small and medium-sized enterprise environments,” Master, School of Technology, 12.2019.

[45] J. S. White, T. Fitsimmons, and J. Matthews, “Quantitative Analysis of Intrusion Detection Systems: Snort and Suricata,” in *ResearchGate*, Apr. 2013, vol. 8757, doi: 10.1117/12.2015616.

[46] M. Pihelgas, “A COMPARATIVE ANALYSIS OF OPENSOURCE INTRUSION DETECTION SYSTEMS,” Master, Faculty of Information Technology, 2012.

[47] G. Khalil, “Open Source IDS High Performance Shootout,” *SANS Institute*, Feb. 2015, [Online]. Available: https://www.sans.org/reading-room/whitepapers/intrusion/open-source-ids-high-performance-shootout-35772.

[48] A. M. St. Laurent, *Understanding Open Source and Free Software Licensing: Guide to Navigating Licensing Issues in Existing & New Software*. “O’Reilly Media, Inc.,” 2004.

[49] “Starting an Open Source Project,” *Open Source Guides*. https://opensource.guide/starting-a-project (accessed May 09, 2020).

[50] C. B. Weinstock and S. A. Hissam, “Making lightning strike twice,” *Perspectives on free and open source software*, pp. 93–106, 2005.

[51] J. Gamalielsson and B. Lundell, “Sustainability of Open Source software communities beyond a fork: How and why has the LibreOffice project evolved?,” *J. Syst. Softw.*, vol. 89, pp. 128–145, Mar. 2014.

[52] S. Wilson, “How to evaluate the sustainability of an open source project,” *OSS Watch*, Nov. 12, 2013. http://oss-watch.ac.uk/resources/evaluatingsustainability (accessed May 09, 2020).

[53] “Cyber Security Threat Report Q1 2019 | Proofpoint,” *Proofpoint*, 2019. https://www.proofpoint.com/us/resources/threat-reports/latest-quarterly-threat-research (accessed May 10, 2020).

[54] AMR, “Kaspersky Security Bulletin 2019. Statistics,” *Securelist*, Dec. 12, 2019. https://securelist.com/kaspersky-security-bulletin-2019-statistics/95475/ (accessed May 10, 2020).

[55] “M-Trends Cyber Security Trends | FireEye,” *FireEye*, 2020. https://www.fireeye.com/current-threats/annual-threat-report/mtrends.html (accessed May 10, 2020).

[56] “Check Point’s 2019 Security Report - Check Point Software,” *Check Point Software*, Mar. 04, 2019. https://blog.checkpoint.com/2019/03/04/check-points-2019-security-report/ (accessed May 10, 2020).

[57] R. Miller, “The OSI Model: An Overview,” *SANS Institute*, Sep. 2001, [Online]. Available: https://www.sans.org/reading-room/whitepapers/standards/osi-model-overview-543.

[58] ISO/IEC, “Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model,” *ISO/IEC*, Nov. 1994, [Online]. Available: https://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip.

[59] N. Ahmad and M. Kashif Habib, “Analysis of Network Security Threats and Vulnerabilities by Development & Implementation of a Security Network Monitoring Solution,” Master, School of Engineering, 09.2010.

[60] F. Glossary, “What is Application Layer Security?,” *F5*. https://www.f5.com/services/resources/glossary/application-layer-security.

[61] “What Is The OSI Model? | Cloudflare,” *Cloudflare*. https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/ (accessed May 04, 2020).

[62] billlattimer, “Windows Network Architecture and the OSI Model - Windows drivers,” *Microsoft*, Apr. 20, 2017. https://docs.microsoft.com/en-us/windows-hardware/drivers/network/windows-network-architecture-and-the-osi-model (accessed May 06, 2020).

[63] K. Holl, “SANS Security Essentials GSEC Practical Assignment v1.4b OSI Defense in Depth to Increase Application Security” *SANS Institute*, 2003, [Online]. Available: https://www.giac.org/paper/gsec/2868/osi-defense-in-depth-increase-application-security/104841.

[64] C. D. McLain, A. Studer, and R. P. Lippmann, “Making Network Intrusion Detection Work With IPsec,” May 2007, Accessed: May 08, 2020. [Online]. Available: https://www.researchgate.net/publication/235180223_Making_Network_Intrusion_Detection_Work_With_IPsec.

