Pontifícia Universidade Católica do Paraná
Programa de Pós-Graduação em Informática

Machine Learning-based Integral Defense-in-Depth to Provide Security in Industrial Control Systems

Paulo Roberto de Oliveira

Advisor

Altair Olivo Santin

Co-Advisor

Eduardo Kugler Viegas

PUCPR

Curitiba
2023

Pontifícia Universidade Católica do Paraná
Programa de Pós-Graduação em Informática

Machine Learning-based Integral Defense-in-Depth to Provide Security in Industrial Control Systems

Paulo Roberto de Oliveira

Thesis presented to the *Programa de Pós-Graduação em Informática* as a partial requirement for the degree of Doctor in Informatics.

Major Field: Computer Science

Supervisor: Altair Olivo Santin

Co-supervisor: Eduardo Kugler Viegas

Curitiba
2023

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Sônia Maria Magalhães da Silva – CRB 9/1191

O48m
2023

Oliveira, Paulo Roberto de
Machine learning-based integral defense-in-depth to provide security in industrial control systems / Paulo Roberto de Oliveira ; advisor: Altair Olivo Santin ; co-advisor: Eduardo Kugler Viegas. – 2023
78 f. ; il. : 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2023
Bibliografia: f. 72-78

1. Sistemas de detecção de intrusão (Segurança do computador). 2. Sistemas de controle supervisório. 3. Aprendizado do computador. 4. Informática. I. Santin, Altair Olivo. II. Viegas, Eduardo Kugler. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD. 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

Curitiba, 06 de fevereiro de 2024.

08-2024

DECLARAÇÃO

Declaro para os devidos fins, que **PAULO ROBERTO DE OLIVEIRA** defendeu a tese de Doutorado intitulada "**Machine Learning-based Integral Defense-in-Depth to Provide Security in Industrial Control Systems**", na área de concentração Ciência da Computação no dia 07 de dezembro de 2023, no qual foi aprovado.

Declaro ainda, que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade firmo a presente declaração.

gov.br
Documento assinado digitalmente
EMERSON CABRERA PARAISO
Data: 06/02/2024 14:33:13-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática

I dedicate this work to my family for unconditional support.

Success is the sum of small efforts repeated day in and day out. - Robert Collier

Abstract

One of the recommended techniques for protecting Supervisory Control and Data Acquisition (SCADA) systems is to develop a Defense in Depth (DiD) security mechanism for Industrial Control System (ICS). However, in the case of a vulnerability existing in the DiD, it may be possible for an attacker to gain control of diverse security mechanisms in the DiD layers and control the system completely. An improved implementation of DiD using the Service Function Chaining (SFC) is presented in this thesis. Using Machine Learning (ML), SFC flows are used to classify and route the network traffic through DiD dynamically. This proposal adopts the diversity of security mechanisms in DiD layers. Since SFC routing flows are inaccessible to an attacker and diversity is adopted, we assume that the attacker cannot control at least one security mechanism in each DiD layer. The DiD layers use the Network Intrusion Detection System (NIDS) and the Deep Packet Inspection (DPI) as security mechanisms. The Security Monitoring Systems (SMS), a machine learning-based anomaly-based detection system, validates the classifications of the NIDS and the DPI. The SMS is 28.6% more reliable in classifying traffic, outperforming security tools available in the literature.

Key-words: Defense-in-Depth; Industrial Control Systems; Intrusion Detection System; Machine Learning Classification; Machine Learning-Based Anomaly Detection; Operation Technology; Service Function Chain; Supervisory Control and Data Acquisition

