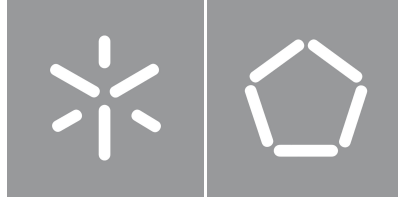




University of Minho
School of Engineering

Afonso Fernando da Costa

Anomaly-Based Intrusion Detection Systems for Industrial Networks



University of Minho
School of Engineering

Afonso Fernando da Costa

**Anomaly-Based Intrusion
Detection Systems for Industrial Networks**

Masters Dissertation

Master's in Engineering of Computer
Networks and Telematic Services

Dissertation supervised by
Henrique Manuel Dinis dos Santos

Copyright and Terms of Use for Third Party Work

This dissertation reports on academic work that can be used by third parties as long as the internationally accepted standards and good practices are respected concerning copyright and related rights. This work can thereafter be used under the terms established in the license below.

Readers needing authorization conditions not provided for in the indicated licensing should contact the author through the RepositóriUM of the University of Minho.

License granted to users of this work:



CC BY-NC-SA

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Acknowledgements

The end of one journey marks the beginning of another, so this work represents the end of a difficult and very important journey in my life. This way I would like to express my trust and respect to all those who have always been by my side.

Express gratitude to Professor Dr. Henrique Santos, supervisor of the dissertation, for sharing knowledge, availability, support and for all his role played throughout the process of investigation.

My gratitude to my parents Miguel and Isabel for their support during the most difficult times, and to other members of my family Deolinda, Asahel, Hélia, Feliciano, Pedro, Dérito, António and João for their affection, comfort, and encouragement.

My gratitude to the Angolan Armed Forces, for its commitment to training and qualification of human resources for the country and to the Board of Higher Military Technical Institute of Angola for their support in all stages and the completion of this work.

I'm deeply grateful to Brigadier General of the Army, Lito Bundi, for his exceptional guidance, unwavering support, and remarkable patience throughout this journey and I'm honored to have had the privilege of his accompaniment.

I would like to express my sincere gratitude to the University of Minho, particularly to the esteemed MERS-TEL Professors, their commitment to excellence has been instrumental in shaping my academic journey.

A word to my colleagues and friends, especially those who dealt with me day by day, my thanks for the friendship, mutual assistance, camaraderie, and joyful moments we have shared.

To all those who have directly or indirectly played a part in this remarkable journey, I want to extend my heartfelt gratitude. Your contributions, whether big or small, have been invaluable, and I am deeply appreciative of the support and encouragement you have provided. Thank you for being an integral part of this meaningful experience.

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, Braga, May 2024

Abstract

The combination of operational technology of industrial networks with information technologies has enabled the increase of attacks on industrial networks, causing professionals and researchers linked to the security area to study tools, mechanisms and techniques capable of detecting and blocking malicious activities in industrial network environments.

This dissertation sets out to comprehensively explore and analyze Anomaly-based Intrusion Detection Systems (IDS) as a central focal point. The primary ambition is to meticulously investigate the efficacy of such systems in identifying, monitoring, and recording abnormal behaviors, detecting malicious activities, and pinpointing potential attacks that exploit remote services and orchestrate denial of service incidents. Through an in-depth examination and critical evaluation, this study aims to contribute to the existing body of knowledge in the realm of cybersecurity, advancing our understanding of IDS capabilities and their significance in safeguarding digital and industrial environments against emerging threats.

In this way, it is intended to create a model relating the attack techniques, their indicators, and the fields, logs, and events generated by the IDS, in order to help detect such attacks, in a way, to evaluate the efficiency of the IDS in detecting malicious activities.

Keywords: intrusion detection system, security in industrial networks, denial of service, brute force.

Resumo

A combinação da tecnologia operacional de redes industriais com as tecnologias da informação tem possibilitado o aumento de ataques a redes industriais, fazendo com que profissionais e pesquisadores ligados à área de segurança estudem ferramentas, mecanismos e técnicas capazes de detectar e bloquear atividades maliciosas em ambientes de redes industriais.

Esta dissertação se propõe a explorar e analisar os Sistemas de Detecção de Intrusão (IDS) baseados em Anomalias como um ponto focal central. A principal ambição é investigar meticulosamente a eficácia de tais sistemas na identificação, monitoramento e registro de comportamentos anormais, detecção de atividades maliciosas e identificação de possíveis ataques que exploram serviços remotos e orquestram incidentes de negação de serviço. Através de um exame aprofundado e avaliação crítica, este estudo visa contribuir para o corpo de conhecimento existente no domínio da segurança cibernética, avançando nossa compreensão dos recursos de IDS e sua importância na proteção de ambientes digitais e industriais contra ameaças emergentes.

Desta forma, pretende-se criar um modelo relacionando as técnicas de ataque, seus indicadores e os campos, logs e eventos gerados pelo IDS, a fim de detectar tais ataques, de forma a avaliar a eficiência do IDS na detecção de atividades maliciosas.

Palavras-chave: sistema de detecção de intrusão, segurança em redes industriais, negação de serviço, força bruta.

Contents

- 1 Introduction 1**
 - 1.1 Motivation and Challenges 2
 - 1.2 Objectives and Expected Results 3
 - 1.3 Research Methodology 3
 - 1.4 Structure of Thesis 4

- 2 State of the Art 5**
 - 2.1 Definition 5
 - 2.2 Host-Based IDS and Network-Based IDS 5
 - 2.3 Intrusion Detection Techniques 5
 - 2.3.1 Anomaly-based detection principles 6
 - 2.4 Industrial Networks 7
 - 2.4.1 Definition 7
 - 2.4.2 Industrial Networks Components 8
 - 2.4.3 Operational Technology and Information Technology 8
 - 2.4.4 Industrial Networks Security 9
 - 2.4.5 Security Thread Model in Industrial Control System 10
 - 2.4.6 Industrial Network Protocols 10
 - 2.5 Related Work 12
 - 2.6 Conclusions 16

- 3 The Proposed Prototype 18**
 - 3.1 MITRE ATT&CK 18
 - 3.1.1 MITRE ATT & CK - ICS Technology Domain 20
 - 3.1.2 Process Control Systems (PCS) 20
 - 3.1.3 Safety Instrumented Systems (SIS) and Protection Systems 21
 - 3.1.4 Engineering and Maintenance Systems 21
 - 3.2 Remote Services 23
 - 3.2.1 Secure Shell SSH 24
 - 3.2.2 Remote Desktop Protocol - RDP 25

3.3	Network Security Monitoring Tools	25
3.3.1	Security Onion	26
3.3.2	Suricata	26
3.3.3	Zeek	27
3.3.4	Elasticsearch, Logstash, and Kibana	32
3.4	Correlation Matrix of Zeek Logs and Attacks Techniques	34
3.5	Detection Framework	35
3.6	Prototype Implementation	37
3.6.1	DOS Syn Flood Attack Detection	37
3.6.2	DOS HTTP Flood Attack Detection	38
3.6.3	Brute Force SSH Attack Detection	39
4	Experimental Tests, Results and Analysis	41
4.1	Experimental Environment	41
4.2	Topology	42
4.3	Tools	43
4.3.1	Hping3	43
4.3.2	SlowHTTPTest	44
4.3.3	Ncrack	45
4.4	Test Methodology	45
4.5	Attack Techniques	46
4.5.1	SYN Flood Test - T1499.001	46
4.5.2	DOS HTTP Flood Test - T1499.002	47
4.5.3	Brute Force Test - T1110	48
4.6	Final results, Evaluation, and Conclusions	49
5	Conclusions and future work	53
5.1	Conclusions	53
5.2	Limitations and Difficulties	54
5.3	Prospect for Future Work	54
	References	55

I	Appendices	57
A	Detection Details	58
A.1	Module icsha syn flood	58
A.2	Module icsha http methods	59
A.3	Module icsha brute force ssh	60
B	Details of Tests	61
B.1	Security Onion configuration to integrate Zeek script	61
B.2	Datasets	62
B.2.1	Normal behavior dataset	62
B.2.2	CICIDS201 Dataset	63

List of Figures

- 1 Industrial networks simple design 7
- 2 The ATT&CK matrix for ISC 18
- 3 The ATT&CK correlation model 19
- 4 Secure shell protocol communication 24
- 5 RDP stack and communication 25
- 6 IDS architecture main components 25
- 7 Security onion high-level architecture Diagram 26
- 8 Security onion dashboard 27
- 9 Suricata alert 27
- 10 Zeek architecture 29
- 11 Zeek Control 32
- 12 Example of jq 33
- 13 Statistical packet time arrival anomaly detection 35
- 14 Architecture of detection framework 36
- 15 Syn flood attack example 37
- 16 Syn flood detection example 37
- 17 HTTP Get and Post flood attack detection example 38
- 18 SSH brute force detection example 39

- 19 Topology used for tests in a controlled environment 42
- 20 Example command of hping3 43
- 21 Example command of slowhttptest 44
- 22 Syn detection log file 46
- 23 SYN flood detection log 47
- 24 DOS HTTP detection log 48
- 25 Zeek brute force detection log 49
- 26 Number of syn flood alerts in experimental tests 50
- 27 Number of HTTP get and post-flood alerts in experimental tests 51

- 28 Syn flood detection code 58

29	Http flood detection code	59
30	Brute force detection code	60
31	Custom scripts folder	61
32	Loaded scripts	61
33	Normal behavior syn ack interval	62
34	Normal behavior packet per second	62

List of Tables

- 1 Non-time series classification 6
- 2 Correlation Matrix of Zeek Logs and Attacks Techniques 34
- 3 Technical specifications of equipments 41
- 4 Syn flood test parameters 46
- 5 Syn flood number of packets 46
- 6 DOS HTTP Test parameters 47
- 7 DOS HTTP test values 48
- 8 Experimental test results 50
- 9 Zeek and Suricata attacks detection of CICIDS2017 dataset. 52
- 10 CICIDS201 dataset information 63

Acronyms

CNN Convolutional Neural Networks

DNP3 Distributed Network Protocol

FTP File Transfer Protocol

GMM Gaussian Mixture Model

HIDS Host-Based Intrusion Detection System

HMI Human Machine Interface

HSMM hidden semi-Markov model

HTTP Hypertext Transfer Protocol

ICS Industrial Control Systems

IDS Intrusion Detection System

IEC International Electrotechnical Commission

IT Information Technologies

LSTM Long Short-Term Memory

Modbus Modicon Communication Bus

NIDS Network-based Intrusion Detection System

NSM Network Security Monitor

OT Operational Technology

PLC Programmable Logic Controllers

RTU Remote Terminal Unit

SCADA Supervisory Control and Data Acquisition

SIEM Security Information and Event Management

SSH Secure Shell

SW Sliding Window

TCP/IP Transmission Control Protocol/Internet Protocol

TTP Tactics Techniques and Procedures

UDP User Datagram Protocol

Chapter 1

Introduction

Industrial control systems (ICS) are specialized software and hardware solutions used to manage industrial environment. These systems are implemented through the application of Supervisory Control and Data Acquisition (SCADA), which comprises sensors, Programmable Logic Controllers (PLC), Remote Terminal Unit (RTU), and Human Machine Interface (HMI) enabling them to control and monitor processed data infrastructure. These components find widespread employment in industrial critical infrastructure environments, such as the oil and petrochemical industries, transportation systems, factories, nuclear power plants, and energy data acquisition (Zhanwei & Zenghui, 2019).

The integration of information technology with operational technology has revolutionized Industrial Control Systems (ICS) resulting in significant economic and operational advantages. This is primarily due to the enhanced communication capabilities and data exchange between SCADA components through protocols such as Modicon Communication Bus (Modbus), Distributed Network Protocol (DNP3), and Profibus, now operating over Transmission Control Protocol/Internet Protocol (TCP/IP). These advancements lead to improve efficiency, streamlined operations, and cost-effectiveness in ICS (Sheng, Yao, Fu, & Yang, 2021; Das, Adepu, & Zhou, 2020).

Historically, these systems have been built to provide high levels of safety and reliability, but the integration with the modern environment called cyber-physical system increase the exposure of ICS to computer network-based attack. Nowadays ICS communications have a typical predefined packet format but intruders can take advantage of it by modifying packet characteristics in a network and perform different malicious activities. Numerous attacks have been detected against ICS, some of them are BlackEnergy, Night Dragon, Duqu, Australia in 2000, Iran in 2010 ("Stuxnet" virus) and Queensland. (Sheng et al., 2021).

1.1 Motivation and Challenges

Industrial networks are responsible for continuous and batch processing and other manufacturing operations of almost every scale, in the last few years, Industrial Ethernet and TCP/IP, smart technologies have been applied to traditional ICS bringing great benefits to the field and improving remote control, efficiency, and high throughput, but this evolution and integration between the cyber and physical control world opened a space to different types of attacks and as a result, the successful penetration of a control system network via remote services such as Secure Shell (SSH) or RDP, can be used to impact those operations directly. The authors in (J. Liu, Yin, Hu, Lv, & Sun, 2018) present problems of the insecure communication process, network segregation and access control.

Nowadays, we have problems related with industrial networks, such as: 1) ICS-specific communication protocols, legacy systems, and limited resources; 2) The lack of real-world ICS datasets; 3) anomaly detection for ICS can't depend exclusively on network protocol, physical process control information is also necessary; 4) The fact of physical process control variable noise that can result in false positives rates; 5) Fixed business logic; 6) Use of signature specifications to detect abnormal behavior which requires human intervention; 7) problems to detect unknown attacks and 0-day attack in multidomain environment (Feng, Li, & Chana, 2017; Zang, Gong, & Hu, 2019).

Furthermore, we can imagine for example: if the signature of a type of attack does not exist in the database of a firewall; if the attacker develops sophisticated methods to bypass existing defenses and successfully carry out an attack on the network or even the possibility of attacks known as "zero-day attack". In this study, we intend to overcome some of the difficulties and particularities mentioned, and also identify the true capacity of network security monitors to detect anomalous behavior. In addition to the possibility of building an isolated, safe, and efficient test execution environment. Such circumstances motivate us to study detection mechanisms based on network anomalies to identify malicious behavior and unrecorded attacks and contribute in the detection of ICS network threats that can compromise ICS availability and integrity

1.2 Objectives and Expected Results

The aim of this work is to study Anomaly-based Intrusion Detection System (IDS) in order to detect attacks in industrial environments based on an open-source platform and focusing the exploration of remote services and DoS.

The detection is based on a model that correlates the logs generated by the IDS during the capture of network traffic, the attacks, and the indicators of the occurrence of an attack, based on this determines the events and logs that the intrusion detection system should trigger. In addition to the previously mentioned objectives, the study also aims to achieve the following:

- Undertake a literature review of the subject matter to identify and analyze the studies already carried out;
- Implement, and evaluate the use of IDS based on behavioral analysis in a simulated network environment with the purpose of malicious activities detention and Network Security Monitoring;
- Build a prototype model of anomaly-based detention for industrial network as a baseline to detect anomalous activities;
- Implement the model in a Network Security Monitor (NSM) tool such as security onion and use zeek as IDS log-based to evaluate.
- Finally, perform different types of malicious activities, such as DoS (HTTP flood, TCP syn), explore remote services vulnerabilities such as ssh, and analyze the effectiveness of the model.

1.3 Research Methodology

To achieve the objectives of this work, we started by carrying out a bibliographic review on the subject, studying significant approaches carried out on the implementation of anomaly-based IDS for industrial network environments, characteristics, methodologies, identification of the problem, motivation, and main challenges. After surveying the main concepts, processes, and techniques, as well as identifying different mechanisms that can be applied to the detection of intrusions in environments of industrial networks, it is intended to develop an experimental study and final simulations, ending with the analysis of the results and the conclusions.

1.4 Structure of Thesis

Beyond this introduction where thesis objectives, methodology, and motivations are presented, this work proceeds with state of art chapter 2, where we identified significant studies with the theoretical approaches about Host-Based IDS and Network-based IDS, Anomaly-based detection principles, behavioral analysis of network traffic and topics related with Industrial Networks components, operations, and security. Chapter 3 presents the used network security monitoring tools, In addition, we present a matrix of IDS logs and attack techniques and prototype detection methodology used in IDS tool. Chapter 4 is dedicated to the experimental tests and analysis of the obtained results. This chapter is subsequently followed by conclusions and an exploration of potential future research directions.

Chapter 2

State of the Art

2.1 Definition

An intrusion detection system is considered a sequence of related actions to detect malicious activity that may compromise the network, a device or a system. It is also considered as a process of audit data, which means collect, identify, and store information about the network traffic, and allow to perform investigations into alarm responding to any type of malicious activities. (J. Liu et al., 2018; Feng et al., 2017).

2.2 Host-Based IDS and Network-Based IDS

The Host-Based Intrusion Detection System (HIDS) use a specialized layer of security software to monitor and analyze suspicious activity in network traffic or logs produced by the host (database server or administrative systems), it can detect internal or external malicious activity and cover signature or anomaly-based approach. Network-based Intrusion Detection System (NIDS) detects malicious activities or abnormal behavior by monitoring and analyzing network traffic changes. It includes a deep packet-by-packet analysis in real-time or close to it. The analysis can be performed in different layers, such as application or transport or network (Kruegel, Valeur, & Vigna, 2005; Stallings & Brown, 2011).

2.3 Intrusion Detection Techniques

According to (Kruegel et al., 2005), IDS detection can be classified into two major techniques, first misuse-based detection usually called signature-based or Heuristic, where there is a database of a set of known attack signatures (Heuristics) that are compared with the current network traffic signature. (Axelsson, 2000) considered that it is a pre-model of the intrusion steps with imprints to search on network traffic, if it matches any IDS alert is generated.

The second is called anomaly-based detection, this technique is characterized by the capture of network traffic behavior of users over a period of time. Here, we analyze the current network behavior, define a model of the expected behavior of the traffic and search for anomalous traffic, meaning usual or unusual

event flows that deviate from the typical characteristic of traffic behavior on the network (Tapiador, García-Teodoro, & Díaz-Verdejo, 2004; Stallings & Brown, 2011). It is also defined as a planned or involuntary event in network traffic flows that differ from considered normal in the context of the network and could result in the congestion of the network or the interruption of the availability of resources. These unpredictable occurrences cause deterioration of network performance including the consequences of spurious traffic caused by network failures, or distrustful behaviors such as network scans for vulnerable ports/services, attacks such as TCP SYN flooding, and DDoS amplification attacks (Zang et al., 2019).

It is considered a normal state of a network when the variables of a communication model comply with the parameters, and do not exceed or limit the flow or packet variables. As mentioned before is considered an anomaly when there is a high variation degree in the network traffic of the communication model, meaning that It isn't a normal behavior of one of the packet variables. This anomaly can be related to network operations or failures in network measurement or monitoring systems or some type of flash crown that can affect one or more pieces of equipment or can be related to some type of attack in our infrastructure.