[65] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee, “Polymorphic Blending Attacks,” 2006, [Online]. Available: https://www.cc.gatech.edu/fac/Wenke.Lee/papers/usenix_security_2006.pdf.

[66] OISF, “6. Suricata Rules — Suricata 5.0.2-dev documentation.” Accessed: May 18, 2020. [Online]. Available: https://suricata.readthedocs.io/en/suricata-5.0.2/rules/index.html.

[67] “Emotet Malware | CISA,” *Cybersecurity & Infrastructure Security Agency*, Jan. 23, 2020. https://www.us-cert.gov/ncas/alerts/TA18-201A (accessed May 17, 2020).

[68] “Resurgence of EMOTET Malware,” *Trend Micro*, Mar. 13, 2020. https://success.trendmicro.com/solution/1118391-malware-awareness-emotet-resurgence (accessed May 17, 2020).

[69] “Advisory: Trickbot,” *National Cyber Security Center*, Dec. 02, 2020. https://www.ncsc.gov.uk/news/trickbot-advisory (accessed May 17, 2020).

[70] “Security Primer - TrickBot,” *CIS*, 09.2019. https://www.cisecurity.org/white-papers/security-primer-trickbot/ (accessed May 17, 2020).

[68] “Emotet Malware | CISA,” Cybersecurity & Infrastructure Security Agency, Jan. 23, 2020. https://www.us-cert.gov/ncas/alerts/TA18-201A (accessed May 17, 2020).

[69] “Resurgence of EMOTET Malware,” Trend Micro, Mar. 13, 2020. https://success.trendmicro.com/solution/1118391-malware-awareness-emotet-resurgence (accessed May 17, 2020).

[70] “Advisory: Trickbot,” National Cyber Security Center, Dec. 02, 2020. https://www.ncsc.gov.uk/news/trickbot-advisory (accessed May 17, 2020).

[71] “Security Primer - TrickBot,” CIS, 09.2019. https://www.cisecurity.org/white-papers/security-primer-trickbot/ (accessed May 17, 2020).

Appendix 1 – Theory Background

1.1 OSI layers

The Interconnection Reference Model, more commonly known as the OSI model, was first published in 1984 by the International Organization for Standardization (ISO). The OSI reference model serves as an essential element of computer networking. The OSI model's primary purpose is to provide standards for equipment manufacturers, so different computer systems could communicate with each other. The OSI model defines an abstract hierarchical model that logically splits the required functions to support system-to-system communication [57] [58]. The OSI model is composed of the following sections, in a top-down manner:

7. **Application**

   The application layer is the top layer in the OSI model. Being defined as the layer where the network meets end-user programs or application processes, and users access network services. The application layer is the only layer that interacts with the end-user; it serves as an interface with the OS and other applications. However, client software does not belong to the application layer. HTTP and SMTP are examples of application layer protocols [59].

   Denial-of-Service attacks (DoS), HTTP floods, SQL injections, cross-site scripting, parameter tampering, and Slowloris attacks are examples of application-layer attacks [60].

6. **Presentation**

   The presentation layer's primary job is to make the data received from the session layer presentable to the application layer. This data may be encoded in a different method that will need translation, it might be encrypted and needs decryption to be readable by the application layer, or it might be compressed and needs decompression.

In the other direction, the presentation layer might need to translate, encrypt or compress data received from the application layer before sending it to the session layer [61] [59].

5. **Session**

The session is known as the time between when the communication opens and closes between the two devices. Opening the communication, ensuring it stays open, allowing all the data transfer, and closing it is the session layer's responsibility.

The session layer can also set checkpoints in the data transfer to be able to resume the data transfer in case of connection loss or disruption [61] [59].

4. **Transport**

The transport layer takes care of the end to end connection between two devices. The transport layer breaks the data received from the session layer into smaller pieces called segments and sends them to the network layer. On the other direction, when the transport layer receives segments from the network layer, it reassembles them together before sending them to the session layer.

The transport layer is also responsible for flow control, ensuring the optimal speed of the transmission and error control, ensuring that received data is complete (in the inter-network communications) [61] [59].