Resumo

Uma das técnicas recomendadas para proteger os sistemas SCADA (Supervisory Control and Data Acquisition, Controle de Supervisão e Aquisição de Dados) é desenvolver um mecanismo de segurança DiD (Defense-in-Depth, Defesa em Profundidade) para os ICS (Industrial Control Systems, Sistemas de Controle Industrial). No entanto, no caso de uma vulnerabilidade existente no DiD, pode ser possível que um invasor obtenha o controle de diversos mecanismos de segurança nas camadas do DiD e controle completamente o sistema. Nesta tese, é apresentada uma implementação aprimorada do DiD usando a cadeia de funções de serviço (SFC). Usando o aprendizado de máquina, os fluxos de SFC são usados para classificar e rotear o tráfego de rede por meio do DiD dinamicamente. Essa proposta adota a diversidade de mecanismos de segurança nas camadas de DiD. Como os fluxos de roteamento SFC são inacessíveis a um invasor e a diversidade é adotada, presumimos que o invasor não pode controlar pelo menos um mecanismo de segurança em cada camada DiD. As camadas DiD usam o sistema de detecção de intrusão de rede (NIDS) e a inspeção profunda de pacotes (DPI) como mecanismos de segurança. O Sistema de Monitoramento de Segurança (SMS), um sistema de detecção de anomalias baseado em aprendizado de máquina, valida as classificações do NIDS e da DPI. O SMS é 28,6% mais confiável na classificação do tráfego, superando as ferramentas de segurança disponíveis na literatura.

Palavras-chave: Defesa em Profundidade; Sistemas de Controle Industrial; Sistema de Detecção de Intrusão; Classificação de Aprendizado de Máquina; Detecção de Anomalias Baseada em Aprendizado de Máquina; Tecnologia de Operação; Cadeia de Funções de Serviço; Controle de Supervisão e Aquisição de Dados

Acknowledgements

First of all, I would like to thank God for the opportunity to complete this work.

I thank my advisors Altair Olivo Santin and Eduardo Kugler Viegas for all their support and teachings during this period.

To the colleagues of the SecPLab group for all the exchange of knowledge during this journey, as well as the understanding of various issues related to my work.

To the professors of the Graduate Program in Computer Science (PPGIa), for so much dedication and work for the benefit of science.

To my family, for all the support during this time, my wife, my son, my parents and my siblings.

To CAPES for helping me to study.

And not later, I would like to thank my colleague Valéria for her great help in this last part of the journey.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

CAPEX	Capital Expense
CNN	Convolutional Neural Network
CVE	Common Vulnerabilities and Exposures
DDoS	Distributed Denial of Service
DiD	Defense in Depth
DoD	Department of Defense
DPI	Deep Packet Inspection
DT	Decision Tree
FN	False Negative
FP	False Positive
GB	Gradient Boosting
GNB	Gaussian Naive Bayes
HIDS	Host-based Intrusion Detection System
HMI	Human-Machine Interface
ICS	Industrial Control System
IDS	Intrusion Detection System
IT	Information Technology
IPS	Intrusion Prevention System

ML	Machine Learning
MLP	MultiLayer Perceptron
NIDS	Network Intrusion Detection System
NFV	Network Function Virtualization
NTP	Network Time Protocol
OS	Operating System
OT	Operation Technology
ONF	Open Network Foundation
PCAP	Packet Capture
Pentest	Penetration Testing
PLC	Programmable Logic Controllers
RF	Random Forest
ROC	Receiver Operator Characteristic
RSP	Rendered Service Path
SaaS	Software-as-a-Service
SCADA	Supervisory Control and Data Acquisition
SDN	Software-Defined Network
SF	Service Function
SFC	Service Function Chaining
SFF	Service Function Forwarder
SFP	Service Function Path
SMS	Security Monitoring Systems
SQL	Structured Query Language

SSL Secure Socket Layer

TN True Negative

TP True Positive

XSS Cross-Site Script

1 Introduction

1.1 Contextualization

In the cybersecurity domain, Möller et al. (2018) asserts that potential adversaries encompass a wide range of entities, from nations and terrorists to criminals, hackers, and business competitors. These adversaries are driven by a variety of motivations, ranging from intelligence gathering and intellectual property theft to conducting denial-of-service attacks, causing embarrassment, or simply enjoying having exploited a high-profile target. Understanding the range of potential adversaries and their underlying motivations is paramount to understanding the diverse landscape of cybersecurity threats and developing effective countermeasures.

As we have seen in recent years, a large proportion of computers in Industrial Control System (ICS) environments have been attacked. In 2022 alone, 21.93% of ICS computers were attacked with malware (CERT, 2023). These are only the attacks that were blocked. The real number could be much higher.