2.3.1 Anomaly-based detection principles

Anomaly-based detection can be classified as self-learning or programmed, self-learning systems learn and build an underlying model according to network behavior in a period of time and can be classified into non-time series and time series. Non-time Series is characterized by the use of a stochastic model involved inside the detector to build a normal behavior of the system and can be divided into Rule modeling or Descriptive statistics, as shown in Table 1.

Non-time series	Description
Rule modelling	<ul style="list-style-type: none"> • System learn itself and study network traffic behavior. • System formulates rules that define its normal functionality. • In the network environment, system analyzes the network traffic and presents warnings or alerts that match with defined rules.
Descriptive statistics	<ul style="list-style-type: none"> • System compiles mono-modal statistics from a system and defines a profile. • System analyse and build a distance vector for traffic and profile. • In the network environment, if the distance between network traffic and the defined profile is more significant an alert is displayed.

Table 1: Non-time series classification

Another approach is programmed, here a specialist teaches the system to analyze and detect anomalies and behavior in network traffic, defines the normal parameters that can be allowed, and also the indication of a security violation. It can be divided into descriptive (simple, rule-based, and threshold) statistics or default deny.

2.4 Industrial Networks

2.4.1 Definition

Industrial networks can be explained or defined as the combination of Ethernet and IP networks and real-time networks or Fieldbus, it's also characterized by the use of computing systems and servers, routers and process systems, or physical processes (Knapp & Langill, 2015).

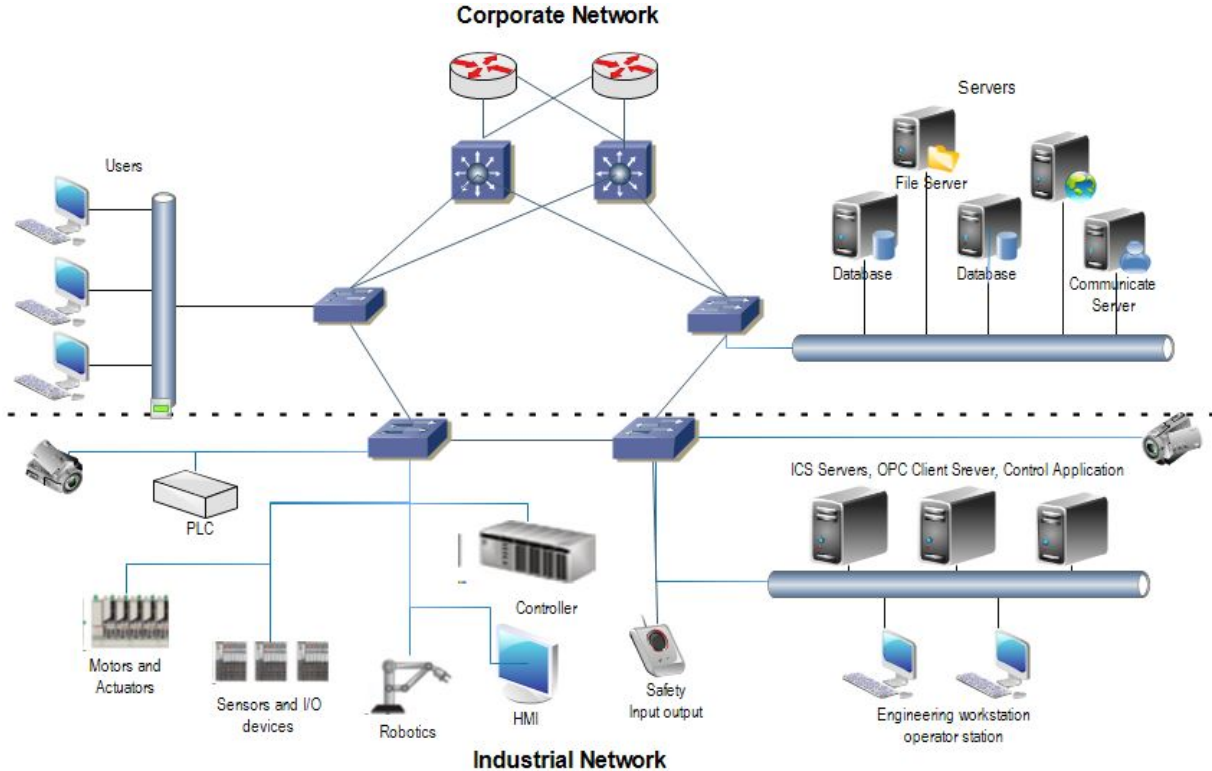


Figure 1: Industrial networks simple design

2.4.2 Industrial Networks Components

In the past, industrial network environments were isolated from business networks and the Internet but recently there have been a lot of transformations. Industrial control systems and supervisory control and data acquisition composed Distributed Control System (DCS), Programmable Logic Controller (PLC), Remote Terminal Unit (RTU), Intelligent Electronic Device (IED), and interface technology which is used to ensure the communication of component, these are the industrial networks behind the industrial companies IT network which provides services and delivers products in areas such as oil and petrochemical, transportation systems, factories, nuclear power plants, and energy data acquisition.

ICS and SCADA are becoming more modern with smart grid modernization, OT and IT integration, and connection to the cloud to provide big-data analytics. At the same time, this connection to another environment and the increase of the accessibility to a system comes with a lot of cyber vulnerabilities meaning that the companies have to be more careful in handling and analyzing network traffic.

2.4.3 Operational Technology and Information Technology

Information Technologies IT domain isn't fully adjusted to industrial process, the authors present three challenges, the fact that industrial process automation combines the two different areas cyber world and physical process which have to be seen as only one in a security point of view; problems of network topology, static applications, small capacity, predicted traffic behavior and simple protocol which difficult the process to collect network behavior; the fact of the real-time of industrial process automation traffic connected with embedded system, that could use more resources than available, which is not an issue in IT networks (Wressnegger, Kellner, & Rieck, 2018).

Operational technology prioritizes availability and safety, meaning that the most important thing is availability, followed by integrity and the last is confidentiality (AIC), in this type of environment all the processes are real-time and the response time to incidents is critical because it will affect the physical equipment or the production/manufacturing environment, as it is critical can not accept high delay and jitter. On the other side availability and reliability are very important, for example, a reboot may not be accepted and this type of operation requires redundancy, planning, schedule, and high availability of the equipment (Zhou et al., 2015).

Different than OT, IT prioritizes confidentiality followed by integrity and availability, usually, it can present different behaviors of network traffic, non-real-time or close to real-time, so the response to an incident needs to be consistent. IT presents high throughput, acceptable delay, and jitter so the emergence is less critical compared to OT operations, it's possible to implement a hard degree security policy and in terms of availability and reliability, it's more flexible.

2.4.4 Industrial Networks Security

According to (Fan, Fan, Wang, & Zhou, 2015) ICS is divided in functional security, physical security, and information security. The main idea of functional security is to keep security conditions even if failures occur, meaning that it's responsible for maintaining the safety functions of the devices. Issues related to handling and mitigating the risks and threats caused by radiation, electric shock, fire, mechanical hazards, and others are assigned to Physical security. International Electrotechnical Commission (IEC)-62443 considers information security of industrial control system "Measures which are taken to protect the system; the system state which is achieved by taking measures of establishing and maintaining system; avoiding unauthorized access to system resources and unauthorized or accidental change, damage or loss; Based on the ability of computer system, preventing unauthorized persons and systems from modifying the software and its data, and preventing them from accessing to system, while allowing authorized persons and systems to do these things; Avoiding illegal or harmful invasion of industrial control system or interfering with the correct and planning operation."

An industrial control system usually consists of four logical layers, first is the operational and management layer where the production plan, scheduler, management, and optimization are made, second centralized monitoring control layer which is responsible of data storage and monitoring and control of the production process, third field devices, responsible for production relay, measure storage and control and the fourth production process which normally includes regulator, sensors, switches/breakers can implement continuous or discrete industrial production (Fan et al., 2015).

2.4.5 Security Thread Model in Industrial Control System

Attackers outside of the company network use brute force, drive-by-compromised or social engineering techniques where they spread phishing to the employees like an email to trick company users into clicking on a link or opening a file, that is how they exploit any active and get access to IT network.

Once they are on the IT network they can access OT by looking to servers, credential, and other networks, they search for vulnerabilities to access the OT network, because all infrastructure is modern and use the TCP/IP model. In OT networks they find engineering workstations, maintenance systems, and software like Energy Maintain System (EMS), Globalization Management System(GMS), Advanced Metering Infrastructure(AMI), HMI, Historian, Engineering workstation, and PLC IED RTU that can have one or more vulnerabilities that can be subject to exploits, this is a critical phase because they can have access to the critical infrastructure SCADA and manage it as they want.

2.4.6 Industrial Network Protocols

In this discussion, we will focus on two frequently encountered industrial network protocols, Specifically, we will examine two fieldbus protocols, namely the Modicon Communication Bus (Modbus) and the Distributed Network Protocol (DNP3).

A. Modbus

Modbus is an application layer messaging protocol that operates at Layer 7 of the OSI model. It enables efficient communication between interconnected assets using a "request/reply" approach. Simple devices, like sensors or motors, utilize Modbus to communicate with complex computers, enabling data reading, analysis, and control functionalities. For a communication protocol to be implemented on a simple device, it is crucial that the message generation, transmission, and reception processes impose minimal processing overhead. Modbus's ability to meet this requirement makes it well-suited for use in PLCs and remote terminal units (RTUs) to transmit supervisory data to an ICS system. Modbus can be transported over Ethernet using TCP, known as Modbus TCP (Knapp & Langill, 2015).

Some Security Concerns presented are the lack of authentication in Modbus sessions is a significant vulnerability. It only requires a valid Modbus address, function code, and associated data to establish a session. The data must contain values from legitimate registers or coils in the slave device, otherwise,

the message will be rejected. However, an attacker can gather additional information about the target either through network traffic analysis or device configuration. Moreover, Modbus supports certain function codes that can be utilized without specific knowledge of the target, enabling straightforward man-in-the-middle (MitM) and replay attacks. This lack of verification for the message source allows unauthorized entities to manipulate the communication (Knapp & Langill, 2015).

Another security concern is the absence of encryption in Modbus. Commands and addresses are transmitted in clear text, making it easy for attackers to capture, spoof, or replay them. By intercepting network packets containing Modbus communications, adversaries can gain access to significant information related to the device's configuration and usage.

Furthermore, in Modbus/TCP, there is a lack of message checksum. This means that a command can be easily spoofed by constructing the Modbus/TCP Application Data Unit (ADU) with desired parameters. The checksum is generated at the transmission layer rather than the application layer, providing an opportunity for malicious actors to manipulate the commands.

B. DNP3

The DNP3 was developed with the objective of ensuring dependable communications in environments such as high levels of electromagnetic interference (EMI) and unreliable transmission media typically found in the electric utility industry. In 1998, DNP3 was expanded to operate over IP by encapsulating it within TCP or User Datagram Protocol (UDP) packets. Since then, it has gained widespread adoption not only in the electric utility sector but also in industries such as oil and gas, water, and wastewater (Knapp & Langill, 2015).

DNP3 (Distributed Network Protocol) is renowned for its exceptional reliability, efficiency, and suitability for real-time data transfers. It leverages standardized data formats and supports time-stamped and time-synchronized data, enhancing the efficiency and reliability of real-time transmissions. The protocol incorporates cyclical redundancy checks (CRCs) extensively, employing up to 17 CRCs in a single DNP3 frame, including in the header and each data block within the payload.

In terms of functionality, DNP3 enables the identification of remote device parameters and utilizes message buffers for event data classes. By comparing incoming messages to known point data, the master station can focus on retrieving new information resulting from point changes or events in the outstation.

This approach reduces unnecessary data retrieval and enhances efficiency (Knapp & Langill, 2015).

While DNP3 incorporates security measures, the complexity of the protocol introduces potential vulnerabilities. Several known vulnerabilities have been reported by ICS-CERT (Industrial Control Systems Cyber Emergency Response Team). Realistic hacks targeting DNP3 include Man-in-the-Middle (MitM) attacks to capture addresses for manipulation of system components. Examples of such manipulation include disabling unsolicited reporting to suppress alarms, spoofing unsolicited responses to the master station to falsify events and deceive operators into taking inappropriate actions, conducting Denial-of-Service (DoS) attacks through broadcast injection to create a storm behavior across the entire DNP3 system, and manipulating time synchronization data leading to synchronization loss and subsequent communication errors (Knapp & Langill, 2015).

2.5 Related Work

The literature presents several proposals that can be used for the detection of anomalies in industrial network environments.

(Zhou et al., 2015) proposed a multimodel-based anomaly intrusion detection system that can identify real attacks in industrial network traffic, they analyze ICP (control theory, and physical process) automation domains like communication, software, and control engineering and focus on two major areas the industrial processes and the intelligent control system. Then they present a multimodel-based anomaly intrusion detection for industrial process automation to detect abnormal behavior in PCSs traffic using special and temporal mechanisms. The authors use the hidden Markov model (HMM) to compile the results and enhance detection accuracy and differentiate attacks and faults. They present an optimized platform based on an optimized performance network engineering tool with a simplified Tennessee Eastman process (TEP) control system. (Wang et al., 2017) Proposed an intrusion detection and analysis based on Modbus TCP industrial control network combining misuse and anomaly detection. They implement the Stereo Depth SD-IDS algorithm which covers I) extraction rules to study the semantic association among the key areas in the Modbus TCP protocol and examines the characteristics of industrial network traffic. II) Deep packet inspection performing real-time packet analyses of time stamp, protocol type, packet length, and other variables for Modbus TCP traffic. To analyze intrusion behavior like DoS attack for example, authors count the number of packets arriving in the network within a period of time and compare it with the threshold value.

(Feng et al., 2017) Propose a multi-level anomaly detection framework based on machine learning and network patterns/signatures combining content level and time series level anomaly detection. They deployed ADS for ICS by monitoring temporal dependencies traffic between PLC, actuators, and sensors. In this study, Long Short-Term Memory (LSTM) network-based time-series and Bloom filter were also used as a probabilistic data structure because of limited resources in the ICS network and to validate if an element is a member of a set, If it's not, then the packet will be considered as a potential anomaly.

(Cai et al., 2021) Present content-agnostic-payload-based to detect abnormal ICP packets in ICSs, with automatic modeling of ICP packet formats and fields (the data units of packet formats). The authors implement Gaussian Mixture Model (GMM) to cluster the packets and label each one as a packet type, to determine the packet type in an unsupervised form. In other side, they used a hidden semi-Markov model (HSMM) to calculate the packet probabilities that reflect the structure of the packet formats and field aspects.

(J. Liu et al., 2018) Present a novel intrusion detection algorithm for ICS based on Convolutional Neural Networks (CNN) to feature extraction and anomaly identification and also process state transition model. According to the authors, this approach can detect anomalies in business processes, 0-day and unknown attacks.

(Wressnegger et al., 2018) Proposed content-based anomaly detection for proprietary binary protocols, they show that content can be used for the undocumented protocol and introduced the idea of representing specific to individual types of messages which characterize the format of message types and the data that they contain, this approach is called prototype models. On another side, they propose Noise-resilient Anomaly Detection to avoid irrelevant, noisy features and perform Large-scale evaluation using authentic SCADA data.

(Khan et al., 2021) Proposed a model for intrusion detection in IIoT-based ICS to detect cyber threats combining temporal and spatial features from data, using convolutional neural networks CNN and long short-term memory LSTM-based autoencoder framework to detect time-series attacks. The authors, implement a neural network to identify anomalies and classify mined features and used a two-step Sliding Window (SW) to have satisfactory comprehension of the latent representations of data features.

(Chang, Hsu, & Liao, 2019) Present framework with a semi-supervised technique to detect anomaly in ICS. Two methods were used in the study, first k-means to analyze the characteristics of the variables in an instant of time and second, convolution autoencoder to analyze its behavior, build a model of normal traffic examining the normal modification imprints of multiple attributes in continuous time.

(Farsi, Fanian, & Taghiyarrenani, 2019) Proposed the use state of process network to detect anomalies, combining analyses of the actual state of the process and state-based responsible to detect an abnormality on network traffic and take advantage of correlation attacks detention and failure diagnosis. This approach observes a window of different conditions to detect irregularities in the network and calculate metrics to differentiate normal from abnormal traffic.

(B. Liu, Chen, & Hu, 2022) Proposed anomaly detection with capabilities to detect attacks that compromised the integrity and availability of the systems. The authors considered three known attacks such as Stuxnet-like, DoS, and Dual channel false data injection (FDI) and used theoretical variation with testing evaluation to analyze, determine and quantify the mode variation. Their approach considers the setup model of attacks, building a detection framework, and designing and deriving a detection variable.

(Sheng et al., 2021) Proposed a model to detect intrusion in ICS by extracting and correlating the network communications pattern and the state of industrial physical instruments, In their study, they analyze and model network and physical process, and when any violation of the model is detected the system considers it as anomaly behavior. After executing several attacks, the result of attack patterns shows that different attacks present various combinations of statistical features that can characterize themselves.

(Das et al., 2020) Proposed use of a combinatorial mechanism to extract useful imprints from a binary dataset also called logical analysis of data. In their study, historical observations were applied to extract imprints and differentiate positive from negative patterns and use this information to build two rule-based classifiers, then these classifiers are used to detect anomalies in ICP and also used dataset Secure Water Treatment (SWaT) system to test the model.

(Myers, Suriadi, Radke, & Foo, 2018) Considers an industrial process as a series of events performed by physical devices or equipment used to provide a service. In their work, modifications in particular packet

variables were analyzed, observing deviation in the sequence of events. A model which represents the temporal communications patterns and the sequence of the event was defined and implemented of a series of mechanisms that involves improvement in business, process discovery, conformance checking, and process enhancement. This approach of process discovery and continually checking the ICS model and pre-processing logs, are fundamental to identify anomalies in ICS network. The use of control flow allow to detect modification of sequence of events in ICP, and can identify potential attacks. They also observe that cyclic data logs with low polling intervals are the most appropriate logging method in ICS and SCADA and evaluate their method to detect anomaly behavior in critical infrastructure of ICS and SCADA by different attacks and to train the model Siemens S7-1200 PLCs, and the National Instruments NI cRIO-9074 PLC datasets were used.

(Clotet, Moyano, & León, 2018) Proposed a model that uses all the information related to communications between the industrial process level of Critical Infrastructures devices, combining with multi-agent and negative selection algorithm to detect anomaly behavior in ICS, first data is prepared, selection of measures and raw data by the agent, second pre-processing which include scaling and dimensional reduction and finally the training and detection phase which include historical data for training and real-time data is analyzed to perform in detection.