3. **Network**

The network layer has the responsibility of facilitating data transfer between two different networks. The network layer breaks the segments received from the transport layer into smaller pieces called packets and sends them to the data link layer. On the way back, the packets are reassembled by the network layer into segments and sent to the transport layer [59].

The network layer is also responsible for packet routing, finding the best physical path for the packet to the destination based on network conditions, the priority of service, and other factors [62].

2. **Data Link**

   The data link layer has the responsibility of facilitating data transfer between the same networks. It receives the packets from the network layer, breaks them into smaller pieces called frames, and sends them to the physical layer, on the way back the received frames will be reassembled to packets and sent to the network layer.

   Data link layer is also responsible for the intra-network communication flow control and error control [62].

   ARP spoofing, MAC Flooding, spanning tree attack are some of the threats for the data link layer [63].

1. **Physical**

   The physical layer converts the frames from the link layer to raw bit-streams sent out via physical equipment like switches and wires. In the opposite direction, It also converts the received bit-streams to the frames and sends them to the data link layers [62].

1.2 Evasion techniques for IDS

The primary purpose of IDS is to predict the network's state and the devices within the network. IDS evasion techniques aim to prevent the IDS from predicting the state of devices and desynchronize the IDS and devices in the network [6].

1.2.1 Time-To-Live Manipulation

Time-To-Live (TTL) field in the Internet Protocol (IP) header represents the limit of hops. The TTL value is set in the origin of the packet according to the packet destination. When a router receives an IP packet, the router will decrement the TTL value by one, if the resulting TTL value is zero, the router discards the packet, but if the resulting TTL value is greater than zero, the router forwards the packet, according to its routing table, to the next hop which could be another router or the final destination. If the next hob is a router, the same procedure happens again until the packet reaches its final destination.

The synchronization of the IDS and the end hosts depend on a router being between the IDS sensor and the end host, and TTL attacks are attacks that interrupt this synchronization.

Attackers can determine hop counts to the router and end-host using network reconnaissance tools. If TTL is equal to the number of hops of the router, the packet gets examined by the IDS, but it does not reach the end host, desynchronizing the end host and the IDS. NIDS that understands network topology can detect these attacks. However, this requires additional processing resources [6].

1.2.2 IDS MAC Address Attack

On an Ethernet-based local area network (LAN), layer two switches usually forward traffic based on the MAC address specified in the frame header to its designated host. In this way, hosts only receive the frames addressed to them, and frames addressed for IDS are only visible to IDS.

When IP traffic enters LAN, the IP header and payload gets encapsulated in the Ethernet frame. If the IDS only examine the IP header and ignore the MAC address, it is possible to perform MAC address attacks by sending an ethernet frame containing the MAC address of the IDS and IP address of the end host, which means the IDS receive the frame while the end host did not receive the frame; thus the IDS loses the synchronization with the end host [6].

1.2.3 IP Fragmentation Attacks

LANs mostly use the Ethernet protocol as the data link layer protocol. Ethernet frames can have a maximum size of 1518 bytes.

In the case of networks that use IP over ethernet, the Ethernet frame encapsulates IP Packets that can be up to 65,535 bytes. The IP packets that exceed the ethernet frame maximum size can be transmitted using IP fragmentation. IP fragmentation adds three new fields to the header:

- **Flags:** Fragmentation allowance
- **Offset:** The fragment position in the reconstructed packet
- **Identification**: The packet that the fragment belongs to

IP fragmentation attacks have two types:

Reassembly Attacks are attacks in which IDS and end-host use different reassembly methods for the overlapping fragmented packets received (IDS and end-host desynchronization).

Do Not Fragment Attacks are possible in a network in which IDS positions itself between two routers that use different MTUs.

MTU, the Maximum Transmission Unit, defines the largest amount of data (PDU) that can transmit on a link. Routers examine the MTU of the link the packet should be sent to and discard the packet if the packet size is larger than the link's MTU, and the packet does not have a fragment flag set. When the IDS is in between two routers with different MTUs, IDS can not determine if the packet is discarded or received by the destination (IDS and end-host desynchronization) [6].