Industrial environments are often part of an ICS (TRENDMICRO, 2023). Such systems are important because they control automated industrial processes, systems, devices, and networks. Therefore, critical data is carried in an ICS as the systems receive it from remote sensors used to control important industrial processes. A Supervisory Control and Data Acquisition (SCADA) system is typically used to manage an ICS.

ICS are often deployed in organizations that provide essential or even dangerous services to the public, such as energy supply, and are therefore considered critical systems. Any failure or malfunction within these systems can result in environmental harm, substantial financial losses, or even loss of life (GHEORGHE et al., 2006).

One of the most famous cases of ICS attacks was Stuxnet (LANGNER, 2011), a malware considered to be the most sophisticated at the time, specifically designed to target the SCADA controlling uranium enrichment centrifuges. This attack occurred in 2010 at a nuclear power plant in Iran, using malware capable of reprogramming Programmable Logic Controllers (PLC). Stuxnet exploited four vulnerabilities in Windows systems, two of which were known and two of which were unknown at the time.

Another major ICS security issue was caused by the Black Energy tool (HEMSLEY; FISHER et al., 2018), which was used to attack a utility company in Ukraine and was used as a Trojan horse for DDoS attacks, digital espionage, and information gathering. One target was the SCADA system, as a specific plug-in was created for it in the attack tool, which exploited a vulnerability in MS Office. The attack took place in 2015, and the attackers gained access to the company's SCADA network, allowing them to shut down 30 substations and leave many Ukrainian users without electricity for days.

These two examples of real-world attacks occurred because the third generation of SCADA began to support Ethernet and TCP/IP architecture as connectivity capabilities. However, developers were not prepared to address security in the Operation Technology (OT) environment as they did in the information technology Information Technology (IT) environment. Therefore, there was room to start using Intrusion Detection System (IDS) in OT.

In general, SCADA can also physically distribute processing as a strategy to survive a total loss of control of a specific system location. However, in OT with ICS, these characteristics require following a standardization such as IEC 62443 (COMMISSION et al., 2018).

IEC 62443 proposes Defense in Depth (DiD), whose goal is to impose multiple and different defensive barriers (security mechanisms organized in multilayers) on the path between a possible starting point for an attack (usually the internet) and its target, commonly a SCADA system (MELL; SHOOK; HARANG, 2016).

The layer-based layout of DiD assumes that if a mechanism is vulnerable in one layer, the same vulnerability will not occur in another layer. Therefore, if the arrangement of defensive mechanisms is well constructed, the overall security of DiD remains intact even if a particular mechanism has vulnerabilities in each layer.

Now suppose that more than one tool is susceptible to vulnerabilities in the DiD layers. In such a scenario, we can deduce two possible behaviors of an attacker faced with the DiD security approach. First, the attacker does not want to attack the vulnerable tool and gives up because easier targets are available. In contrast, in a second scenario, an attacker may spare no effort to attack the vulnerable tool in the DiD, targeting the ICS.

In general, the DiD must provide resistance to the attacker to make him give up. However, if the attacker is highly motivated to break into the ICS target, he may be able to control at least one defense mechanism in each layer of the multi-layered security arrangement that complicates the DiD.

Admittedly, the DiD security barrier arrangement imposes resistance on attackers by requiring more complex work on the part of the attacker. However, it is important to keep in mind that the essence of the DiD protection scheme has a significant limitation in that the multi-layered scheme can be compromised individually and the attacker can reach the SCADA in the ICS. The idea of making the ICS completely secure by adding layers of security is only valid if at least one of the DiD layers has no vulnerability, which cannot be guaranteed and cannot be considered reliable.

1.2 Motivation

As previously mentioned, when attackers encounter layers of security before reaching their target, there are two possible scenarios. They may either choose to abandon their efforts and seek out an easier target, or they may persist in their attempts to escalate through each layer, carefully examining potential vulnerabilities until they reach the ICS target.