(Jie et al., 2018) Proposed data mining process by observing historical data process variable content to detect anomaly behavior such as malicious activities, device failure, data injections, and reliability assessment of control system. Under the normal circumstances of ICS operation, the normal characteristics of the variable are identified and taken to form the normal model (normal association rules), then the authors implement sampling and comparison of variables with the real-time database during the data mining process, on these bases a reliability parameter is given to describe the anomaly state of the industrial control system.

(Khalili, Sami, Khozaei, & Pouresmaeeli, 2019) Presents a division of the main states of CPSs operation such as normal states, normal transitions between the normal states, and normal time intervals for transitions, and proposes a state-based IDS for stage-based CPSs. In their approach, they collect information from the sensors and actuators and build a model of normal transactions between the states and intervals between transactions to be able to detect three types of anomalies such as anomalous states, anomalous between the normal states transitions, and anomalous time-intervals between the normal transitions.

(Zhanwei & Zenghui, 2019) Proposed an anomaly detection based on ICS communication behavior, in their approach to detect anomaly they extract the behavior sequences and built a normal model which include control behavior and normal process behavior and then analyzed and compared the real-time extracted data sequences behavior with the calculated predicted behavior sequences based on normal model. According to the authors, it's possible to detect anomaly observing if the measurement data and control data are in the normal range.

(Kalech, 2019) Considers time as an important factor in anomaly detention which uses legal commands but abnormal time duration. Proposed an anomaly-based method to detect attacks that use authorized operations and modify the time interval between operations which can destroy SCADA components. They consider the use of the temporal pattern recognition technique. In their work, extracted data features were done with machine learning-based algorithms namely self-organized map artificial neural networks and Hidden Markov Models.

2.6 Conclusions

Anomaly detection can be applied using a statistical model, here it is possible to create a normal model based on network traffic statistics and if traffic is out of the boundaries it means that there is a variation degree in network traffic. These occur in the communications between devices and the variations could be an increase in inbound or outbound traffic, an increase of sessions, an increase in the total number of bytes, an increase of traffic to the same destination or other metric, so there isn't a standard for it, In this case, NSM based on behavioral analysis can play an extremely important role in the detection of irregular communications as well as in the identification of packets above the established threshold and can be useful in to detect unknown attacks like zero-day attacks.

In Operational Technology (OT) environment there is a problem of the high number of false positive alerts but it can be minimized not only because of the definition of a good model but also because it is an ICS environment, where most of the traffic is predictable since there are different isolated areas, which makes anomaly detection more accurate. Another way to detect is by analyzing the protocol, characteristics like unusual messages, deformed messages, sequential errors, and deviations from the normal functioning of

a protocol, this analysis is also critical in detecting remote services and zero-day exploits.

It is acceptable to correlate anomaly detection as a bunch of rules that can be used to detect a certain type of behavior that a heuristic-based system couldn't detect. As mentioned before these rules are based on statistical variation for example the total byte count from one area or zone increases > 40% or thresholds or Total Destination address > 20 (Knapp & Langill, 2015).

Taking into account the different topics analyzed in this chapter, we can also conclude that the most frequent attacks related to remote services are aimed at engineering workstations, some servers, and devices that interact with OT but also communicate with the external environment, we can also highlight other problems related to the design where many times there is no segmentation between OT and IT. It is very important to take into consideration that although anomalies detection is crucial to detect unknown attacks, a good security model must contain both aspects, that is, based on signatures detection and traffic behavior detection, and this second, the greater the diversity of the traffic and its variation from its normal operation, the greater the probability of occurrence of false positives.

Chapter 3

The Proposed Prototype

This chapter explains the ATT&CK Matrix for the ICS domain, it also presents NSM tools that are used for detection. Furthermore, we present the correlation matrix of IDS logs and attack techniques and based on it a framework for NSM intrusion detection system was built.

3.1 MITRE ATT&CK

The MITRE ATT&CK is widely regarded as an open framework utilized as a knowledge base to build specific threat models and methodologies within the public or private sectors in order to understand the strategies that adversaries might employ and facilitate the detection, prevention, and mitigation of malicious activities, effectively addressing global cybersecurity concerns (Alexander, Belisle, & Steele, 2020). MITRE ATT&CK boasts an extensive database of Tactics Techniques and Procedures (TTP), offering an open-source repository of adversarial behaviors relevant to various information security disciplines. These disciplines encompass Threat Intelligence, Adversary Emulation, Gap Analysis, as well as detection and analytics. The MITRE ATT&CK framework serves as a valuable resource for cybersecurity professionals, providing comprehensive insights into the tactics and techniques commonly employed by adversaries during cyberattacks (Strom et al., 2020).

Initial Access	Execution	Persistence	Privilege Escalation	Evasion	Discovery	Lateral Movement	Collection	Command and Control	Inhibit Response Function	Impair Process Control	Impact
12 techniques	9 techniques	6 techniques	2 techniques	6 techniques	5 techniques	7 techniques	11 techniques	3 techniques	14 techniques	5 techniques	12 techniques
Drive-by Compromise	Change Operating Mode	Hardcoded Credentials	Exploitation for Privilege Escalation	Change Operating Mode	Network Connection Enumeration	Default Credentials	Adversary-in-the-Middle	Commonly Used Port	Activate Firmware Update Mode	Brute Force I/O	Damage to Property
Exploit Public-Facing Application	Command-Line Interface	Modify Program	Hooking	Exploitation for Evasion	Network Sniffing	Exploitation of Remote Services	Automated Collection	Connection Proxy	Alarm Suppression	Modify Parameter	Denial of Control
Exploitation of Remote Services	Execution through API	Module Firmware		Indicator Removal on Host	Remote System Discovery	Hardcoded Credentials	Data from Information Repositories	Standard Application Layer Protocol	Block Command Message	Module Firmware	Denial of View
External Remote Services	Graphical User Interface	System Firmware		Masquerading	Remote System Information Discovery	Lateral Tool Transfer	Data from Local System		Block Reporting Message	Spoof Reporting Message	Loss of Availability
Internet Accessible Device	Hooking	Valid Accounts		Rootkit	Wireless Sniffing	Program Download	Detect Operating Mode		Block Serial COM	Unauthorized Command Message	Loss of Control
Remote Services	Modify Controller Tasking			Spoof Reporting Message		Remote Services	I/O Image		Change Credential		Loss of Productivity and Revenue
Replication Through Removable Media	Native API					Valid Accounts	Monitor Process State		Data Destruction		Loss of Protection
Rogue Master	Scripting						Point & Tag Identification		Denial of Service		Loss of Safety
Spearphishing Attachment	User Execution						Program Upload		Device Restart/Shutdown		Loss of Safety
Supply Chain Compromise							Screen Capture		Manipulate I/O Image		Loss of View
Transient Cyber Asset							Wireless Sniffing		Modify Alarm Settings		Manipulation of Control
Wireless Compromise									Rootkit		Manipulation of View
									Service Stop		Theft of Operational Information
									System Firmware		

Figure 2: The ATT&CK matrix for ICS

By harnessing MITRE ATT&CK’s repository, security practitioners can enhance their knowledge of prevailing cyber threats, enabling them to adopt proactive approaches in bolstering their organization’s resilience against potential cyber threats. Moreover, the shared open-source nature of the database fosters collaboration and collective intelligence within the cybersecurity community, fostering a united front in the ongoing battle against cyber adversaries (Strom et al., 2020).

MITRE ATT & CK presents different techniques that can be used to perform malicious activity in ICS. Since the focus of our study is to detect abnormal behavior in ICS, we will use the Mltre to understand and explore how attackers compromise the availability of a resource and also how they get unauthorized access and perform malicious activity in industrial networks, that is why our focus is on initial access and specifically Remote services and Denial of Services.

Initial access encompasses various techniques aimed at compromising OT and Information Technologies (IT) resources of both public and private organizations, as well as external services, websites, and other entities. These techniques serve as the attacker’s means to gain entry into ICS. Examples of such techniques include drive-by compromise, exploit public-facing application, remote services, and wireless compromise, among others. An important piece of information is that each high-level component of ATT&CK exhibits a connection or relationship with other components within the framework. The description fields provided in the previous section highlight these relationships, and they can be effectively visualized in figure 3 (TA0108, 2018; Strom et al., 2020).

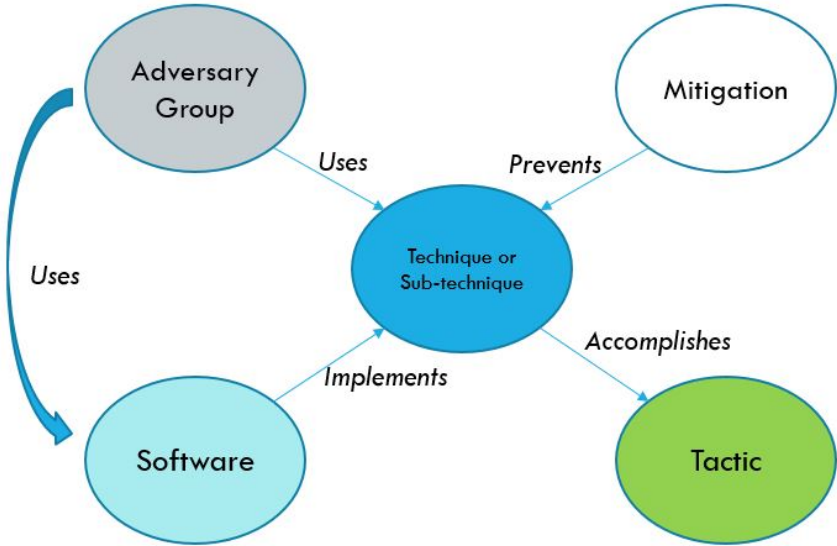


Figure 3: The ATT&CK correlation model

3.1.1 MITRE ATT & CK - ICS Technology Domain

ICS is an all-encompassing term that refers to various types of control systems commonly utilized in industries and critical infrastructures. A large oil refinery where numerous processes, such as temperature control, pressure regulation, and flow monitoring, are essential for smooth operations. SCADA systems enable efficient and safe automation of these physical processes (Alexander et al., 2020). For example, the SCADA system oversees and manages the entire refinery, collecting real-time data from sensors scattered throughout the facility. It continuously monitors variables like temperature in different sections, ensuring they stay within safe ranges. If any parameter deviates from the predefined thresholds, the SCADA system can automatically trigger corrective actions, such as shutting down a specific unit to prevent any potential hazards.

The diversity of these critical processes poses a challenge in defining the proper scope and abstraction level for this technology domain. To deal with this diversity, ATT&CK for ICS categorizes assets based on their classes, which represent distinct types of components found in industrial control systems. These asset classes may include PLC, HMI, and RTU. By focusing on asset classes, ATT&CK for ICS ensures that the specific security challenges associated with each type of component are thoroughly addressed (Alexander et al., 2020).

ATT&CK for ICS enumerates the following high-level systems adversary behaviors that could be affected: Process Control Systems (Process Control Operator Interface, Monitoring Real-Time, Historical Data and Alarming), Safety Instrumented System and Protection Systems and Engineering and Maintenance Systems (Alexander et al., 2020).

3.1.2 Process Control Systems (PCS)

These systems are responsible for managing and optimizing complex processes industrial processes. These systems receive inputs from various sensors and process instruments, and based on predefined control strategies, they provide appropriate outputs to ensure the process operates efficiently and safely. It includes Programmable Logic Controllers (PLCs) and Distributed Control Systems (DCS). Adversary behaviors targeting this system can disrupt or manipulate the physical processes, potentially leading to operational issues, equipment damage, or safety hazards (Knapp & Langill, 2015; Alexander et al., 2020).

3.1.3 Safety Instrumented Systems (SIS) and Protection Systems

These Systems are critical components of industrial processes designed to enhance safety by taking the process to a safe state in case of hazardous conditions. These systems are composed of sensors, logic solvers, and final control elements and are specialized and dedicated to safeguarding personnel, equipment, and the environment from potential harm or damage.

3.1.4 Engineering and Maintenance Systems

These systems are critical for configuring, diagnosing, and ensuring the smooth operation of the various systems discussed earlier in industrial settings. These systems play a key role in managing and supporting the overall infrastructure. They are used by engineers, technicians, and maintenance personnel to perform tasks ranging from setup and calibration to troubleshooting and repair (Knapp & Langill, 2015; Alexander et al., 2020).

1. Configuration: Engineering systems are employed to set up and configure the different components of industrial systems. For instance, engineers use these systems to define control strategies, set operational parameters, and establish communication protocols for sensors, actuators, and controllers.
2. Diagnosis and Monitoring: Maintenance systems provide tools for continuous monitoring of industrial processes and equipment. Engineers and maintenance personnel can access real-time data, trends, and performance metrics to identify anomalies or potential issues that require attention.
3. Predictive Maintenance: These systems enable predictive maintenance practices, where data analysis and machine learning techniques are used to predict equipment failures or performance degradation. This proactive approach helps schedule maintenance activities before critical failures occur, reducing downtime and operational disruptions.
4. Remote Access and Control: In many cases, engineering and maintenance systems allow remote access to the equipment. Engineers and technicians can connect to the devices over a network or through secure channels to perform diagnostics, configuration changes, and troubleshooting from a centralized location.

5. **Vendor-Supplied Software and Tools:** Many industrial equipment and control systems come with proprietary software and tools provided by the vendors. These software packages are specifically designed to interface with the respective equipment, offering comprehensive features for configuration, diagnostics, and maintenance.
6. **Firmware and Software Updates:** Engineering and maintenance systems are essential for deploying firmware and software updates to industrial devices. Keeping the devices up to date with the latest versions ensures optimal performance, security, and compatibility with other components.
7. **Historical Data and Documentation:** These systems often maintain historical data and documentation of equipment performance, maintenance activities, and operational changes. This information serves as a valuable resource for analyzing trends, conducting audits, and ensuring regulatory compliance.

Engineering and maintenance systems, being critical components in industrial environments, are not immune to potential attacks. Malicious actors may target these systems to disrupt operations, steal sensitive information, or cause harm to the infrastructure. Some attacks and their potential consequences for engineering and maintenance systems in ICS are:

1. **Malware and Ransomware Attacks:** Attackers may deploy malware or ransomware through phishing emails or compromised software in some cases driven by compromised updates to infiltrate engineering and maintenance systems. This can lead to unauthorized access, data theft, and system disruptions. In the case of ransomware, critical engineering and maintenance data may be encrypted, leading to operational downtime until a ransom is paid or the data is recovered.
2. **Denial-of-Service (DoS) Attacks:** Cybercriminals may launch DoS attacks against engineering and maintenance systems, overwhelming the system's resources and rendering it unavailable. As a result, engineers and maintenance personnel may lose access to critical tools and information needed for their tasks, leading to delays in maintenance activities and potential operational risks.
3. **Insider Threats:** Insider threats from disgruntled employees or individuals with privileged access to the systems can be especially concerning. These insiders may intentionally disrupt engineering and maintenance operations, compromise data, or steal sensitive information for malicious purposes.
4. **Supply Chain Attacks:** Engineering and maintenance systems often rely on vendor-supplied software and tools. Attackers may compromise these software packages during the supply chain

process, inserting backdoors or malicious code. When the compromised software is deployed in the industrial environment, it can lead to unauthorized access, data theft, or even sabotage.

5. **Zero-Day Exploits:** Attackers may discover and exploit previously unknown vulnerabilities (zero-day exploits) in the engineering and maintenance systems' software or firmware. This could provide them with unauthorized access to critical components and sensitive data.
6. **Man-in-the-Middle (MitM) Attacks:** In cases where remote access to engineering and maintenance systems is enabled, attackers may attempt MitM attacks to intercept communications and gain unauthorized access to the systems. This can lead to data manipulation, unauthorized control of industrial devices, and potential safety hazards.
7. **Data Breaches:** If attackers successfully breach engineering and maintenance systems, they may gain access to sensitive information, such as proprietary design data, operational logs, and maintenance schedules. Such data breaches can have severe consequences, including intellectual property theft, safety risks, and regulatory compliance issues.
8. **Social Engineering:** Social engineering techniques, such as phishing or pretexting, can be used to trick employees with access to engineering and maintenance systems into revealing sensitive information or providing unauthorized access to attackers.

The consequences of these attacks can be severe and far-reaching. Operational disruptions, safety hazards, financial losses, damage to reputation, and legal liabilities are among the potential outcomes. Protecting engineering and maintenance systems through robust cybersecurity measures, employee training, regular software updates, and secure network configurations is essential to mitigate these risks and maintain the integrity and safety of industrial environments.

3.2 Remote Services

Remote services are tools used to provide access to any enterprise network from outside, can be used to support remote access, data transmission, authentication, and other remote functions allowing end users or administrators to perform maintenance, repair applications, devices, or another type of resource remotely or virtually. These services such as Secure Shell (SSH), Remote Desktop Protocol (RDP), Virtual

Network Computing (VNC), and Server Message Block (SMB) clearly facilitate remote interaction with enterprise systems or resources (T1021, 2020).

Attackers always try to get unauthorized access to enterprise infrastructure and today with the merge of OT and IT exploration of Remote services vulnerabilities becomes notable not only to traverse across assets and network segments but also to get access to Engineering and Maintenance Systems, Engineering Workstations, Control Server, Human-Machine Interface and perform malicious activity to ICS.

3.2.1 Secure Shell SSH

Secure Shell protocol is a cryptographic network protocol that provides secure communication between client and server over an unsecured network, allowing authorized users to have access to the resources and have remote command-line login and execute remote commands. The server can be configured with a public-private key or a default password for the authentication process and exchange the key with the user public key (T1021/004, 2020).

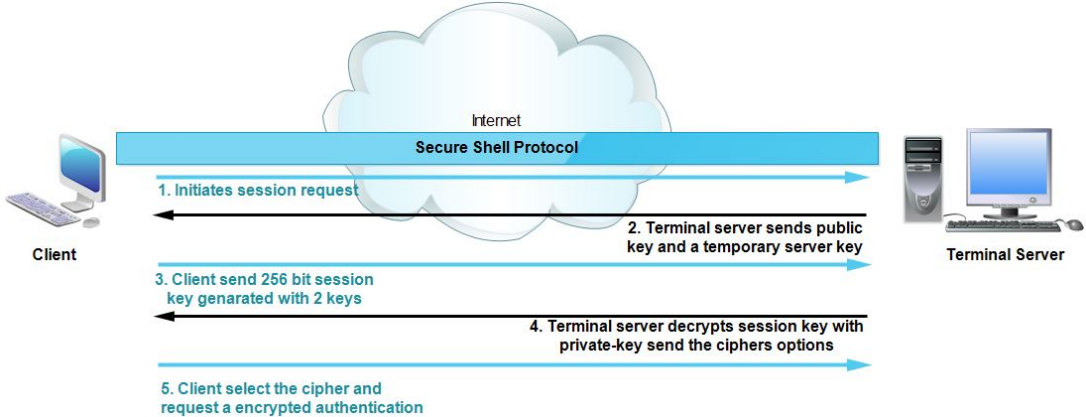


Figure 4: Secure shell protocol communication

In the real world, attackers will attempt to get access to a resource using brute force or another type of technique, zeek logs help us to detect using one of the ssh logs generated, in this case, we can use SSH Info field "auth attempts", more about this detention is detailed in next session.