1.2.4 Encryption

Hosts can communicate using IPsec or other forms of encryption; this would leave IDS with no option to inspect the traffic unless the IDS gets the decryption keys [64]. The encrypted traffic can contain malware and other malicious content.

Using a bitwise XOR function to encrypt is one of the methods used for encrypting malware. Typical hardware architecture supports XOR encryption and is computationally fast; however, in some cases, NIDS can decrypt the traffic.

Entropy measurements can be used for the IPSec encrypted traffic; compressed data can have similar levels of entropy, leading to false positives [6].

1.2.5 Polymorphic Blending Attack

Signature-based IDS detects the attacks by comparing traffic to predefined patterns. Attackers can use code transforming techniques like encryption, byte substitution, and code obfuscation to create several instances of the same malware that look different but perform the same malicious function to evade the IDS, which is called Polymorphic Attack [65].

Anomaly-based IDS often can detect polymorphic attacks. They develop statistical and heuristic profiles through the learning phase. The profile contains byte frequency

distribution, byte sequence occurrence, packet lengths, and character distribution. The IDS alerts as soon as the traffic patterns exceed the configured threshold.

Polymorphic blending attacks are attacks designed to evade the anomaly-based IDS by matching the network profile. In order to perform polymorphic blending attacks, the malware monitors the target network to develop a normal representative profile for the network, once the profile is developed, the malware generates polyphonic instances using techniques like byte padding and ciphering with different length keys between the variants to blend and make the detection more difficult [6].

1.3 Rules Options

1.3.1 Meta-settings

Meta-settings do not affect Suricata's inspection; they affect the way Suricata reports events by providing more information about the inspected packet.

Possible meta-settings are *msg* (message), *sid* (signature ID), *rev* (revision), *gid* (group ID), *classtype, reference, priority, metadata,target* [66].

1.3.2 IP and TCP Keywords

IP keywords are properties that are related to IP or IP options in the header of the packet, and it can be *TTL* (Time-to-live), *ipopts*, *sameip*, *ip_proto*, *Id*, *geoip*, *fragbits* (IP fragmentation), *fragoffset* and *tos* [7].

TCP keywords are the options that can be set according to TCP protocol properties, and they can be *seq* (TCP sequence number), *ack* (specific TCP acknowledgement number), and *window* (specific TCP window size) [7].

1.3.3 ICMP Keywords

ICMP keywords are the options that can be set according to TCP protocol properties, which can be *itype* (specific ICMP type/number), *icode* (ICMP code), *icmp_id* (ICMP id-value), *icmp_seq* (ICMP sequence number) [7].

1.3.4 Payload Keywords

Payload keywords inspects the content of the packet's payload, these options can be *content*, *nocase* (distinction between upper and lower case), *depth*, *startswith*, *offset*, *distance*, *within*, *isdataat*, *dsize* (size of the packet payload), *replace*, *pcre* (Perl Compatible Regular Expressions) [7].

1.3.5 Flow Keywords

All packets having the same Tuple (protocol, source IP, destination IP, source-port, destination-port), belong to the same flow and are internally connected.
The options that are based on the flow properties are: *flowbits*, *flow* (direction of the flow), *flowint*, *stream_size [7]*.

1.3.6 HTTP Keywords

HTTP keywords are HTTP protocol-based keywords that inspect the data transmitted for HTTP packet, both request and response. These keywords can be *HTTP Primer, http_method, http_uri and http_raw_uri, urilen, http_protocol, http_cookie, http_client_body* and much more[7].

1.3.7 File Keywords

Suricata can extract files from the traffic. Several keywords can be set in a rule to match different file properties. These keywords can be filename, file ext (file extension), *filemagic* (libmagic information), *filestore* (store the file in disk), *filemd5* (MD5 checksums), *filesha1* (SHA1 checksums), *filesha256* (SHA256 checksums) and *filesize* [7].

1.3.8 SSL/TLS and SSH Keywords

Suricata have several keywords that can match the properties of TLS/SSL handshake, as tls_cert_subject (TLS/SSL certificate Subject field), tls_cert_issuer (TLS/SSL certificate Issuer field), tls_cert_serial (serial number in a certificate), tls_cert_fingerprint (certificate's SHA-1 fingerprint), tls_cert_expired, and ssl_version, tls.subject [7].