In the latter scenario, attackers may be successful regardless of the time spent finding vulnerabilities and controlling security layers. To provide high-level security for systems and networks, the DiD technique depends on several aspects, including the level of security tool (in an isolated analysis), the best mix of security tools, the number of employed layers, and the attacker's profile and goals. This problem motivated us to propose a novel approach that utilizes the concepts of DiD as recommended by IEC 62443.

1.3 Hypothesis

A DiD security architecture with multi-layered integration and a late-fusion classification engine makes the Intrusion Detection System (IDS) of ICS more reliable.

1.4 Objectives

The aim of this thesis was to design and implement an SFC architecture that can enhance the security level of critical infrastructures, including industrial control systems. The architecture must be comprehensive to prevent potential attackers from manipulating security layers such as IDS, Deep Packet Inspection (DPI), etc. This is accomplished by preventing attackers from controlling the flows within the SFC-based DiD environment. These flows are managed dynamically based on the type of incoming traffic. Machine learning algorithms determine the optimal path, mitigating potential threats.

1.4.1 Specific Objectives

To achieve the objective of this work, we delineated the following specific objectives:

- To develop a dataset containing ICS and IT traffic information among clients, SCADA Server, security tools, and possible attacks against it.
- To develop an Integral Defense in Depth model based on SFC.
- To develop the Security Monitoring Systems (SMS), since it is a key aspect of this work.
- To integrate the SMSs to the Integral Defense in Depth model.
- To develop a Proof of Concept to demonstrate that the integral Defense in Depth model, along with the SMS, provides a higher level of security and represents an interesting choice for protecting critical infrastructures.

1.5 Contributions

The main contributions of this work are the following:

- The DiD technique is implemented in an integrated way using an SFC flow classifier that applies machine learning to dynamically select the most likely security tool to identify an attack. This aims to prevent an attacker from controlling a security mechanism in a DiD layer.
- The late-fusion classifier offers a solution that can improve the reliability of traffic classification. To ensure more accurate decision-making, multiple layers are utilized to determine whether the traffic is an attack or not. Such validation is presented in Chapter 5.

Some by-products generated from this work are the following:

- A dataset collecting events classified as attacks by security mechanisms on various platforms. Its purpose is to model the attacks in a pre-classifier to provide security isolation through SFC technology implementing the multilayer DiD. The goal is to prevent attackers from knowing the security barrier they will face until reaching the ICS.
- A NIDS and Host-based Intrusion Detection System (HIDS) specific to SCADA systems, implemented through late-fusion classification and embedded in the SMS, which will be discussed later in this thesis.

1.6 Publications

This section presents a list of papers that have been published or are under review, addressing security challenges in ICS environments. The table 1.1 includes the titles of the publications, the Brazilian Qualis, the places of publication, and the list of authors.

Table 1.1: Performed Publications During This Work.

Title	**Qualis**	**Conference/Journal**	**Authors**
A Dynamic Machine Learning Scheme for Reliable Network-Based Intrusion Detection	A3	The 37th International Conference on Advanced Information Networking and Applications (AINA-2023)	Eduardo K. Viegas, Everton de Matos, Paulo R. de Oliveira, Altair O. Santin
A Dynamic Network-based Intrusion Detection Model for Industrial Control Systems	A2	The 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2023)	Paulo R. de Oliveira, Altair O. Santin, Pedro Horchulhack, Eduardo K. Viegas, Everton de Matos
Toward a Reliable Network-based Intrusion Detection Model for SCADA: a Classification with Reject Option Approach **(Under review)**	A2	The 39th ACM/SIGAPP Symposium On Applied Computing	Paulo R. de Oliveira, Altair O. Santin, Pedro Horchulhack, Eduardo K. Viegas, Everton de Matos
Defense-in-Depth and Machine Learning-based Intrusion Detection for Industrial Control Systems **(Under review)**	A1	Computer Networks (Journal)	Paulo R. de Oliveira, Altair O. Santin, Pedro Horchulhack, Eduardo K. Viegas, Aldri Santos

1.7 Organization

The outline of the paper is as follows. Section 2 deals with the background, while Section 3 reports on related work. Section 4 presents the proposed machine learning-based defense in depth to provide security in ICS. Section 5 presents prototype and experimental results, and finally, Section 6 concludes and discusses future work.