3.2.2 Remote Desktop Protocol - RDP

RDP is a Microsoft protocol that allows the remote session to an enterprise Windows system over an asymmetric network connection, where the keyboard and/or mouse (input devices) from the client to the remote server. In the past few years has become a popular attack vector, because Attackers could use brute force methods and crack passwords and then get access to the system for malicious activities such as dropping malware, elevating user credentials, still or poisoning data, and so many malicious activities (T1021/001/, 2020). Figure 5 shows RDP stack communication.

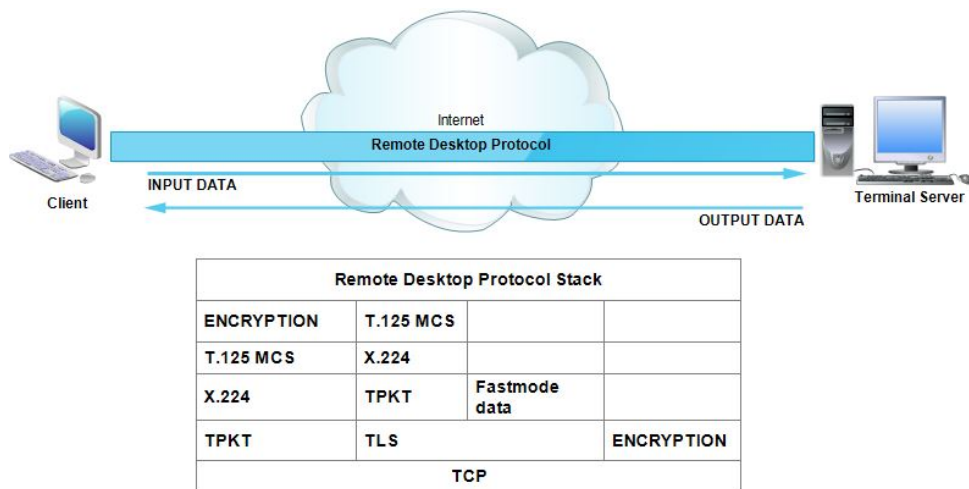


Figure 5: RDP stack and communication

3.3 Network Security Monitoring Tools

A network security tool for IDS architecture presents four main components, such as analyzers, sensors, response units, and storage. Figure 6 shows their relationship.

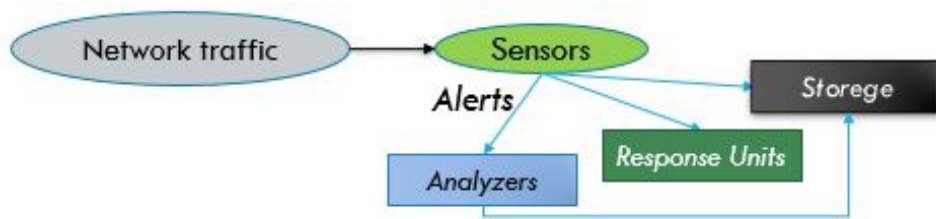


Figure 6: IDS architecture main components

3.3.1 Security Onion

Security Onion is an open-source Linux-based distribution that offers a comprehensive platform for network security monitoring, threat hunting, and log management. It integrates several open-source projects, including playbook, fleetDM, osquery, CyberChef, Elasticsearch, Logstash, Kibana, Suricata, Zeek, and Wazuh, which are tailored to facilitate network and system security monitoring. These tools are seamlessly incorporated into the distribution and can be easily configured and managed through a user-friendly web-based interface (Security Onion Solutions, 2023).

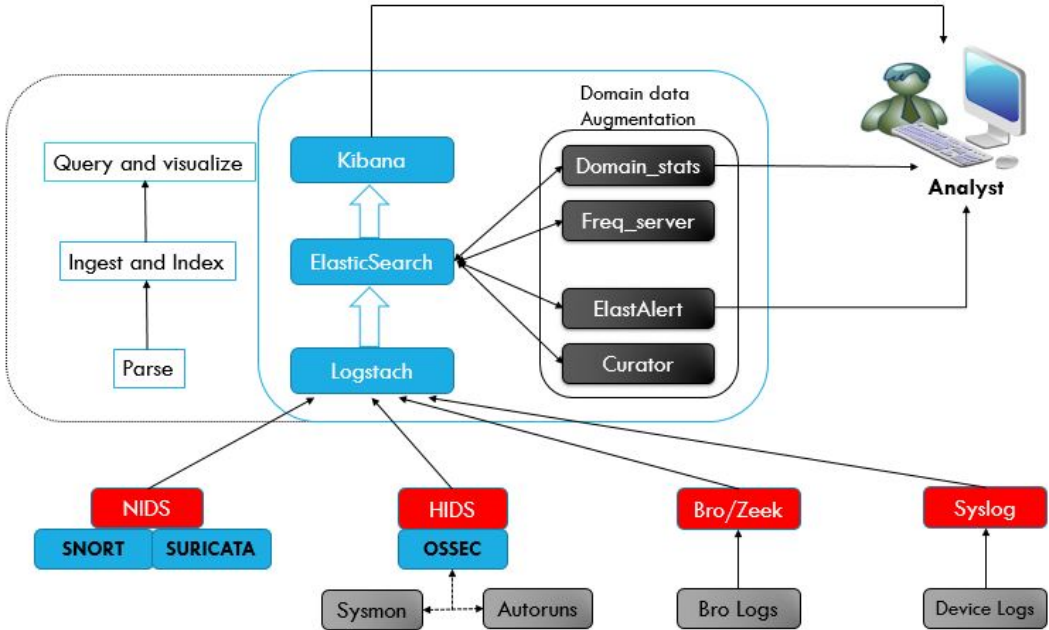


Figure 7: Security onion high-level architecture Diagram

Overall, Security Onion is an advanced and highly flexible platform that enables security professionals to efficiently identify and respond to security threats. By integrating a diverse range of tools, Security Onion aims to provide a comprehensive and cohesive solution for network security monitoring and log management.

3.3.2 Suricata

Suricata is an open-source network threat detection engine designed to analyze network traffic based on signatures. Network traffic refers to the data that flows across a computer network between different devices and systems. This data may include text, images, videos, or any other type of digital information that is transmitted across the network. The analysis of network traffic can help to detect potential security

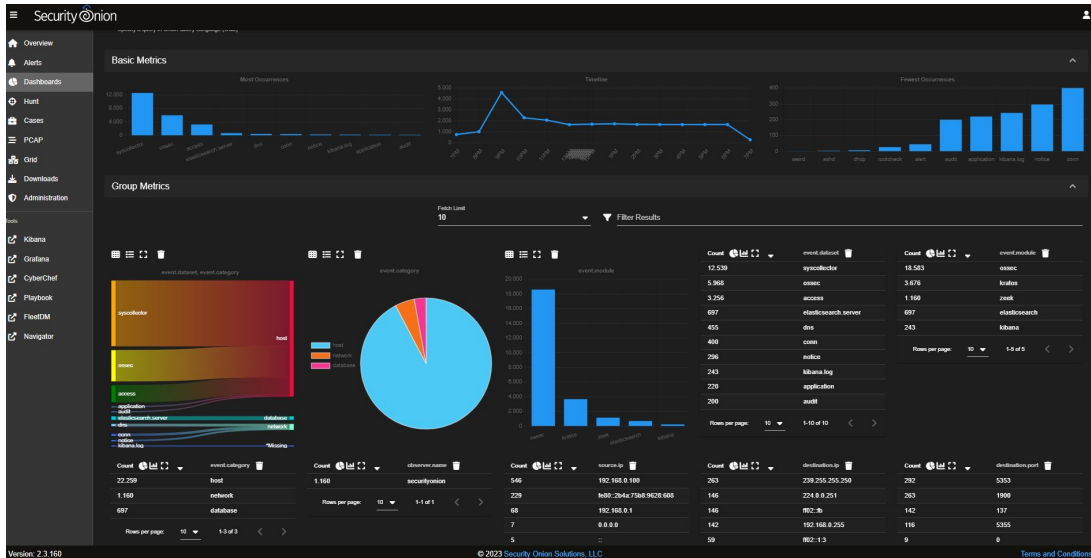


Figure 8: Security onion dashboard

threats, such as unauthorized access, data breaches, malware infections, and other types of attacks. Figure 9 shows Suricata alerts on the Security Onion environment (Nagadevara, 2017).

The Suricata alert interface displays a list of detected threats with the following columns: Timestamp, source.ip, source.port, destination.ip, destination.port, rule.name, rule.category, and event.severity_label. The alerts include:

Timestamp	source.ip	source.port	destination.ip	destination.port	rule.name	rule.category	event.severity_label
2020-07-10 21:32:18.857 +00:00	10.7.10.101	49723	91.200.103.114	80	ET INFO Dotted Quad Host DLL Request	Potentially Bad Traffic	medium
2020-07-10 21:32:18.862 +00:00	91.200.103.114	80	10.7.10.101	49723	ET MALWARE Likely Evil EXE download from dotted Quad by MSXMLHTTP M1	A Network Trojan was detected	high
2020-07-10 21:32:19.327 +00:00	91.200.103.114	80	10.7.10.101	49723	ET POLICY PE EXE or DLL Windows file download HTTP	Potential Corporate Privacy Violation	high
2020-07-10 21:32:19.327 +00:00	91.200.103.114	80	10.7.10.101	49723	ET MALWARE Likely Evil EXE download from dotted Quad by MSXMLHTTP M2	A Network Trojan was detected	high
2020-07-10 21:34:17.795 +00:00	190.136.178.52	449	10.7.10.101	49727	ET POLICY OpenSSL Demo CA - Internet Widgits Pty (O)	Not Suspicious Traffic	low
2020-07-10 21:34:19.750 +00:00	10.7.10.101	49728	172.217.6.179	443	ET POLICY IP Check Domain (myxofemalip.com in TLS SNI)	Potential Corporate Privacy Violation	high
2020-07-10 21:41:16.611 +00:00	190.136.178.52	449	10.7.10.101	49739	ET POLICY OpenSSL Demo CA - Internet Widgits Pty (O)	Not Suspicious Traffic	low

Figure 9: Suricata alert

The integration of Suricata with network security monitoring tools such as Security Onion provides security professionals with a powerful solution for detecting security threats. The extensive rules and signature language of Suricata, combined with its support for Lua scripting, allow for the efficient analysis of network traffic to identify and respond to potential security threats.

3.3.3 Zeek

Zeek is a passive open-source network traffic analyzer that is commonly used as a Network Security Monitor (NSM) to investigate suspicious or malicious activities. Zeek employs the conventional libpcap

library for packet capture, intended for employment in network monitoring and analysis, and generates a comprehensive set of logs that describe network activity, including Conn.log, HTTP.log, DNS.log, SSL.log, SMTP.log, FTP.log, DHCP.log, Software.log, and Weird.log. Conn.log provides a log of every connection seen on the wire, while HTTP.log offers a log of all HTTP sessions with their requested URIs, key headers, MIME(Multipurpose Internet Mail Extensions) types, and server responses. DNS.log provides a log of all DNS requests with replies, SSL.log provides a log of all SSL/TLS certificate chains seen, and SMTP.log provides a log of key content of SMTP sessions. Additionally, FTP.log provides a log of FTP control connections and file transfers, DHCP.log provides a log of all DHCP transactions, Software.log provides a log of all software installs, uninstalls, and updates, and Weird.log provides a log of all unusual or unexpected behavior that Zeek observes (Team, 2023).

Zeek writes all this information into well-structured tab-separated or JSON log files by default, which can be processed by external software for post-processing or analysis. Moreover, users can opt to have external databases or Security Information and Event Management (SIEM) products consume, store, process, and present the data for querying.

A. Zeek Architecture

At a profoundly elevated level, Zeek is architecturally organized into two principal components. Its core, known as the event engine, undertakes the task of processing the incoming packet stream, transforming it into a sequence of higher-level events. These events represent network activities in a policy-neutral manner, meaning that they describe what has been observed without providing any information about the underlying reasons or the significance of such observations (Bou-Harb, 2020).

B. Zeek Event Engine

The event engine layer performs intricate analysis on the network packets at a low level. It receives raw packets directly from the network layer, specifically through packet capture mechanisms, and subsequently arranges them based on their respective connections. Additionally, the layer adeptly reassembles data streams while skillfully decoding application layer protocols. Whenever it encounters any information that holds potential relevance to the policy layer, it efficiently generates an event, thus facilitating seamless integration with the broader network monitoring and analysis processes (Bou-Harb, 2020; Team, 2023).

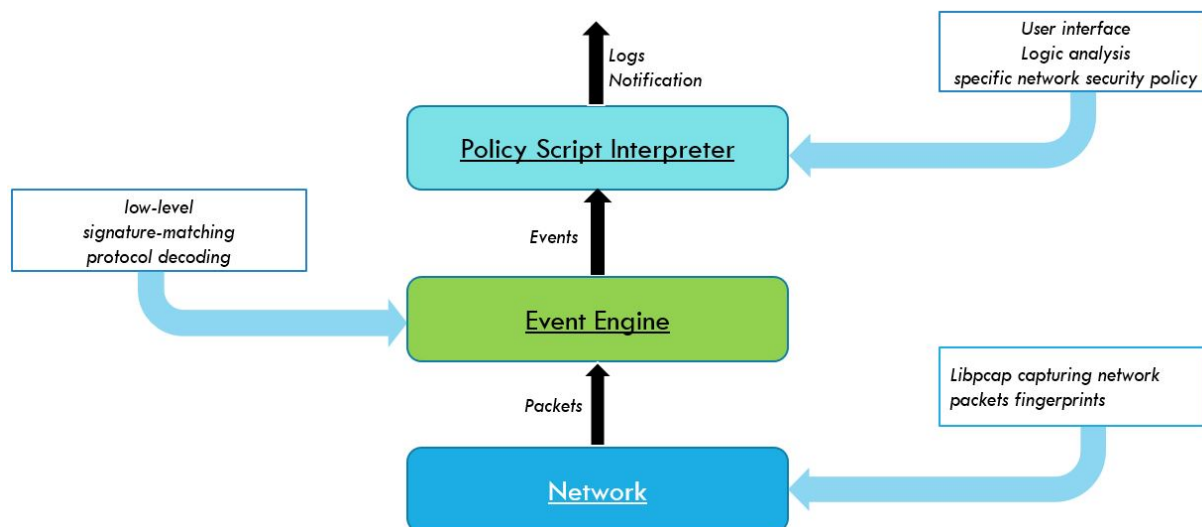


Figure 10: Zeek architecture

The event engine constitutes a comprehensive assembly of various analyzers, each carrying out well-defined and specific tasks. These tasks encompass a range of functionalities, such as protocol decoding, signature-matching, and backdoor identification, among others. Typically, each analyzer is accompanied by an accompanying default script, designed to implement a general policy that can be easily adjusted to suit the unique attributes of the local environment. Within its structure, the event engine can be meticulously divided into four major parts, each serving a crucial role in the network monitoring and analysis process (Bou-Harb, 2020; Team, 2023).

Packet analysis encompasses the examination of lower-level protocols, commencing from the link layer and progressing upwards. Session analysis, on the other hand, focuses on the scrutiny of application-layer protocols, such as Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and others. Further, file analysis delves into the dissection of file content transmitted across sessions. The event engine features a sophisticated plugin architecture that enables the seamless integration of additional functionalities, be it packet analysis, session analysis, or file analysis, from external sources beyond the core Zeek code base. (Bou-Harb, 2020).

C. State Management

The fundamental data structure employed by Zeek is the connection, which adheres to common flow identification mechanisms, specifically the 5-tuple approach. This 5-tuple structure encompasses the source IP address and port number, the destination IP address and port number, and the protocol currently in

use. In the context of connection-oriented protocols like TCP, defining a connection is relatively straightforward. However, for other protocols such as UDP and ICMP, Zeek adopts a flow-like abstraction to efficiently aggregate packets (Bou-Harb, 2020).

Notably, each packet within Zeek's analysis belongs to precisely one connection, ensuring a systematic categorization of network data for further inspection and analysis. This robust data structure forms the foundation of Zeek's network monitoring and analysis capabilities, enabling the comprehensive examination of network flows and the extraction of valuable insights from the observed traffic.

D. Transport Layer Analyzers

On the transport layer, Zeek diligently examines TCP and UDP packets. Specifically, in the case of TCP, Zeek's associated analyzer meticulously monitors the multitude of state changes that occur during the communication process. It proficiently maintains a record of acknowledgments, adeptly handles retransmissions, and addresses various other aspects crucial for ensuring reliable and efficient data transmission over TCP connections. Zeek's thorough analysis of TCP traffic enables it to capture and interpret the intricacies of this connection-oriented protocol, thereby facilitating comprehensive network monitoring and analysis (Bou-Harb, 2020).

E. Application Layer Analyzers

The examination of application layer data within a connection is contingent upon the specific service being utilized. Zeek employs dedicated analyzers for a diverse array of protocols, such as HTTP, SMTP, or DNS, each tailored to perform in-depth analysis of the corresponding data stream. For instance, Zeek's HTTP analyzer scrutinizes HTTP traffic, extracting and interpreting essential information like HTTP methods, URLs, headers, and responses. Similarly, the SMTP analyzer focuses on the intricacies of email communication, while the DNS analyzer delves into domain name system traffic, deciphering domain queries and responses (Bou-Harb, 2020).

By employing these specialized analyzers, Zeek ensures comprehensive and detailed analysis of application layer data, thus enabling network administrators and analysts to gain valuable insights into the nature of communication and data exchanges across various services. This multifaceted approach to application layer analysis enhances Zeek's capabilities in network monitoring and facilitates a deeper understanding of the network's behavior and potential security implications.

F. Script Policy Interpreter

Zeek's second principal component, the script interpreter, plays a pivotal role in establishing the semantic context surrounding the events. This critical component operates by executing a set of event handlers meticulously written in Zeek's custom scripting language. Through the adept utilization of these scripts, the system becomes capable of expressing a site's individualized security policy, dictating the appropriate actions to be taken in response to distinct types of activities identified by the monitoring process (Bou-Harb, 2020).

The script interpreter serves as the conduit for imparting intelligence and decision-making capabilities to Zeek, as it transforms raw data into actionable insights. By leveraging the expressive power of its custom scripting language, Zeek ensures that the monitoring and analysis activities remain closely aligned with the specific security requirements and desired responses unique to each site. As a result, Zeek's adaptability and flexibility are substantially enhanced, enabling it to cater to diverse network environments and security postures (Bou-Harb, 2020).