SSH keywords can match the properties of SSH connections, these keyboards ssh_proto (SSH protocol), ssh_version, ssh.protoversion, ssh.software version [7].

Appendix 2 – Analysis

2.1 Malware Statistics Summary

Checkpoint	**Malware Category:** Crypto Miners 38% Botnet 28% Mobile Malware 27% Banking 18% Infostealer 18% Ransomware 7%	**Banking Trojans:** Trickbot 32% Ramnit 19% Ursnif 8% Danabot 6% Dridex 6% Qbot 3% other 26%	**Info stealer:** Lokibot 17% AgnetTesla 16% Hawkeye 14% Formbook 12% Pony 9% Nanocore 8% Other 33%
	Overall top families: Emotet 18% Jsecoin 15% XMRig 14% Cryptoloot 14% Coinhive 12% Trickbot 11% Lokibot 10% Agent Tesla 10% Hawkeye 8% Formbook 7%	**Cryptomining:** Jsecoin 22% XMRig 21% Cryptoloot 21% Coinhive 18% WannaMine 3% Rubyminer 2% other 13%	**Mobile Malwares:** Hiddad 21% xHelper 16% Necro 10% AndroidBauts 10% Guerilla 9% other 41%
Kaspersky labs	**Banking Malware:** zbot 23.10% RTM 21.60 % EMOTET 12.30% SpyEye 7.10% Nymaim 5.80% Trickster 4.80% Ramnit 4.40% Neurevt 3.10% CryptoShuffler 1.90% Danabot 1.30%	**Crypto-ransomware:** WannaCry 23.56% phny 16.81% GandCrypt 12.17% Gen 6.26% Crypmod 5.08% Encoder 4.65% Shade 2.66% PolyRansome 2.43% Crypren 2.28% Stop 1.94%	**Miners:** Trojan.Win32.Miner.bbb 13.45% Trojan.Win32.Miner.ays 11.35% Trojan.JS.Miner.m 11.12% Trojan.Win32.Miner.gen 9.32%
	Web-based attacks: Malicious URL Trojan.Script.Generic Trojan.Script.Miner.gen Trojan-Clicker.HTML.Iframe.dg Trojan.BAT.Miner.gen Trojan-Downloader.JS.Inor.a Trojan.PDF.Badur.gen DangerousObject.Multi.Generic 0.21 Trojan-Downloader.Script.Generic 0.17 Trojan-PSW.Script.Generic 0.15 Trojan.Script.Agent.gen 0.15	**Local threats:** DangerousObject.Multi.Generic 26.43 Trojan.Multi.BroSubsc.gen 9.48 Trojan.Script.Generic 6.19 Trojan.Multi.GenAutorunReg.a 5.94 HackTool.Win64.HackKMS.b 4.40	

Proof point Q1 2019	**Malware Category:** Botnet 61% Banking 21% Credential Stealer 9% Downloader 7% RAT 1% Keylogger 1% Ransomware 0%	**Banking malware:** IcedID 44% The Trick 24% QBot 18% Ursnif 9% Ramnit 3% UrlZone Banker 2% Dridex 0% DanaBot 0% other 0%	
FireEye	**Malware Category:** Backdoor 46% Dropper 15% Credential Stealer 9% Ransomware 7% POS 7% other 16%	**Overall top families:** Bacon 19% Empire 14% TrickBot 13% ShortBench 13% QakBot 11% other 30%	

Table 4. Summary of trending malware based on report [53][54][55][56]

2.2 Malware Case Study

Based on the malware statistics reports, the lab investigation scenario is based on Emotet and Trickbot. This section introduces how Emotet and Trickbot deliver themselves into the victim device and its impact.

2.2.1 Emotet

First appearing in 2014, Emotet is one of the most costly and destructive malware affecting all the public and private industries, even the government. Emotet infections have cost state, local, tribal, and territorial(SLTT) governments up to $1 million per incident [67].

Emotet is an advanced, modular trojan that initially behaves as a downloader or dropper of other banking trojans. It can also show polymorphic behavior that can evade typical signature-based detection using auto-start registry keys and services and Dynamic Link Libraries (DLLs) to continuously evolve and update its capabilities.