2 Background

This chapter presents the theoretical foundation related to Industrial Control Systems, Defense in Depth, Software Defined Networks, Network Function Virtualization, Service Function Chaining, Machine Learning and Honeypot techniques. These concepts are essential foundations for understanding this thesis.

2.1 Industrial Control Systems

Industrial Control System (ICS) are integrated architectures used to manage and control industrial processes and associated infrastructure in sectors such as manufacturing, energy, and transportation (KAYAN et al., 2022). It integrates hardware, software, and network components to enable their automation and optimize industrial processes (TRENDMICRO, 2023), including machinery, sensors, data acquisition, and communication interfaces.

The SCADA system is widely used in ICSs, as it enables the control and automation of Programmable Logic Controllers (PLC) through an Human-Machine Interface (HMI), using various protocols such as Modbus, DNP3, and OPC for seamless communication and data exchange (ALANAZI; MAHMOOD; CHOWDHURY, 2022). Figure 2.1 presents the main components of SCADA Systems. Those systems are not full control systems but are the major representative of ICS hardware and software and operating at the supervisory level (DANEELS; SALTER, 1999).

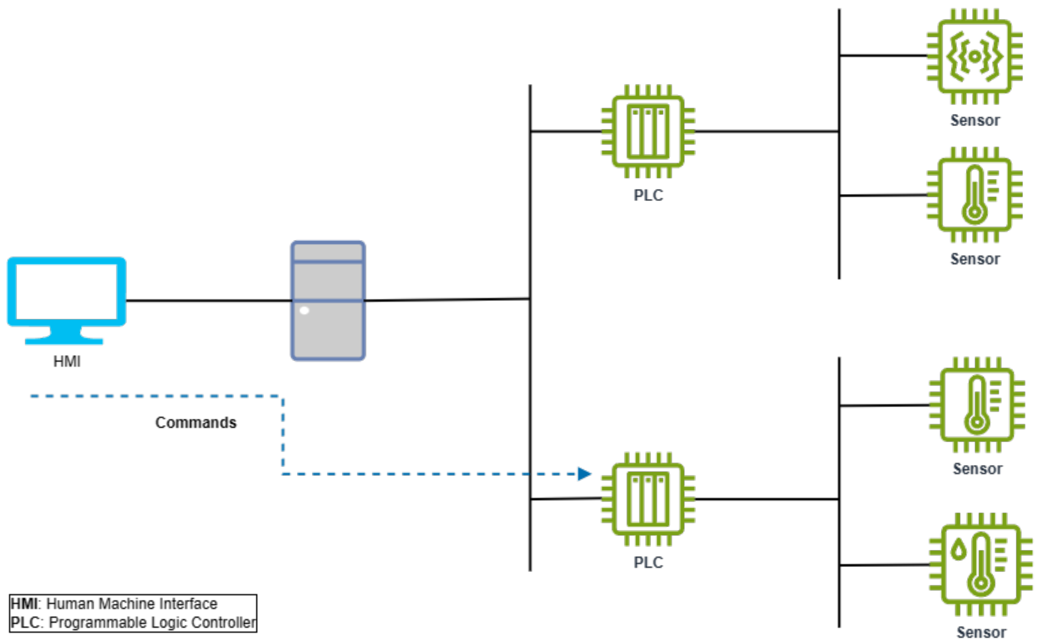

Sensor
Sensor
PLC
HMI
Commands
Sensor
PLC
Sensor
HMI: Human Machine Interface
PLC: Programmable Logic Controller

Figure 2.1: The Communication Components of a SCADA System. Adapted from (LIU, 2022)

In this context, MODBUS plays a crucial role as a widely used communication protocol in industrial automation and control systems (THOMAS, 2008). It provides a simple and effective way for devices such as sensors, actuators, and controllers to exchange data on the same network. With support for both serial communication and TCP/IP, MODBUS is versatile and facilitates communication between devices from different manufacturers in SCADA systems.

SCADA systems have several main functions, including local and remote control of industrial processes, real-time data gathering, monitoring, and processing, exchanging data with various devices such as sensors, motors, valves, and others, and recording events into log files (AUTOMATION, 2018).