In a broader context, scripts within Zeek possess the capability to derive a wide range of desired properties and statistics from the input traffic. It is worth noting that all of Zeek's default output is generated from scripts that are thoughtfully included in the distribution. Zeek's custom language boasts an array of extensive domain-specific types and support functionalities. A crucial aspect of Zeek's language is its ability to allow scripts to maintain state over time, thereby empowering them to effectively track and correlate the evolution of observed phenomena across connection and host boundaries. This feature proves invaluable in gaining deeper insights into network activity and establishing meaningful relationships between various data points.

Moreover, Zeek scripts have the capacity to generate real-time alerts, enhancing its capabilities for detecting and responding to potential security threats promptly. Additionally, scripts can even execute arbitrary external programs as per demand. This particular functionality provides the opportunity to implement active responses to potential attacks, thereby bolstering the security posture of the system.

G. Zeek Control

Zeek Control is an interactive shell designed to facilitate the effortless operation and administration of Zeek installations. This tool proves invaluable for managing Zeek on a single system or efficiently coordinating multiple Zeek installations across a traffic-monitoring cluster.

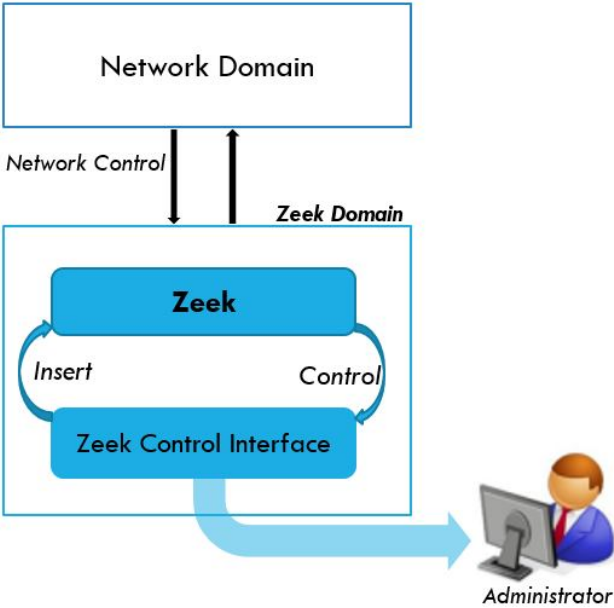


Figure 11: Zeek Control

By employing Zeek Control, network administrators can streamline the process of deploying, starting, stopping, and monitoring Zeek instances, effectively simplifying the management of complex network monitoring setups. The interactive nature of the shell enables administrators to perform various tasks through a user-friendly interface, promoting efficient and seamless management of Zeek installations and enhancing the overall effectiveness of traffic-monitoring operations.

3.3.4 Elasticsearch, Logstash, and Kibana

Security Onion provides a combination of Zeek, Elasticsearch, Logstash, and Kibana, Zeek produces a comprehensive set of logs that provide detailed information on network activity, which can be further analyzed and processed through Elasticsearch, Logstash, and Kibana (see Figure 7). Elasticsearch is a search and analytics engine that distributes, stores, and indexes large volumes of data. It offers advanced search capabilities and supports complex queries. Logstash is a data processing pipeline that accepts data from

different sources and processes it accordingly. It processes and transforms the Zeek logs and includes additional data before forwarding it to Elasticsearch for indexing.

Kibana is a powerful data visualization tool that allows users to interact and explore data stored in Elasticsearch. This combination offers a robust solution for network security monitoring and threat detection and It empowers security administrators to process and analyze vast amounts of network data and detect potential security threats effectively, thereby enabling a timely response.

The stored logs can be viewed and analyzed through various interfaces, including Dashboards, Hunt using hunt queries, and Kibana. It is possible to configure Zeek to output logs in JSON format and parse those JSON logs from the command line using jq.

```
head -8 conn.log | jq -j '.duration, ", ", .proto, ", ", \
    .["id.orig_h"], ":", .["id.orig_p"], ", ", \
    .["id.resp_h"], ":", .["id.resp_p"], "\n"'
```

```
0.220463, udp, 192.168.0.130:64277, 192.168.0.2:53
0.000502, udp, 192.168.0.130:55106, 192.168.0.2:53
0.001759, udp, 192.168.0.130:53881, 192.168.0.2:53
0.044681, tcp, 192.168.0.130:49965, 236.5.220.65:443
0.000434, udp, 192.168.0.130:53785, 192.168.0.2:53
0.014844, udp, 192.168.0.130:60696, 192.168.0.2:53
0.001553, udp, 192.168.0.130:59251, 192.168.0.2:53
0.000861, udp, 192.168.0.130:58172, 192.168.0.2:53
```

Figure 12: Example of jq

3.4 Correlation Matrix of Zeek Logs and Attacks Techniques

Objectives	Techniques	Description	Indicators	Zeek Logs	Fields
the attacker sends repeated SYN, RST, or FIN packets to every port to is to disrupt the normal functioning and overwhelm a target system, typically a network server, and flooding it with a high volume of packets.	Denial-of-service SYN flood (TCP flags - ACK floods, RST floods, FIN floods)	Send SYN packets to an engineering workstation or a server, the target responds with SYN-ACK packets and awaits acknowledgment from the client, which can lead to the victim maintaining its ports in an open state. This can have the adverse effect of rendering certain resources unavailable when genuine devices attempt to access them.	<ul style="list-style-type: none"> - High volume of SYN packets. - A large number of half-open connections (syn received in victim) - Network bandwidth saturation experience * - high volume of RST packets or abnormal connection termination patterns * 	conn.log tcp.log	conn_state, history
The attacker Executes a DDoS attack by sending HTTP GET or HTTP POST requests causing the application to be unavailable and resource exhaustion.	HTTP methods Get or Post	The attacker overwhelms the target application and web servers by inundating them with a massive volume of HTTP GET or POST requests. The sheer number of these requests puts a strain on the server's connection threads, leading to the allocation of system resources to handle the influx of incoming requests.	<ul style="list-style-type: none"> - Unusually high number of GET or POST requests. - Unusual traffic patterns (spikes in connection) - A lot of requests originating from a single IP - Unusual patterns in user-agent strings (requests coming from bots) - Increase of server resources utilization* - Degraded server performance* 	conn.log Notice.log http.log	id.orig_h, id.resp_h, proto, orig_bytes, resp_bytes, orig_pkts, resp_pkts, conn_state, history Note and msg method
The attacker aims to gain unauthorized access through remote access services.	Brute Force Attacks	Attackers can use a script or a program that to try different usernames and passwords in order to log in and have remote access to the engineering workstation.	<ul style="list-style-type: none"> - High number of attempts to log in to a system - large volumes of RDP connections - Unusual certificate - Unusual version of cipher commands, or control messages - Unusual RDP port requests, - Long time remote session duration - Unusual source IP address - irregular directions, and flags 	Conn.log Rdp.log	Resp_p = tcp/3389, tcp/22 orig_bytes and resp_bytes higher than 1000 or duration take too much time could be any anomaly id.resp_p different name or cookie in port 3389. orig_h, resp_h, resp_p, c\$cookie;
	Credential Theft Password-Cracking Man-in-the-Middle Attacks	Attackers can install a key logger or social engineering or phishing to get the user credential Attackers capture network traffic between source and destination, use sophisticated tools to decrypt, and have access to user credentials		Ssl.log notice.log Ssn.log notice.log x509.log	id.orig_h, id.orig_p, id.resp_h id.resp_p, version cipher Note, msg auth_success, auth_failure, orig_h, resp_h, resp_p, auth_attempts, version. Src, dst, msg () Subject, Unauthorized certificates

Important notes:
In the case of SMB, we believe that there are other tools that can be used to detect malicious activity in a more efficient way than Zeek providing better results.
** Indicators for which Zeek cannot detect or it is necessary to carry out a series of calculations, it is best to integrate it with another tool.*

Table 2: Correlation Matrix of Zeek Logs and Attacks Techniques

3.5 Detection Framework

Anomaly detection proves advantageous and covers the limitations of heuristics detection methods. It presents a "rule-less" approach to identifying threatening behavior. Anomaly detection of remote services involves the use of some variables like packet sizes, times, directions, flags, and thresholds that can be used to compare the number of attempts the same ip tried to get access to a resource or the limited bandwidth allowed.

Our detection approach uses time statistical analysis to detect variations in packet arrival interval times. Figure 13 shows normal packet arrival T1, T2, T3, T4 and T8. Abnormal packet arrival are T5, T6 and T7, delta T is the difference between arrival time and DT is defined threshold time according.

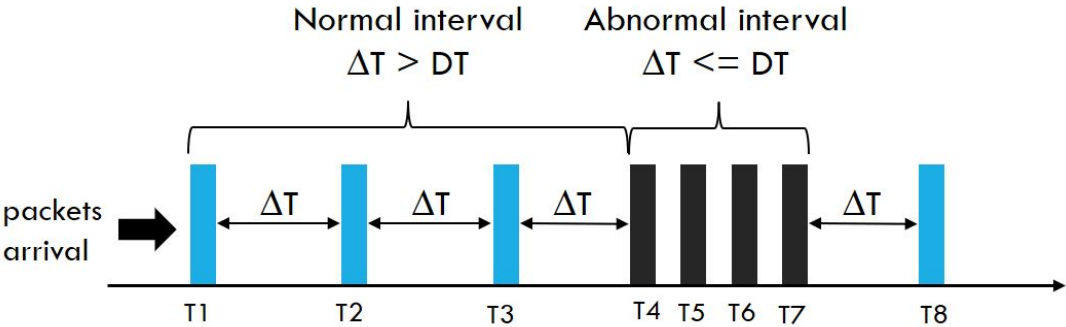


Figure 13: Statistical packet time arrival anomaly detection

In this statistical time packet arrivals model, we record the below information numerical values for each connection to calculate the variance as each packet arrives and compare with the defined time threshold.

- The arrival time of the previous packet
- The actual arrival time of the packet
- The value of the difference in arrival time

To fulfill the objectives of this work, we use a clean sample of ICS normal operation as baselines of normal behavior. In this process, we analyze the rate of SYN packets per second (around 2.9 and 3), the number of ssh and RDP attempts (defined as 3 attempts), and the average number of get and post requests. Our approach is divided into different phases such as:

- Define the baseline thresholds of packet size and the number of attempts to access a resource.
- Identify and use the zeek fields and events that can help to detect that type of malicious activity.
- Compare the communications fingerprints with the defined thresholds.
- Detect malicious activity and identify the source, destination, port, and activity information.
- Store the results in a new log file according to the type of malicious activity detected.

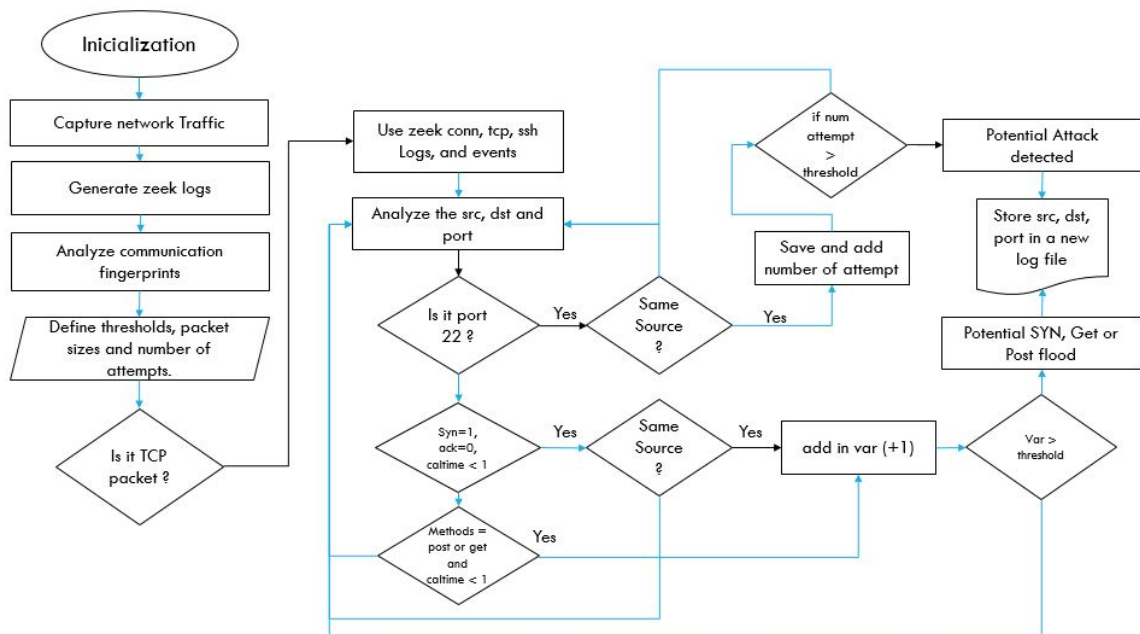


Figure 14: Architecture of detection framework

3.6 Prototype Implementation

3.6.1 DOS Syn Flood Attack Detection

The communication between server and client over TCP uses a three-way handshake mechanism to establish a connection, in this mechanism the client sends a syn packet to the server where Set SYN=1, which is a control bit flag that signifies a request to establish connection and data synchronization, set ACK=0, this is an ACK control bit flag that signifies that there is no acknowledgment of request and define an initial sequence number ISN which is random unique identifiers in the form of a 32-bit number. The server reply with syn_ack packet, set ACK=1, meaning acknowledgment of the request, defines the server's ISN by adding one to the random ISN sent by the client and allocates resources for this connection to be established, then stays in a wait state designated transmission control block until receiving the final Ack that the client sends, where set SYN=0 that there is no more request, set ACK=1, acknowledgment of request and add one to the server's ISN to close the connection.

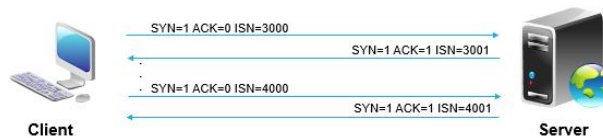


Figure 15: Syn flood attack example

Denial of services Syn flood explores this last step of tcp three-way handshake, keeping the server in TCB state waiting for any ACK coming from the client to finish the connection as successful, this step can be seen in figure 15. Detection with Zeek can be done by observing the amount of syn sent by the client in a small time slot as shown in Figure 16.

```
#Check if it is from the same source and the bit SYN is true according with type of log is "S"
#If (is_orig && flags == "S" && ack ==0)
#If ( flags == "S" && ack ==0 )
if (is_orig && flags == "S" && ack ==0 && c_interv <= 1.0 )
{
    syn_flood_counter[ip] +=1;

    # Check if the SYN counter exceeds the threshold
    if (syn_flood_counter[c$id$orig_h] > syn_flood_threshold)
    {
        #print fmt("Possible DOS SYN flood attack from %s was detected", c$id$orig_h);
        Log::write(LOG, Info($tms=network_time(),
            $s_host=c$id$orig_h,
            $dst_host=c$id$resp_h,
            $used_port=c$id$resp_p,
            $used_msg="Possible DOS SYN flood attack was detected"
        ));

        syn_flood_counter[c$id$orig_h] = 0;
    }
}
```

Figure 16: Syn flood detection example

3.6.2 DOS HTTP Flood Attack Detection

Denial of services attack can be used to compromise DNS server (not in scope of this study) or web server. HTTP flood attack is a type of Layer 7 attack that sends a large number of HTTP requests to a web server involving HTTP GET or HTTP POST methods. In this attack, one or different computers send multiple requests for images, and files to a server, and when the server is inundated it will not receive requests from a legitimate user. HTTP post deal also with bandwidth capacity because could involve sending amount of requests with the command to be pushed on the server side (for example a database) causing a large consumption of resources and leading to saturation of the Server.

```
if (c$method == "POST" && c_interv <= 1.0 )
{
    psum +=1;
    if (psum > threshold_post_req)
    {
        # print fmt("Abnormally high number of POST requests detected from %s within %s", c$id$orig_h, threshold_interval);
        Log::write(LOG, Info($tms=network_time(),
            $src_host=c$id$orig_h,
            $dst_host=c$id$resp_h,
            $used_port=c$id$resp_p,
            $used_method=" high number of POST requests detected"
        ));
        psum = 0;
    }
}

if (c$method == "GET" && c_interv <= 1.0 )
{
    gsum +=1;
    if (gsum > threshold_get_req)
    {
        # print fmt("Abnormally high number of GET requests detected from %s within %s", c$id$orig_h, threshold_interval);
        Log::write(LOG, Info($tms=network_time(),
            $src_host=c$id$orig_h,
            $dst_host=c$id$resp_h,
            $used_port=c$id$resp_p,
            $used_method=" high number of GET requests detected"
        ));
        gsum = 0;
    }
}
}
```

Figure 17: HTTP Get and Post flood attack detection example

Detection can be done by analyzing traffic patterns and packet inspection associated to protocol, logging, messaging, and control network traffic. Our approach is based on the detection of anomalous network flows or uncommon data flows in time intervals using Zeek connection, HTTP and TCP log as shown example code in Figure 17.

3.6.3 Brute Force SSH Attack Detection

Brute force technique acquire to have access to accounts in situations where passwords are unknown or when password hashes are obtained is a common practice. In the absence of password knowledge for a specific account or a group of accounts, an Attacker may employ a systematic approach to repeatedly or iteratively guess the password. The process of brute forcing passwords can occur either through engagement with a service that verifies the authenticity of provided credentials or offline, by testing them against previously obtained credential data, such as password hashes (T1110, 2013).

```
event zeek_init()
{
  Log::create_stream(LOG, [$columns=Info, $path="icsha_bforce_ssh"]);
}

# Access the SSH record as it is sent on to the logging
event SSH::log_ssh(c: SSH::Info)
{
  #Compare number of attempt with the defined threshold
  if (c$auth_attempts >= attempts_threshold) {
    print fmt("Potential brute force attack detected from IP: %s, to DST: %s, Port: %s, Protocol Version: %s.",
              Log::write(LOG, Info($tms=network_time(),
                                  $src_host=c$id$orig_h,
                                  $dst_host=c$id$resp_h,
                                  $used_port=c$id$resp_p)));
  }
}
```

Figure 18: SSH brute force detection example

To implement this detection method effectively, the script can be designed to leverage data from log and event, specifically the ssh.log and event SSH. Within this log and event, certain variables, such as auth_attempts and auth_success, are of particular interest and may serve as focal points for analysis. However, a well-crafted script should offer sufficient customization options to strike the right balance between sensitivity to potential threats and minimizing false positives.

Timestamps (ts) and uid values within the log entries provide valuable indices that can be used to uncover potential brute-force attempts. By carefully examining these timestamps and uid values, administrators can discern patterns indicative of suspicious login activities.

One possible approach involves employing the Zeek method to extract specific fields from the ssh.log and then obtain the desired information. This selective extraction allows administrators to focus on the essential data points relevant to this detection method. By strategically selecting and analyzing these fields, network administrators can gain deeper insights into potential security breaches, ultimately enhancing the network's monitoring and defense capabilities.