Emotet spread through spam or phishing emails that contain malicious links or attachments (PDF or macro-enabled Microsoft word). The initial infection begins with an opening malicious link, file, or macro-enabled Microsoft Word document included in the email. As soon as Emotet is downloaded, it establishes persistence by injecting code into explorer.exe and other running processes. It is also able to collect valuable

information like OS version, location, system name. After that, it usually connects to a remote C&C server, and reports a new infection, uploads the collected data, download and run file, receives instruction or configuration. The C&C server domain name usually is a 16-letter name that ends in ".eu."

Emotet instances often show paths like AppData\Local or AppData\Roaming directories.

Also, Emotet creates randomly-named files in the system root directories can run as Windows services and propagate the malware to adjacent systems via accessible administrative shares.

Emotete's impact [68]:

- Other dangerous malware groups can rent Emotet-infected devices for delivering more malicious payloads.
- Stealing device sensitive data like system name, OS
- Stealing financial and banking information or user credentials
- Executing backdoor command to connect to other malicious websites

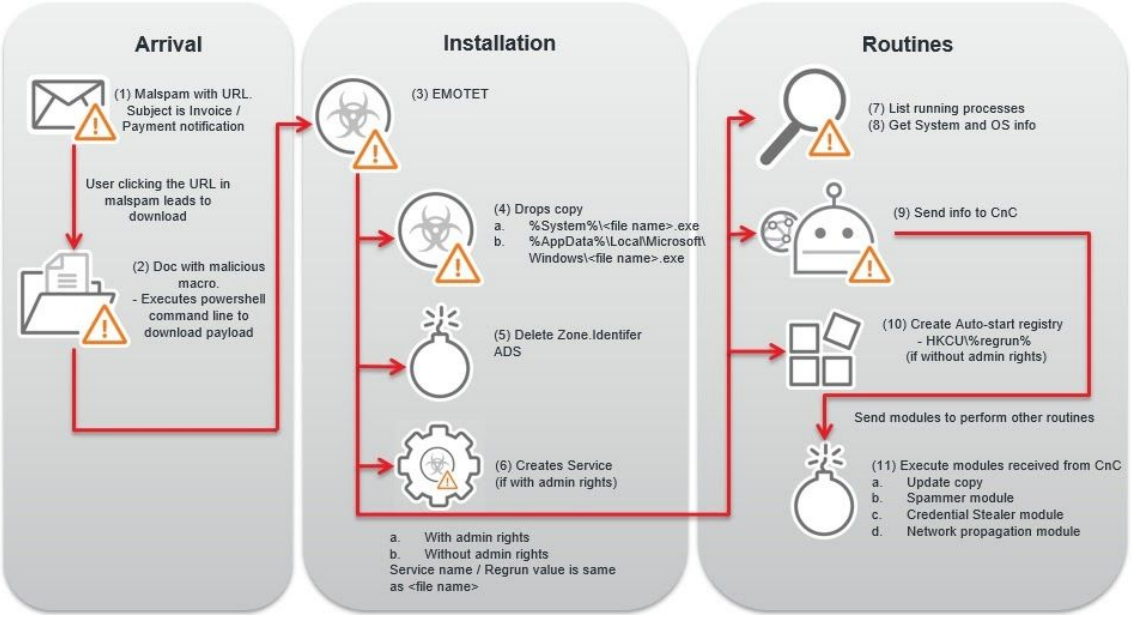

Figure 3. Emotet infection [68]

2.2.2 Trickbot

Trickbot is an establishment that often spreads through well-crafted phishing emails, designed to appear as from trusted commercial or government brands. It can also be delivered as one of the Emotet post infections. It is a banking Trojan that affects both businesses and individuals.

Trickbot's goal is obtaining online accounts and including bank accounts to access personally identifiable information that can be used in identity fraud. It can also be used for delivering other malware, including ransomware and exploit kits [69].

The same as Emotet, Trickbot can:

- Steal sensitive personal or financial information including banking login credentials
- Obtain detailed information about the infected host and the network
- Steal online credential and data including browsing history and cookies
- Connecting an infected host to the malicious networks by executing backdoors and giving full control of the device to criminal third-parties
- Spread through the whole victim's network using SMB shares
- Download further malicious files or other malware [70]