Critical infrastructures, such as SCADA systems, are increasingly becoming targets for potential attackers due to their connection with other networks. ICS users and developers have noticed a significant increase in cyber-attacks on their systems (KASPERSKY, 2023).

Attacks on SCADA systems can have physical manifestations in the real world, as noted by (MILLER; ROWE, 2012). This means that an attack launched in the virtual environment can have significant physical consequences within an organization or company. As a result, SCADA systems are highly attractive to attackers who seek to interrupt or damage physical devices or services. These factors have contributed to the increase in attacks on SCADA systems.

Attackers are highly motivated to disrupt ICS systems due to their critical nature. They may even utilize multiple zero-day vulnerabilities to achieve their goal, as noted by (SHENG et al., 2023). In practice, attackers analyze their target ICS infrastructure over extended periods to effectively craft their attacks.

To ensure comprehensive safeguarding of these systems, operators must implement a range of security solutions, including authentication, authorization, firewalls, and VPNs. NIDS tools are commonly used to evaluate ICS network traffic.

Thus, DiD is also one of the techniques proposed to mitigate security problems in ICS environments, as it can use different tools to compose a strong and complex security solution (COMMISSION et al., 2018). Therefore, the DiD concept will be discussed in the next subsection.

2.2 Defense in Depth

Defense in Depth (DiD) is a strategy designed to strengthen the overall defense by implementing a series of security measures in a cascading fashion (COMMISSION et al., 2018), (MELL; SHOOK; HARANG, 2016), (SECURITY, 2016).

For network security, DiD is the practice of implementing multiple layers of defensive barriers between potential attackers and their desired targets. This approach aims to provide a comprehensive and robust security architecture that can withstand various types of attacks. (MELL; SHOOK; HARANG, 2016). As stated by Runnels (2002), A security tool alone cannot be considered foolproof, as threats can exist within the tools themselves and attackers are constantly improving their attack methods. Therefore, the DiD technique is proposed as a means of minimizing such security issues.

In information security, this strategy involves creating layers of barriers to hinder potential attackers. Each layer is dedicated to a distinct security solution, including firewalls, traffic monitors, Deep Packet Inspection (DPI), Intrusion Prevention System (IPS), Intrusion Detection System (IDS), among others.

Runnels (2002) also states that the DiD approach establishes layers of defense that reinforce each other to reduce vulnerabilities and aid in the detection and response to potential attacks. Consequently, when attackers discover and exploit a vulnerability, they encounter another barrier (layer) and the cycle continues, deterring unauthorized access attempts.

In essence, DiD functions as a mechanism designed to delay and frustrate attackers, forcing them to expend more resources and time to breach the system or successfully achieve their objectives.

In a practical implementation, several questions arise when working with DiD, such as Is there a single optimal model? How many layers should be built? Is there a minimum number of layers to use? What are the constraints and who is responsible for these layers? (RUNNELS, 2002).

The answer to the first question is that there is no single best model, because the techniques used in each type of attack can be very different. The model proposed by the US Department of Defense (DoD) suggests the use of three broad layers: people, operations, and technology. On the other hand, commercial developers model it by focusing on their products, such as antivirus, firewalls, and others. Other approaches mix the two earlier models, using security tools, employee training, and physical security.

The answers to the remaining questions depend on the specific environment in which the mitigation mechanism is deployed (whether it's a critical environment or not), the desired level of security, and the sensitivity of the data in transit, among other factors. Consequently, there is no one-size-fits-all answer to these questions, as they should be tailored to the unique security requirements of each company or organization.

Therefore, certain aspects to consider when building a robust DiD strategy include (i) the isolation level of each security tool, (ii) the diversity of tools within a given system/network, and (iii) the interconnection of security layers. (SECURITY, 2016).

To ensure that DiD is effective, a vulnerability present in one layer cannot be present in the next layer. Thus, each layer can "compensate" for possible vulnerabilities in earlier layers. Figure 2.2 shows a four-layer DiD scheme, its vulnerabilities (small ellipses) and possible exploits (arrows).