Another way of Brute force ssh attack detection is by logging and monitoring a number of authentication failures across the same or various accounts and also detecting the execution of commands used in brute force techniques. Our approach is based in the analyses of connection or/and ssh logs and SSH events generated by Zeek, in this case, we define a threshold value, then compare the number of attempts provided ssh info and if authentication attempts are higher or equal to the threshold value then any potential brute force attack detected and the results are stored in a bforce.log file as shown example in figure 18.

Chapter 4

Experimental Tests, Results and Analysis

This chapter focuses on the experiment and evaluation of the detection model. The experiment helps us determine the accuracy and performance of the IDS in abnormal detection. The experiment contains malicious which are generated by software components such as hping3 and Slowhttptest and to validate the model we use a dataset. Furthermore, the performance metric used in this study was the number of alerts generated by the IDS.

4.1 Experimental Environment

The lab supporting this research was built in a physical server equipped with a type-1 hypervisor, specifically VMware¹ ESXi. The server operated under a free license and hosted the following operating systems: Security Onion², Ubuntu Desktop³ 22.04, Kali Linux⁴, and windows⁵. Additionally, a laptop model HP, running Windows 10 as host and Ubuntu 20.04, employed a type-2 hypervisor known as VirtualBox⁶. The network device employed to distribute network traffic was a TP-Link wireless router that featured five ports, table 3 shows the used equipment.

Type	Model	Processor	RAM	HDD	Operating system
Physical	HP ProLiant ML 350p G8	Intel Xeon E5-2690 2.90GHz/3.80 GHz	64 GB	1.2 TB x 2	ESXI
Virtual	-	8 vCPUs	16 GB	300 GB	Security onion
Virtual	-	4 vCPUs	4 GB	60 GB	Kali Linux
Virtual	-	4 vCPUs	4GB	60 GB	Windows (Eng workstation)
Virtual	-	4 vCPUs	4GB	60 GB	Windows (Server)
Physical	HP Notebook - 15-bs080wm	Intel(R) Core(TM) i7-7500U 2.70GHz - 2.90 GHz	16 GB	916 GB	Windows 10 Pro 64 bits
Virtual	-	1 vCPUs	4GB	50 GB	Ubuntu 20.04
Physical	TP-Link wireless router	400 MHz	32 MB	-	3.20.1 Build 211111 Rel.32795n (4555)

Table 3: Technical specifications of equipments

¹ <https://www.vmware.com> [Accessed: 15-Mar -2023].

² <https://securityonionsolutions.com> [Accessed: 1-Nov-2022].

³ <https://ubuntu.com> [Accessed: 20-Nov-2022].

⁴ <https://www.kali.org> [Accessed: 20-Nov-2022].

⁵ <https://www.microsoft.com/pt-pt/windows/?r=1> [Accessed: 1-Apr-2023].

⁶ <https://www.virtualbox.org>[Accessed: 25-Oct-2022].

4.2 Topology

The designed environment serves the purpose of simulating real network traffic and conducting comprehensive attack tests, aiming to validate the functionality of Zeek in capturing various events and effectively detecting potential attacks while generating corresponding log files. It is worth mentioning that the network card of the Security Onion system has been configured in promiscuous mode. The network itself is established on the 192.168.0.0 IP address range with a subnet mask of 255.255.255.0. The IP addresses for the Administrator and a user are assigned via wireless connectivity, while other devices are configured to receive IP addresses via wired connections.

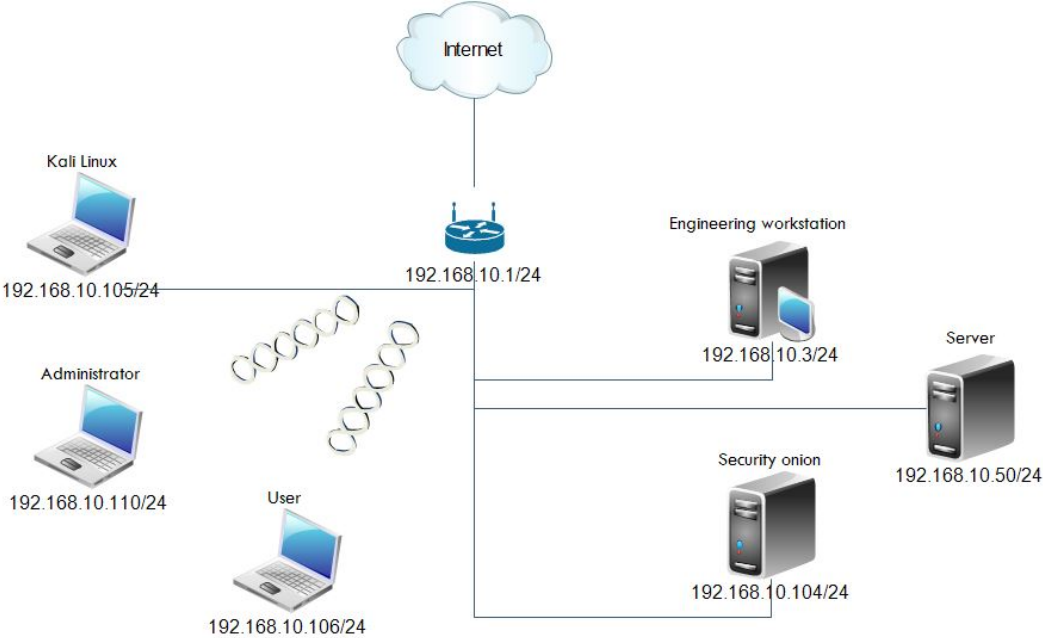


Figure 19: Topology used for tests in a controlled environment

4.3 Tools

In the experimental environment, we use a variety of tools to conduct our experiments, such as Hping3, SlowHTTPTest and Ncrack.

4.3.1 Hping3

Hping3 is an open-source network traffic generator capable of generating ICMP/UDP/TCP packets. Its functionality is based on utilizing the ping program with ICMP connections. This tool is capable to handle fragmentation and accommodate arbitrary packet body and size and can be used to transfer files under supported protocols. Various types of traffic can be directed towards a host, allowing the user to customize parameters such as different speeds and numbers of packets. However, it is essential to recognize that this tool can potentially be misused for malicious purposes, such as conducting Distributed Denial of Service (DDoS) attacks.

```
root@kali:~# hping3 -h
usage: hping3 host [options]
[-h --help show this help] [-v --version show version][-c --count packet count] [-i --interval wait (uX for X microseconds, for example -i u1000)
--fast alias for -i u10000 (10 packets for second)
--faster alias for -i u1000 (100 packets for second)
--flood sent packets as fast as possible. Don't show replies.]
[-n --numeric numeric output] [-q --quiet quiet] [-I --interface interface name (otherwise default routing interface)]
[-V --verbose verbose mode] [-D --debug debugging info] [-z --bind bind ctrl+z to ttl (default to dst port)]
[-Z --unbind unbind ctrl+z]
--beep beep for every matching packet received]

Mode
default mode TCP
[-0 --rawip RAW IP mode] [-1 --icmp ICMP mode] [-2 --udp UDP mode]
[-8 --scan SCAN mode]
Example: hping --scan 1-30,70-90 -S www.target.host] [-9 --listen listen mode]

IP
[-a --spoof spoof source address] [--rand-dest random destination address mode. see the man.] [--rand-source random source address mode. see the man.]
[-t --ttl ttl (default 64)] [-N --id id (default random)] [-W --winid use win* id byte ordering]
[-r --rel relative id field (to estimate host traffic)] [-f --frag split packets in more frag. (may pass weak acl)]
[-x --morefrag set more fragments flag] [-y --dontfrag set don't fragment flag] [-g --fragoff set the fragment offset]
[-m --mtu set virtual mtu, implies --frag if packet size > mtu] [-o --tos type of service (default 0x00), try --tos help]
[-G --rroute includes RECORD_ROUTE option and display the route buffer] [--lsrr loose source routing and record route]
[--ssrr strict source routing and record route] [-H --ipproto set the IP protocol field, only in RAW IP mode]

ICMP
-C --icmptype icmp type (default echo request)
-K --icmpcode icmp code (default 0)
--force-icmp send all icmp types (default send only supported types)
--icmp-gw set gateway address for ICMP redirect (default 0.0.0.0)
--icmp-ts Alias for --icmp --icmptype 13 (ICMP timestamp)
--icmp-addr Alias for --icmp --icmptype 17 (ICMP address subnet mask)
--icmp-help display help for others icmp options

UDP/TCP
[-s --baseport base source port (default random)] [-p --destport [+][+<port> destination port (default 0) ctrl+z inc/dec]
[-k --keep keep still source port] [-w --win winsize (default 64)] [-O --tcpoff set fake tcp data offset (instead of tcphdrlen / 4)]
[-Q --seqnum shows only tcp sequence number] [-b --badcksum (try to) send packets with a bad IP checksum
many systems will fix the IP checksum sending the packet
so you'll get bad UDP/TCP checksum instead.] [-M --setseq set TCP sequence number] [-L --setack set TCP ack] [-F --fin set FIN flag]
[-S --syn set SYN flag] [-R --rst set RST flag] [-P --push set PUSH flag] [-A --ack set ACK flag] [-U --urg set URG flag]
[-X --xmas set X unused flag (0x00)] [-Y --ymas set Y unused flag (0x00)] [--tcpexitcode use last tcp->th_flags as exit code]
[--tcp-mss enable the TCP MSS option with the given value] [--tcp-timestamp enable the TCP timestamp option to guess the hz/uptime]
```

Figure 20: Example command of hping3

4.3.2 SlowHTTPTest

SlowHTTPTest is an open-source tool designed for testing HTTP server responses to Slowloris-type attacks. The Slowloris attack is a type of Denial of Service (DoS) attack that aims to exhaust a server's resources, particularly its capacity to handle concurrent connections, by sending slow and incomplete HTTP requests. This tool implements most common low-bandwidth application layer Denial of Service attacks, such as Slowloris, Slow HTTP POST, Slow Read attack (based on TCP persist timer exploit) by draining concurrent connections pool, Apache Range Header attack by causing very significant memory and CPU usage on the server.

```
root@kali:~# slowhttpstest -h
slowhttpstest, a tool to test for slow HTTP DoS vulnerabilities - version 1.8.2
Usage: slowhttpstest [options ...]
Test modes:
  [-H          slow headers a.k.a. Slowloris (default)] [-B          slow body a.k.a R-U-Dead-Yet] [-R          range attack a.k.a Apache killer]
  [-X          slow read a.k.a Slow Read]
Reporting options:
  [-g          generate statistics with socket state changes (off)] [-o file_prefix save statistics output in file.html and file.csv (-g required)]
  [-v level    verbosity level 0-4: Fatal, Info, Error, Warning, Debug]
General options:
  [-c connections target number of connections (50)] [-i seconds interval between followup data in seconds (10)]
  [-l seconds   target test length in seconds (240)] [-r rate connections per seconds (50)]
  [-s bytes     value of Content-Length header if needed (4096)] [-t verb verb to use in request, default to GET for slow headers and response and to POST for slow body] [-u URL absolute URL of target (http://localhost/)]
  [-x bytes     max length of each randomized name/value pair of followup data per tick, e.g. -x 2 generates X-xx: xx for header or &xx=xx for body, where x is random character (32)] [-f content-type value of Content-type header (application/x-www-form-urlencoded)]
  [-m accept    value of Accept header (text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5)]
Probe/Proxy options:
  [-d host:port all traffic directed through HTTP proxy at host:port (off)] [-e host:port probe traffic directed through HTTP proxy at host:port (off)]
  [-p seconds   timeout to wait for HTTP response on probe connection, after which server is considered inaccessible (5)] [-j cookies value of Cookie header (ex.: -j "user_id=1001; timeout=9000")]
Range attack specific options:
  [-a start     left boundary of range in range header (5)] [-b bytes limit for range header right boundary values (2000)]
Slow read specific options:
  [-k num       number of times to repeat same request in the connection. Use to multiply response size if server supports persistent connections (1)] [-n seconds interval between read operations from recv buffer in seconds (1)]
  [-w bytes     start of the range advertised window size would be picked from (1)] [-y bytes end of the range advertised window size would be picked from (512)]
```

Figure 21: Example command of slowhttpstest

SlowHTTPTest allows security testers or network administrators to simulate such attacks in a controlled environment to assess the vulnerability of their web servers. By sending slow and partial HTTP requests, it can test how well the server responds and whether it can effectively mitigate Slowloris attacks.

4.3.3 Ncrack

Ncrack is an open-source network authentication cracking tool used for security auditing and penetration testing. Its primary purpose is to identify weak user credentials and perform brute-force attacks against various network services such as SSH, RDP, FTP, HTTP, and more. Ncrack is a powerful utility that allows security professionals to test the strength of passwords and find potential security vulnerabilities in a network infrastructure.

4.4 Test Methodology

The test methodology is divided into five distinct steps, with consideration given to the proper functioning of the main tools.

- The first step involves the use of Zeek online test environment ⁷ to analyze and evaluate the performance of the script, ensuring that it operates as intended.
- The second step, the script is integrated into the actual environment of Zeek within the Security Onion.
- The third step, the execution phase encompasses various tasks. These include selecting appropriate techniques, conducting network reconnaissance to identify potential victims, and executing the attack itself.
- The fourth step, the analysis of the Zeek log file to assess the effectiveness of the detection framework. This evaluation involves verifying if the framework successfully detects attacks in the experimental environment and generates new log files specific to the type of detection.
- Finally, in the last step, use of Intrusion Detection Evaluation Dataset⁸ CIC-IDS2017 and validate if the model can detect malicious activities against a large amount of normal and malicious traffic.

⁷ <https://try.bro.org/#/tryzeek> [Accessed: 2-May-2023].

⁸ <https://www.unb.ca/cic/datasets/ids-2017.html> [Accessed: 22-Jun-2023].

4.5 Attack Techniques

In this phase, we initially conduct various types of attacks to calibrate the model. Subsequently, we introduce a malicious dataset comprising attacks. Upon completion of each validation, we compare the count of zeek scripts generated with the count of suricata detected rules.

4.5.1 SYN Flood Test - T1499.001

This experiment is used to test and adjust the Syn flood detection module under malicious traffic generated using the following commands:

hping3 -S -U -i uxx -V -p <port number> --rand-source <IP>	
-S	set the SYN TCP Flag
-U	set the URG TCP Flag
-p	port specification
-V	verbose
--fast	mode enable
--rand-source	will send packets with random source addresses.
-i uX	- representing the amount of time should wait before sending the new packet. -i --interval wait (uX for X microseconds, for example -i u1000)

Table 4: Syn flood test parameters

The experiment uses different values of the time that should wait before sending the new packet and analyzes of the number of alerts that are generated by malicious traffic. These are counted using log files in IDS, table 5 shows the different intervals used in this experiment.

-i u1000 sends 100 packets per second.
-i u200 sends 500 packets per second.
-i u100 sends 1000 packets per second.

Table 5: Syn flood number of packets

During the attack, security onion generated a new log file called "icsha_syn_flood.log" shown in Figure 22, where we can analyze the source, destination, and port.

```
[root@securityonion ~]# ls /nsm/zeek/logs/current/
broker.log      conn.log      icsha_syn_flood.log  ntp.log      stats.log      stdout.log
capture_loss.log  dns.log      known_hosts.log     ssl.log      stderr.log     weird.log
[root@securityonion ~]#
```

Figure 22: Syn detection log file

After the training using Kali Linux tools, we tested and validated the script for DOS syn flood detection with the CICIDS2017. Figure 23 shows the log generated "icscha_syn_flood.log", origin IP, destination IP, port, and a brief description of the detection.

```
#path icscha_syn_flood
#open 2023-07-09-17-14-42
#fields tms v_host dst_host used_port used_msg
#types time addr addr port string
1499255020.523461 192.168.10.9 54.65.28.113 443 Possible DOS SYN flood attack was detected
1499255020.960266 192.168.10.9 54.65.28.113 443 Possible DOS SYN flood attack was detected
1499255021.225504 192.168.10.9 54.65.28.113 443 Possible DOS SYN flood attack was detected
1499255021.418665 192.168.10.9 52.197.242.182 443 Possible DOS SYN flood attack was detected
1499255022.047860 192.168.10.9 52.33.209.128 443 Possible DOS SYN flood attack was detected
1499255022.114639 192.168.10.9 52.197.242.182 443 Possible DOS SYN flood attack was detected
1499255022.254226 192.168.10.9 72.21.91.29 80 Possible DOS SYN flood attack was detected
1499255022.816733 192.168.10.9 52.197.242.182 443 Possible DOS SYN flood attack was detected
1499255023.007568 192.168.10.9 54.65.28.113 443 Possible DOS SYN flood attack was detected
1499255023.705916 192.168.10.9 54.65.28.113 443 Possible DOS SYN flood attack was detected
1499255024.407918 192.168.10.9 54.65.28.113 443 Possible DOS SYN flood attack was detected
1499255056.048747 192.168.10.12 202.55.13.210 443 Possible DOS SYN flood attack was detected
1499255056.998944 192.168.10.14 192.168.10.3 49666 Possible DOS SYN flood attack was detected
1499255056.998947 192.168.10.14 192.168.10.3 49666 Possible DOS SYN flood attack was detected
1499255056.999314 192.168.10.14 192.168.10.3 88 Possible DOS SYN flood attack was detected
1499255056.999362 192.168.10.14 192.168.10.3 88 Possible DOS SYN flood attack was detected
1499255057.431752 192.168.10.14 192.168.10.3 389 Possible DOS SYN flood attack was detected
1499255057.431755 192.168.10.14 192.168.10.3 389 Possible DOS SYN flood attack was detected
1499255057.437142 192.168.10.14 192.168.10.3 389 Possible DOS SYN flood attack was detected
1499255057.437146 192.168.10.14 192.168.10.3 389 Possible DOS SYN flood attack was detected
1499255057.537038 192.168.10.12 54.200.60.113 443 Possible DOS SYN flood attack was detected
1499255057.745949 192.168.10.12 72.21.91.29 80 Possible DOS SYN flood attack was detected
1499255057.765838 192.168.10.12 91.189.88.149 80 Possible DOS SYN flood attack was detected
1499255057.783099 192.168.10.12 91.189.88.149 80 Possible DOS SYN flood attack was detected
1499255057.783186 192.168.10.12 91.189.95.83 80 Possible DOS SYN flood attack was detected
1499255057.808742 192.168.10.12 202.55.13.210 443 Possible DOS SYN flood attack was detected
1499255058.064750 192.168.10.12 202.55.13.210 443 Possible DOS SYN flood attack was detected
1499255059.205769 192.168.10.14 64.4.54.36 443 Possible DOS SYN flood attack was detected
1499255059.440741 192.168.10.12 202.55.13.210 443 Possible DOS SYN flood attack was detected
1499255059.622117 192.168.10.14 65.52.98.233 443 Possible DOS SYN flood attack was detected
```

Figure 23: SYN flood detection log

4.5.2 DOS HTTP Flood Test - T1499.002

Denial of services of HTTP methods was carried out using kali-linux tools Slowhttptest, to observe and correct the script. The experimental test under malicious traffic was done in port 80, using the following commands:

slowhttptest -c 1000 -H -g -o slowhttp -i 10 -r 200 -t GET -u http://192.168.1.103/	
-c	Number of connection Slowloris mode
-H	Slowloris mode
-g	Generate statistics
-o	Output file name slowhttp.
-i	Seconds to wait for data
-r	Connections per second
-t	verb to use in request
-u	Target URL http://192.168.0.103/

Table 6: DOS HTTP Test parameters

This experiment uses different values of connection per second in order to observe the number of alerts that are generated by malicious traffic, table 7 shows the different test values used in this experiment.