DiD finds significant applications within cloud computing application architectures. For example, in a Software-as-a-Service (SaaS) model, various layers separate users from the underlying application data (LYONS, 2011). It also plays a critical role in scenarios involving repositories of sensitive data where strict access controls are imperative, such as in federal government systems.

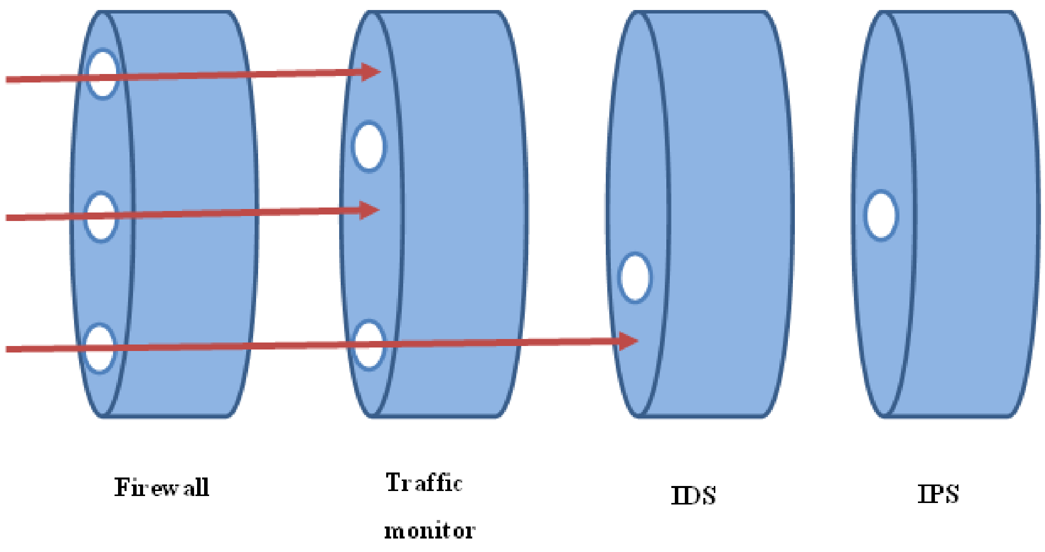


Figure 2.2: SCADA Communication With a Four-layers DiD Scheme. Adapted from (COMMISSION et al., 2018)

However, like any security technique, DiD comes with its own set of challenges, which are discussed in the following section.

2.2.1 Defense in Depth Security Problems

The Defense in Depth technique needs refinement to adapt to today's environments. According to Gerritz (2018), DiD is not enough because it leaves a large attack surface that allows attackers to go undetected, leaving companies/data/people/nations vulnerable. The author also says that this technique requires an advanced approach that must go beyond prevention layers.

Group et al. (2018) mentions that if an attacker can penetrate a particular vulnerability on one host, he is likely to be able to penetrate the same vulnerability on another host. Therefore, different layers cannot have the same vulnerabilities.

A critical consideration regarding security tools in general is their potential vulnerabilities that can undermine their proper functionality. For example, in 2019, a vulnerability was discovered in Iptables that could lead to a buffer overflow and even the execution of arbitrary code (CVE-2019-11360). Similarly, there are several Common Vulnerabilities and Exposures (CVEs) associated with Snort, such as CVE-2020-3299, which could allow an unauthenticated attacker to bypass HTTP policies configured in the tool. A scenario like the one in the figure 2.2 with DiD and the tools mentioned could be problematic.

In this context, if both tools have security issues such as these, they may operate inefficiently, leaving the system exposed with a significant attack surface. It is important to note that it took more than a year to update the vulnerabilities reported in these CVEs. Therefore, additional security methods within the security architecture would be required to prevent a successful attack even if the tools have vulnerabilities.

To mitigate these limitations, we propose the use of SFC along with DiD to provide a high level of security in ICS environments. Therefore, SFC-related topics are discussed in the following subsections.

2.3 Service Function Chaining and its components

Service Function Chaining (SFC) is built on two leading technologies, NFV and SDN. These two technologies, along with cloud computing and network virtualization, promise to reduce CAPEX and operational expenses, increase network flexibility and scalability, and accelerate time to market for new applications and services. (NGUYEN et al., 2017).