Parameters	Get	Post
Total Number of connection	200	
Content-Length Header value	4096	
Connection per second	100	
	200	
	300	

Table 7: DOS HTTP test values

After different tests and detection model adjustments, we move to the next phase where CICIDS2017 was used. Figure 24 shows the result of the detection models log "icsha_http_methods.log" generated by Zeek in security onion after the injection of CICIDS2017 traffic.

```
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path icsha_http_methods
#open 2023-07-09-17-14-42
#fields tms src_host dst_host used_port used_method
#types time addr addr port string
1499255013.524092 192.168.10.14 13.107.4.50 80 high number of GET requests detected
1499255015.586684 192.168.10.14 13.107.4.50 80 high number of GET requests detected
1499255016.611005 192.168.10.14 13.107.4.50 80 high number of GET requests detected
1499255017.635904 192.168.10.14 13.107.4.50 80 high number of GET requests detected
1499255018.661582 192.168.10.14 13.107.4.50 80 high number of GET requests detected
1499255019.736802 192.168.10.14 13.107.4.50 80 high number of GET requests detected
1499255020.824124 192.168.10.14 13.107.4.50 80 high number of GET requests detected
1499255021.917682 192.168.10.14 13.107.4.50 80 high number of GET requests detected
1499255023.013833 192.168.10.14 13.107.4.50 80 high number of GET requests detected
1499255080.788056 192.168.10.14 23.194.181.138 80 high number of GET requests detected
1499255083.429486 192.168.10.14 198.41.215.182 80 high number of GET requests detected
1499255083.857867 192.168.10.14 198.41.215.182 80 high number of GET requests detected
1499255109.138705 192.168.10.12 72.21.91.29 80 high number of POST requests detected
1499255112.309293 192.168.10.12 72.21.91.29 80 high number of POST requests detected
1499255116.325097 192.168.10.14 184.84.243.199 80 high number of GET requests detected
1499255117.257639 192.168.10.14 72.21.91.29 80 high number of POST requests detected
1499255117.268929 192.168.10.14 72.21.91.29 80 high number of POST requests detected
1499255117.286485 192.168.10.14 72.21.91.29 80 high number of POST requests detected
1499255119.119819 192.168.10.14 172.217.11.46 80 high number of POST requests detected
1499255119.561653 192.168.10.14 172.217.11.46 80 high number of POST requests detected
1499255123.364860 192.168.10.14 13.107.4.50 80 high number of GET requests detected
```

Figure 24: DOS HTTP detection log

4.5.3 Brute Force Test - T1110

In the execution of brute force attack first, we identify the ip address and mask to be able to execute nmap and scan the network. Then we use nmap⁹ to recognize all active hosts on the network which ports 22 is

⁹ <https://nmap.org> Accessed: 22-Jun-2023].

open and hydra To perform ssh brute force. we execute the following command:

```
hydra -L <user.txt> -P password.txt ssh://192.168.0.102 -t 8
```

Figure 25, shows the result detection on Zeek's current log "/nsm/zeek/logs/current/icsha_bforce_ssh.log" where we can see the time stamp, source, destination, and the used port. The detection script can be seen in subsection 3.6.3.

```
#path icsha_bforce_ssh
#open 2023-07-11-17-36-48
#fields tms src_host dst_host used_port used_msg
#types time addr addr port string
1499188159.570965 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188159.883415 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188159.883415 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188159.883415 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188159.883415 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188159.883415 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188159.883415 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188159.883415 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188159.883415 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188172.018465 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188172.018465 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188172.163988 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188172.163988 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188172.225881 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188173.081661 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188173.081661 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188173.081661 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188173.215824 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188173.215824 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188185.608661 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
1499188185.608661 172.16.0.1 192.168.10.50 22 Potential brute force attack detected - Protocol SSH
```

Figure 25: Zeek brute force detection log

4.6 Final results, Evaluation, and Conclusions

The integration of normal traffic samples of industrial networks for model training, coupled with the predictable nature of industrial environments, has enabled the establishment of acceptable threshold values for defining normal behavior. This includes determining intervals for legitimate requests and the permissible frequency interval of such requests. By setting these threshold values, the generation of False Positive alerts has been significantly reduced.

However, it is essential to recognize that these threshold values may vary depending on the specific type of industrial environment being monitored. Different industrial sectors, such as energy, manufacturing, transportation, or water treatment, may exhibit distinct patterns of network activity and communication behavior. The unique characteristics and operational requirements of each industrial setting can influence the optimal threshold values for defining normal behavior.

Table 8 represents a sample of the results of the initial testing phase in the experimental environment, when generating partially malicious traffic to fine-tune the model, it became evident that the process re-

Number of packets	Number of syn flood alert
u1000	87 alerts
u200	493
u100	964

Number of connection	HTTP methods alerts
100	61
200	173
300	235

Table 8: Experimental test results

sulted in a significant consumption of VM resources, notably memory and processor utilization. These resource indicators, which suggest potential anomalies or malicious activities, could be effectively identified and detected using alternative tools since Zeek, in its current configuration, may not efficiently capture such anomalies.

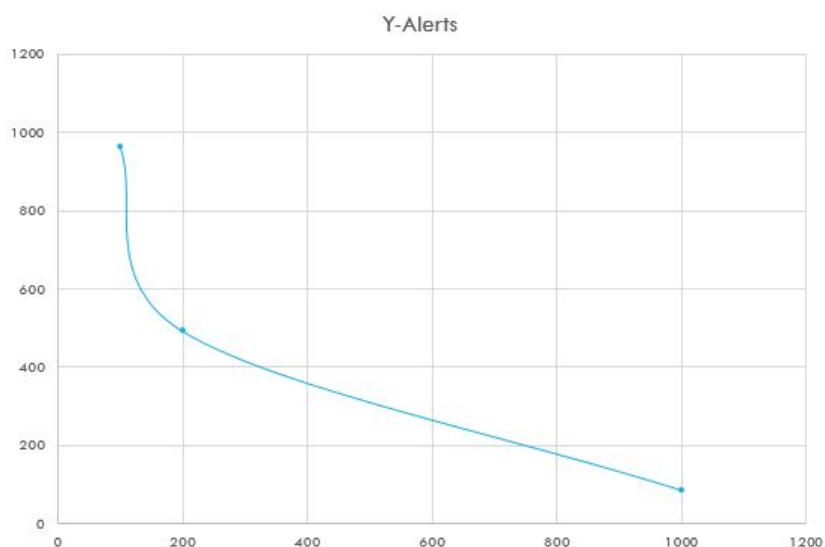


Figure 26: Number of syn flood alerts in experimental tests

The approximate results shown in Figure 26 and Figure 27 as well as in table 8 were taken from measurements over an interval of 8 to 12 seconds and with a threshold value of 10.

In the testing phase of the model, specifically in the detection of syn flood attacks, we utilized a dataset CIC-IDS2017 that possessed the largest volume and diversity of malicious traffic. Consequently, we suc-

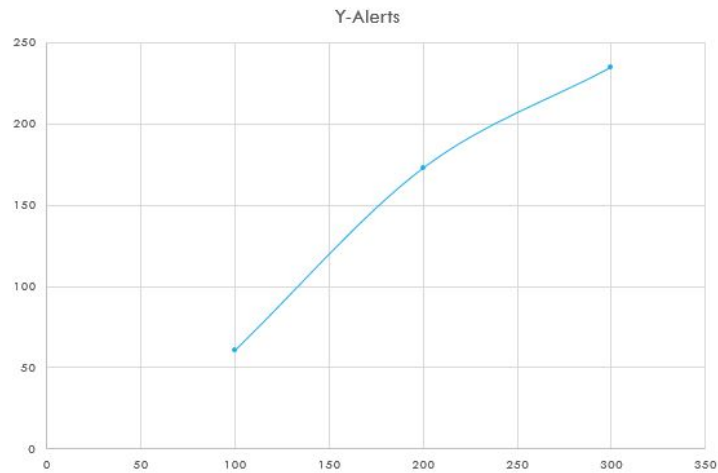


Figure 27: Number of HTTP get and post-flood alerts in experimental tests

cessfully identified a total of 6406 alerts, employing a threshold value of 50, taking into account that involves 172.16.0.1 to 192.168.10.50. It was also possible to detect 3638 of 4208 alerts HTTP methods.

However, using the same dataset Suricata detected a total of 4,884 alerts, which encompassed attacks targeting victim addresses and falling under the category "ET MALWARE Spoofed MSIE 7 User-Agent Likely Ponmocup cat A Network Trojan" with a high-risk classification. These findings were derived from a universe of 5,000 generated alerts. Additionally, It was possible to observe that during the initial experimental test phase conducted using Kali Linux in the aforementioned illustrated topology the description of the rule in Suricata, namely "ET DROP Spamhaus DROP Listed Traffic Inbound group 2" category, differed from the rule generated for attack alerts in the dataset.

In the context of a brute force attack detection test using the dataset, we were able to make some observations. Specifically, we identified a cumulative count of 617 alerts generated by Suricata, employing the defined rules "ET SCAN Potential SSH Scan OUTBOUND" and "ET SCAN Potential SSH Scan". Among these alerts, there were also additional occurrences related to true positive alerts involving different protocols, including FTP, SMB, and DNS. Furthermore, when utilizing the same dataset in Zeek, we observed a comparatively higher count of 964 alerts, indicating repeated attempts exceeding a threshold of three to gain access to port 22 which is the default port of ssh protocol.

Our detection model with script approach empowers administrators to identify potential brute-force attempts, SYN flood, HTTP get, and post-flood and strengthen the network's resilience against security

Technique	Zeek		Suricata	
	Number of Alerts	Description	Number of Alerts	Description
Syn flood attacks	4992 of 6406	Possible DOS SYN flood attack was detected	4884 of 5000	ET MALWARE Spoofed MSIE 7 User-Agent Ponmocup cat A Network Trojan
http flood (GET and Post)	3638 of 4208	high number of GET requests detected high number of POST requests detected		
Brute Force	964	Potential brute force attack detected - Protocol SSH	617	ET SCAN Potential SSH Scan OUTBOUND ET SCAN Potential SSH Scan

Table 9: Zeek and Suricata attacks detection of CICIDS2017 dataset.

threats. With careful consideration of variables, customization, and data analysis, the detection method achieves a delicate balance between accuracy and efficiency in identifying potential security incidents and also reduces the number of False-negative alerts in ICS environment.

In summary, Zeek’s scripting language serves as a powerful tool for network monitoring, analysis and anomaly behavior detection, enabling the extraction of valuable information, dynamic response mechanisms, and the enhancement of overall network security by generating detailed logs of network activity, which can be used to investigate suspicious or malicious activities. Its ability to analyze network traffic passively and generate application-layer transcripts makes it a valuable tool for network security professionals. By integrating Zeek with Security Onion, security administrators can efficiently process and analyze the data to identify potential security threats and respond to them in a timely manner.

It is essential to recognize the severity and potential consequences of such attacks on industrial control systems, as they can pose significant risks to critical infrastructure and public safety. Effective countermeasures and robust security strategies are crucial in safeguarding these systems against potential DoS attacks or brute force and ensuring their continued reliable operation.

This research may help in developing tailored and effective security measures to mitigate potential threats in industrial environments. By identifying potential weaknesses and evaluating the performance of Anomaly-based IDS, This study can assist in the development of robust and adaptive cybersecurity strategies for safeguarding industrial control systems. Besides all, it would be important to validate the models using distinct methodologies and real-world ICS traffic in controlled environments, evaluating the model alongside other approaches cited in the relevant literature would present a promising challenge and a significant stride towards advancement.

Chapter 5

Conclusions and future work

This last chapter presents the conclusions obtained in this dissertation, difficulties, and limitations as well as the proposals for future work.

5.1 Conclusions

Security of computer networks has become a paramount concern for all companies, and this concern becomes even more pronounced when these networks are integrated into industrial environments. Industrial control systems and critical infrastructure are prime targets for cyberattacks due to their potential impact on public safety, operations, and the economy. This thesis focuses on studying and evaluating anomaly detection-based IDS for industrial environments based on an open-source platform using a model approach and focusing on the exploration of remote services and DOS which are highly relevant and critical.

The research was guided through the ATT&CK Matrix for industrial environments and through generating a matrix that correlates the attacks, their indicators and the logs that can be used to mitigate them. An in-depth study of some tools presented by security onion was carried out with main emphasis on Zeek, its architecture, and script processing for generating alerts and how to record such logs according to the desired information. On the other hand, some attack techniques that can affect some components of industrial networks were studied, as well as some industrial network protocols and their main vulnerabilities.

As a result of this research and with the creation of a matrix, it was possible to develop a model that has three modules, the first for the detection of DOS syn flood, the second for the detection of DOS HTTP get and post flood and the third for the detection of brute force over the SSH protocol. The modules use zeek events and logs and statistical network packet time arrival to detect anomalies.

The parameters of this model were adjusted using a sample of normal traffic from industrial networks, it was tested in an experimental environment and validated using a dataset from the Canadian Institute for Cybersecurity that contains various malicious activities. The model revealed immense advantages, highlighting the reduction of false positive alerts, and the results are satisfactory.

Finally, we can conclude that Zeek can be used to detect attacks based on the analysis of network traffic behavior and it is essential to adopt a multi-layered and integrated approach to network security, combining the strengths of various monitoring tools to effectively identify and respond to cybersecurity threats in industrial control systems.

5.2 Limitations and Difficulties

Experiencing difficulties at each stage of this research was a constant aspect; however, as time progressed, we persevered and gained valuable insights from these challenges. Some of the difficulties and limitations we encountered were:

- The testing phase was delayed due to limitations in computing resources and there was a scarcity of practical material regarding configurations within the Security Onion environment.
- Limited literature, including articles and books, exists on the subject of utilizing scripts to generate Zeek logs and alerts within the Security Onion environment.
- The shortage of time-restricted the ability to conduct further tests encompassing various threshold values and different types of attacks.
- The lack of sufficient samples of Industrial Control Systems (ICS) traffic exhibiting both normal behavior and malicious activity, particularly in the context of engineering machines and servers connected to ICS, presents a significant challenge.

5.3 Prospect for Future Work

- Exploit the security onion in order to allow the sending of emails to the security managers after the detection of an attack by Zeek and build a graphical interface that allows changing the threshold values according to the environment.
- Explore other Zeek logs and events that can be used to detect abnormal behavior in industrial environments and frame logs involving protocols in industrial environments.
- Adapt the model to detect other types of attacks, such as malware, ftp and RDP brute force
- Build a model that exploits Zeek statistics to dynamically calculate threshold values.
- Finally, Exploring security onion for work related to network auditing thus opening up a field of research that is crucial nowadays.

References

- Alexander, O., Belisle, M., & Steele, J. (2020). *Mitre att&ck for industrial control systems: Design and philosophy*. Retrieved March 2020, from https://attack.mitre.org/docs/ATTACK_for_ICS_Philosophy_March2020.pdf
- Axelsson, S. (2000). *Intrusion detection systems: A survey and taxonomy*. Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden.
- Bou-Harb, E. (2020). Zeek intrusion detection series - cyber center for security and analytics.
- Cai, J., Wang, Q., Luo, J., Liu, Y., Liao, L., & fewer, S. (2021). *Capbad: Content-agnostic, payload-based anomaly detector for industrial control protocols*. IEEE Internet of Things Journal.
- Chang, C. P., Hsu, W. C., & Liao, I. E. (2019). *Anomaly detection for industrial control systems using k-means and convolutional autoencoder*. 27th International Conference on Software, Telecommunications and Computer Networks, SoftCOMI.
- Clotet, X., Moyano, J., & León, G. (2018). *A real-time anomaly-based ids for cyber-attack detection at the industrial process level of critical infrastructures*. International Journal of Critical Infrastructure Protection.
- Das, T. K., Adepur, S., & Zhou, J. (2020). *Anomaly detection in industrial control systems using logical analysis of data*. Computers and Security.
- Fan, X., Fan, K., Wang, Y., & Zhou, R. (2015). *Overview of cyber-security of industrial control system*. International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications.
- Farsi, H., Fanian, A., & Taghiyarrenani, Z. (2019). *A novel online state-based anomaly detection system for process control networks*. International Journal of Critical Infrastructure Protection.
- Feng, C., Li, T., & Chana, D. (2017). *Multi-level anomaly detection in industrial control systems via package signatures and lstm networks*. 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).
- Frazão, I., Abreu, P. H., Cruz, T., Araújo, H., & Simões, P. (2018). Denial of service attacks: Detecting the frailties of machine learning algorithms in the classification process.
- Jie, X., Wang, H., Fei, M., Du, D., Sun, Q., & Yang, T. C. (2018). *Anomaly behavior detection and reliability assessment of control systems based on association rules*. International Journal of Critical Infrastructure Protection.
- Kalech, M. (2019). *Cyber-attack detection in scada systems using temporal pattern recognition techniques*. Computers and Security.
- Khalili, A., Sami, A., Khozaei, A., & Poursmaeeli, S. (2019). *Abnormal detection method of industrial control system based on behavior model*. Computers and Security.
- Khan, I. A., Moustafa, N., Pi, D., Sallam, K. M., Zomaya, A. Y., & Li, B. (2021). *A new explainable deep learning framework for cyber threat discovery in industrial iot networks*. IEEE Internet of Things Journal.
- Knapp, E. D., & Langill, J. T. (2015). *Industrial network security, second edition securing critical infrastructure networks for smart grid, scada, and other industrial control systems*.
- Kruegel, C., Valeur, F., & Vigna, G. (2005). *Intrusion detection and correlation - challenges and solutions*. University of California, Santa Barbara, USA, Springer Science.
- Liu, B., Chen, J., & Hu, Y. (2022). *Mode division-based anomaly detection against integrity and availability attacks in industrial cyber-physical systems*. Computers in Industry.
- Liu, J., Yin, L., Hu, Y., Lv, S., & Sun, L. (2018). *A novel intrusion detection algorithm for industrial control systems based on cnn and process state transition*. IEEE 37th International Performance Computing and Communications Conference, IPCC.

- Myers, D., Suriadi, S., Radke, K., & Foo, E. (2018). *Anomaly detection for industrial control systems using process mining*. Computers and Security.
- Nagadevara, V. (2017). Evaluation of intrusion detection systems under denial of service attack in virtual environment.
- Security Onion Solutions, L. (2023). *security-onion*. Retrieved 26/04/2023, from <https://docs.securityonion.net/en/2.3/about.html>security-onion
- Sheng, C., Yao, Y., Fu, Q., & Yang, W. (2021). *A cyber-physical model for scada system and its intrusion detection*. Computer Networks.
- Stallings, W., & Brown, L. (2011). *Computer security principles and practice*. UNSW Canberra at the Australian Defence Force Academy.
- Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., & Thomas, C. B. (2020). *Mitre attck: Design and philosophy*. (Retrieved March 2020, from www.attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf)
- T1021, C. M. (2020). *Mitre att&ck*. (Accessed: 10/05/2023, from <https://attack.mitre.org/techniques/T1021/>)
- T1021/001/, C. M. (2020). *Mitre att&ck*. (Accessed: 10/05/2023, from <https://attack.mitre.org/techniques/T1021/001/>)
- T1021/004, C. M. (2020). *Mitre att&ck*. (Accessed: 10/05/2023, from <https://attack.mitre.org/techniques/T1021/004/>)
- T1110, C. M. (2013). *Mitre att ck*. Retrieved 07/05/2023, from <https://attack.mitre.org/techniques/T1110/>
- TA0108, T. C. M. (2018). *Mitre att&ck*. (Accessed: 05/05/2023, from <https://attack.mitre.org/tactics/TA0108/>)
- Tapiador, J., García-Teodoro, P., & Díaz-Verdejo, J. (2004). Anomaly detection methods in wired networks: A survey and taxonomy. *Computer Communications*, 27, 1569-1584. doi: 10.1016/j.com-com.2004.07.002
- Team, T. Z. P. (2023). *Zeek documentation*. Retrieved 26/04/2023, from <https://docs.zeek.org/en/master/index.html>
- Wang, Y., Fan, K., Lai, Y., Liu, Z., Zhou, R., Yao, X., & Li, L. (2017). *Intrusion detection of industrial control system based on modbus tcp protocol*. IEEE 13th International Symposium on Autonomous Decentralized Systems, ISADS.
- Wressnegger, J., Kellner, A., & Rieck, K. (2018). *Zoe: Content-based anomaly detection for industrial control systems*. 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN.
- Zang, X. D., Gong, J., & Hu, X. Y. (2019). *An adaptive profile-based approach for detecting anomalous traffic in backbone*. IEEE.
- Zhanwei, S., & Zenghui, L. (2019). *Abnormal detection method of industrial control system based on behavior model*. Computers and Security.
- Zhou, C., Huang, S., Xiong, N., Yang, S. H., Li, H., Qin, Y., & L, X. (2015). *Design and analysis of multimodel-based anomaly intrusion detection systems in industrial process automation*. IEEE Transactions on Systems, Man, and Cybernetics: Systems.

Part I

Appendices

Appendix A

Detection Details

A.1 Module icsha syn flood

```
event zeek_init()
{
    Log::create_stream(LOG, [$columns=Info, $path="icsha_syn_flood"]);
}
event tcp_packet(c: connection, is_orig: bool, flags: string, seq: count, ack: count, len: count, payload: string)
{
    #receive the ip addresss
    local ip = c$cid$orig_h;

    local calcu_interv: interval;
    local c_interv: double;

    calcu_interv = network_time() - interval_start;
    c_interv = interval_to_double(calcu_interv);

    #tcp.flags.syn == 1 and tcp.flags.ack == 0
    #Check if it is from the same source and the bit SYN is true according with type of log is "S"
    #if (is_orig && flags == "S" && ack ==0 )
    #if ( flags == "S" && ack ==0 )
    if (is_orig && flags == "S" && ack ==0 && c_interv <= 1.0 )
    {
        syn_flood_counter[ip] +=1;

        # Check if the SYN counter exceeds the threshold
        if (syn_flood_counter[c$cid$orig_h] > syn_flood_threshold)
        {
            #print fmt("Possible DOS SYN flood attack from %s was detected", c$cid$orig_h);
            Log::write(LOG, Info($tms=network_time(),
                $s_host=c$cid$orig_h,
                $dst_host=c$cid$resp_h,
                $used_port=c$cid$resp_p,
                $used_msg="Possible DOS SYN flood attack was detected"
            ));

            syn_flood_counter[c$cid$orig_h] = 0;
        }
    }

    if (flags == "S" && ack ==1 && c_interv <= 1.0 )
    {
        syn_flood_counter2 +=1;

        # Check if the SYN counter exceeds the threshold | maybe I can add here a interval
        if (syn_flood_counter2 > syn_flood_threshold)
        {
            #print fmt("Possible DOS SYN flood attack to %s was detected", c$cid$orig_h);
            Log::write(LOG, Info($tms=network_time(),
                $s_host=c$cid$resp_h,
                $dst_host=c$cid$orig_h,
                $used_port=c$cid$resp_p,
                $used_msg="Possible DOS SYN flood attack was detected"
            ));

            syn_flood_counter2 = 0;
        }
    }

    interval_start = network_time();
}
}
```

Figure 28: Syn flood detection code

A.2 Module icsha http methods

```
module icsha_http_methods;
export {
  ##psum - count the number of post and requests
  global psum: count = 0;
  global gsum: count = 0;
  const threshold_post_req: int = 50; # Number of POST requests threshold
  const threshold_get_req: int = 50; # Number of GET requests threshold

  global interval_start: time = 0;

  #generation of Logs
  redef enum Log::ID += { icsha_http_methods::LOG };
  type Info: record{
    tms: time &log;
    src_host: addr &log;
    dst_host: addr &log;
    used_port: port &log;
    used_method: string &log;
  };
}
event zeek_init()
{
  Log::create_stream(LOG, [$columns=Info, $path="icsha_http_methods"]);
}
event HTTP::log_http(c: HTTP::Info)
{
  local calcul_interv: interval;
  local c_interv: double;
  calcul_interv = network_time() - interval_start;
  c_interv = interval_to_double(calcul_interv);
  if (c$method == "POST" && c_interv <= 1.0 )
  {
    psum +=1;
    if (psum > threshold_post_req)
    {
      # print fmt("Abnormally high number of POST requests detected from %s within %s", c$id$orig_h, threshold_interval);
      Log::write(LOG, Info($tms=network_time(),
        $src_host=c$id$orig_h,
        $dst_host=c$id$resp_h,
        $used_port=c$id$resp_p,
        $used_method=" high number of POST requests detected"
      ));
      psum = 0;
    }
  }
  if (c$method == "GET" && c_interv <= 1.0 )
  {
    gsum +=1;
    if (gsum > threshold_get_req)
    {
      # print fmt("Abnormally high number of GET requests detected from %s within %s", c$id$orig_h, threshold_interval);
      Log::write(LOG, Info($tms=network_time(),
        $src_host=c$id$orig_h,
        $dst_host=c$id$resp_h,
        $used_port=c$id$resp_p,
        $used_method=" high number of GET requests detected"
      ));
      gsum = 0;
    }
  }
  interval_start = network_time();
}
```

Figure 29: Http flood detection code

A.3 Module icsha brute force ssh

```
module icsha_bforce_ssh;

export {

    # Track the number of attempt per user/source IP address
    global attempts_threshold: count = 3;

    #generation of Logs
    redef enum Log::ID += { icsha_bforce_ssh::LOG };

    #const allowed: table[subnet] of set[subnet] = {[192.168.1.104/32] = set(87.106.1.47/32, 87.106.12.77/32)} &redef;

    type Info: record{
        tms: time &log;
        src_host: addr &log;
        dst_host: addr &log;
        used_port: port &log;
    };
}

event zeek_init()
{
    Log::create_stream(LOG, [$columns=Info, $path="icsha_bforce_ssh"]);
}

# Access the SSH record as it is sent on to the logging
event SSH::log_ssh(c: SSH::Info)
{
    #Compare number of attempt with the defined threshold
    if (c$auth_attempts >= attempts_threshold) {
        print fmt("Potential brute force attack detected from IP: %s, to DST: %s, Port: %s, Protocol Version: %s.",
            c$id$orig_h, c$id$resp_h, c$id$resp_p, c$version);
        Log::write(LOG, Info($tms=network_time(),
            $src_host=c$id$orig_h,
            $dst_host=c$id$resp_h,
            $used_port=c$id$resp_p));
    }
}
}
```

Figure 30: Brute force detection code

Appendix B

Details of Tests

B.1 Security Onion configuration to integrate Zeek script

1. Add Custom scripts /opt/so/saltstack/local/salt/zeek/policy/custom/<\$custom-module>

```
[root@securityonion /]# ls /opt/so/saltstack/local/salt/zeek/policy/custom/ics/scripts/
icsha_bforce_ssh.zeek  __load__.zeek
[root@securityonion /]# ls /opt/so/saltstack/local/salt/zeek/policy/custom/ics/scripts2/
icsha_syn_flood.zeek  __load__.zeek
[root@securityonion /]# ls /opt/so/saltstack/local/salt/zeek/policy/custom/ics/scripts3/
icsha_http_methods.zeek  __load__.zeek
[root@securityonion /]#
```

Figure 31: Custom scripts folder

2. Add the custom folder to "/opt/so/saltstack/local/pillar/zeek/init.sls" and "/opt/so/saltstack-default/pillar/zeek/init.sls"

3. Set permissions on the file

```
chown socore:socore /opt/so/saltstack/local/pillar/zeek/init.sls
```

```
chown socore:socore /opt/so/saltstack/default/pillar/zeek/init.sls
```

4. Restart "zeek so-zeek-restart"

5. Check if the script is in loaded logs grep scrip "grep script /nsm/zeek/logs/current/loaded_scripts.log"

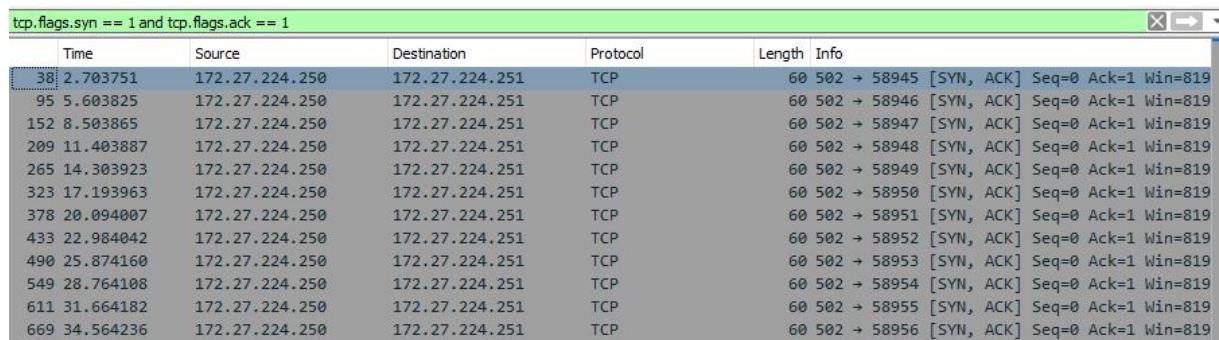
```
[root@securityonion /]# grep script /nsm/zeek/logs/current/loaded_scripts.log
#path loaded_scripts
/opt/zeek/lib64/zeek/plugins/packages/zeek-community-id/scripts/__load__.zeek
/opt/zeek/lib64/zeek/plugins/packages/zeek-af_packet-plugin/scripts/__load__.zeek
/opt/zeek/lib64/zeek/plugins/packages/zeek-af_packet-plugin/scripts/init.zeek
/nsm/zeek/spool/installed-scripts-do-not-touch/auto/cluster-layout.zeek
/nsm/zeek/spool/installed-scripts-do-not-touch/site/local.zeek
/opt/zeek/share/zeek/policy/custom/ics/scripts/__load__.zeek
/opt/zeek/share/zeek/policy/custom/ics/scripts/icsha_bforce_ssh.zeek
/opt/zeek/share/zeek/policy/custom/ics/scripts2/__load__.zeek
/opt/zeek/share/zeek/policy/custom/ics/scripts2/icsha_syn_flood.zeek
/opt/zeek/share/zeek/policy/custom/ics/scripts3/__load__.zeek
/opt/zeek/share/zeek/policy/custom/ics/scripts3/icsha_http_methods.zeek
```

Figure 32: Loaded scripts

B.2 Datasets

B.2.1 Normal behavior dataset

The normal behavior without attacks, normal testbed operation was extracted from the paper "Denial of Service Attacks: Detecting the frailties of machine learning algorithms in the Classification Process" (Frazão, Abreu, Cruz, Araújo, & Simões, 2018). The environment involves a small-scale process automation scenario that was developed utilizing MODBUS/TCP equipment. The objective of this scenario was to create a testbed that emulates a Cyber-Physical System (CPS) process, which was controlled by a Supervisory Control and Data Acquisition (SCADA) system using the MODBUS/TCP protocol.



The image shows a Wireshark packet capture table with the filter `tcp.flags.syn == 1 and tcp.flags.ack == 1`. The table contains 15 rows of data, each representing a packet. The columns are Time, Source, Destination, Protocol, Length, and Info. The packets are all TCP SYN-ACK packets from source 172.27.224.250 to destination 172.27.224.251. The sequence numbers range from 58945 to 58956, and the acknowledgment numbers range from 819 to 819. The time intervals between packets are approximately 2.5 ms.

Time	Source	Destination	Protocol	Length	Info
38 2.703751	172.27.224.250	172.27.224.251	TCP	60	502 → 58945 [SYN, ACK] Seq=0 Ack=1 Win=819
95 5.603825	172.27.224.250	172.27.224.251	TCP	60	502 → 58946 [SYN, ACK] Seq=0 Ack=1 Win=819
152 8.503865	172.27.224.250	172.27.224.251	TCP	60	502 → 58947 [SYN, ACK] Seq=0 Ack=1 Win=819
209 11.403887	172.27.224.250	172.27.224.251	TCP	60	502 → 58948 [SYN, ACK] Seq=0 Ack=1 Win=819
265 14.303923	172.27.224.250	172.27.224.251	TCP	60	502 → 58949 [SYN, ACK] Seq=0 Ack=1 Win=819
323 17.193963	172.27.224.250	172.27.224.251	TCP	60	502 → 58950 [SYN, ACK] Seq=0 Ack=1 Win=819
378 20.094007	172.27.224.250	172.27.224.251	TCP	60	502 → 58951 [SYN, ACK] Seq=0 Ack=1 Win=819
433 22.984042	172.27.224.250	172.27.224.251	TCP	60	502 → 58952 [SYN, ACK] Seq=0 Ack=1 Win=819
490 25.874160	172.27.224.250	172.27.224.251	TCP	60	502 → 58953 [SYN, ACK] Seq=0 Ack=1 Win=819
549 28.764108	172.27.224.250	172.27.224.251	TCP	60	502 → 58954 [SYN, ACK] Seq=0 Ack=1 Win=819
611 31.664182	172.27.224.250	172.27.224.251	TCP	60	502 → 58955 [SYN, ACK] Seq=0 Ack=1 Win=819
669 34.564236	172.27.224.250	172.27.224.251	TCP	60	502 → 58956 [SYN, ACK] Seq=0 Ack=1 Win=819

Figure 33: Normal behavior syn ack interval

In Figure 33 shown filter for SYN packets with an acknowledgment using the following filter: `tcp.flags.syn == 1 and tcp.flags.ack == 1`. We also check SYN packets without an acknowledgment using the following filter: `tcp.flags.syn == 1 and tcp.flags.ack == 0` and identified that the interval between the then is high then 2.5 ms. Figure 34 shows Wireshark's graphs for a visual representation of the uptick in traffic and the number of packets per second.

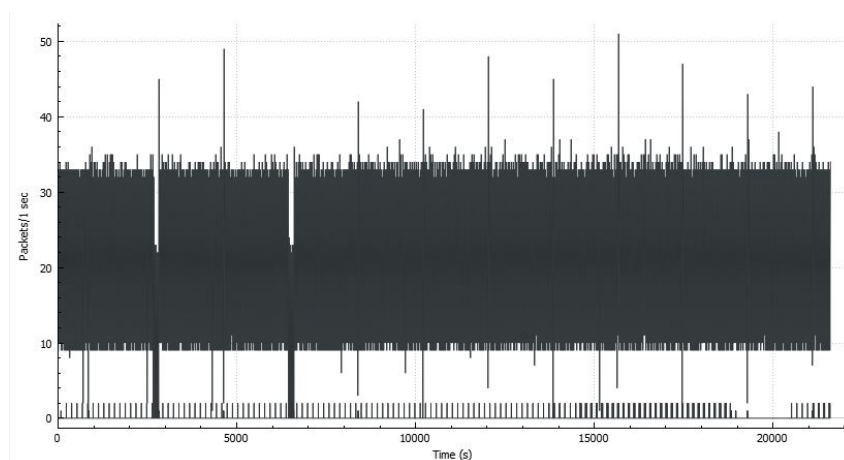


Figure 34: Normal behavior packet per second

B.2.2 CICIDS201 Dataset

Intrusion Detection Evaluation Dataset (CIC-IDS2017) was created to test and validate anomaly-based intrusion detection models. The topology involve use of Modem, Firewall, Switches, Routers, differents operating systems like Windows, Ubuntu and Mac OS X. Furthermore were generated 25 users traffic based on the HTTP, HTTPS, FTP, SSH, and email protocols.

Day	Description	Technique	Hosts	Process	
Tuesday July 4, 2017	attacks + Normal Activity	Brute Force FTP and SSH	Attacker: Kali, 205.174.165.73 Victim: Webserver Ubuntu, 205.174.165.68 (Local IP: 192.168.10.50)	Attack: 205.174.165.73 -> 205.174.165.80 (Valid IP of the Firewall) -> 172.16.0.1 -> 192.168.10.50 Reply: 192.168.10.50 -> 172.16.0.1 -> 205.174.165.80 -> 205.174.165.73	11G
Wednesday July 5, 2017	attacks + Normal Activity	DoS / DDoS HTTP, HTTPS Heartbleed	Attacker: Kali, 205.174.165.73 Victim: Webserver Ubuntu, 205.174.165.68 (Local IP:192.168.10.50)	Attack: 205.174.165.73 -> 205.174.165.80 (Valid IP of the Firewall) -> 172.16.0.1 -> 192.168.10.50 Reply: 192.168.10.50 -> 172.16.0.1 -> 205.174.165.80 -> 205.174.165.73	13G

Table 10: CICIDS201 dataset information