2.3.1 Software Defined Network

Software-Defined Network (SDN) is defined as a dynamic, manageable, cost-effective and adaptable architecture that redefines how networks are structured, managed and adapted, making it ideal for the high-bandwidth, dynamic nature of today's applications (NETWORKING, 2019). SDN embodies the idea of decoupling network control and forwarding functions, making network control programmable and abstracting the infrastructure for network services and applications.

At its core is a centralized controller with the authority to manage and orchestrate network behavior. By separating control from the physical infrastructure, administrators can dynamically shape traffic, optimize resource allocation, and respond to network events with unprecedented agility.

Controllers are entities that implement the control plane in the SDN model and can apply, modify, or remove entries in the flows table using the OpenFlow protocol (FOUNDATION, 2015). They operate in a manner analogous to an external server that has a global and centralized view of the network, which includes switches, machines, and all flows traveling in this scenario.

SDN introduces the concept of programmability, allowing network administrators to manage network devices, policies, and configurations through software interfaces. This programmable nature enables rapid adjustments to network behavior, streamlining service provisioning and facilitating rapid responses to changing traffic patterns or security threats. With SDN, what was once a complex and time-consuming process becomes streamlined and adaptive.

In a broader perspective, SDN is transforming network equipment and devices from closed, vendor-specific entities to open and versatile systems through its innovative technology. This shift enables the segregation of control and data planes, empowering networks to be programmed using open interfaces (NGUYEN et al., 2017).

The benefits of SDN are many. For example, it simplifies network management by providing a holistic view of the network and enabling consistent policy enforcement across the infrastructure. SDN improves network security by centralizing threat detection and enabling real-time response to potential breaches. SDN also optimizes network utilization and performance through dynamic traffic routing.

The flexibility of SDN also paves the way for innovations such as Network Function Virtualization, where network services traditionally implemented as dedicated hardware appliances can be abstracted and run as software instances on generic servers. This consolidation not only reduces costs, but also accelerates service provisioning and scaling.

2.3.2 Network Function Virtualization

Network Function Virtualization (NFV) can be seen as a complement to the SDN model (HAN et al., 2015), and the combination of NFV and SDN tends to increase performance, simplify compatibility between specialized technologies, and facilitate network maintenance procedures.

While NFV focuses on virtualizing network functions, SDN centralizes network control and enables dynamic network configuration. Together, NFV and SDN provide a flexible and agile network environment that is well suited for modern data centers, telecommunications networks, and cloud computing infrastructures.

The main motivation of NFV is to migrate the network functions implemented by dedicated hardware performing specific functions (e.g. firewall,

Table 2.1: Comparison between SDN and NFV Technologies.

SDN x NFV	**SDN**	**NFV**
Objective	Separation of control and data, centralization of control, and programmability of network	Relocation of network functions from dedicated appliances to generic servers
Target Location	Campus, data center/cloud	Service provider network
Target Devices	Commodity servers and switches	Commodity servers and switches
Formalization	Open Network Foundation (ONF)	ETSI NFV Working Group
Protocols	OpenFlow	None

proxy, gateway, load balancer, etc.) to generic devices (common x86 architectures). This feature allows virtualized network functions to be installed on any node in the network, making the scenario fully dynamic and responsive to the needs of the service consumer and extending the capabilities of the service provider.

In other words, network functions previously realized in expensive hardware platforms are now implemented as software appliances placed on low-cost commodity hardware or running in the cloud computing environment (NGUYEN et al., 2017).

A comparison between the concepts of SDN and NFV is shown in table 2.1. Note that the goals of both techniques are fundamental concepts of SFC, which is described in the next section.

Recently, the SFC technique has been presented in scientific works as a mechanism to improve the security level of applications in different computing environments. We assume that Defense in Depth can also be improved if it is built and deployed using SFC properties. Therefore, it is important to know the basic concepts of the technique as well as its components.

2.3.3 Service Function Chaining

Currently, provisioning services in computer networks requires manual configuration and instantiation of network functions, which requires human intervention and makes the network vulnerable to errors. When there are many

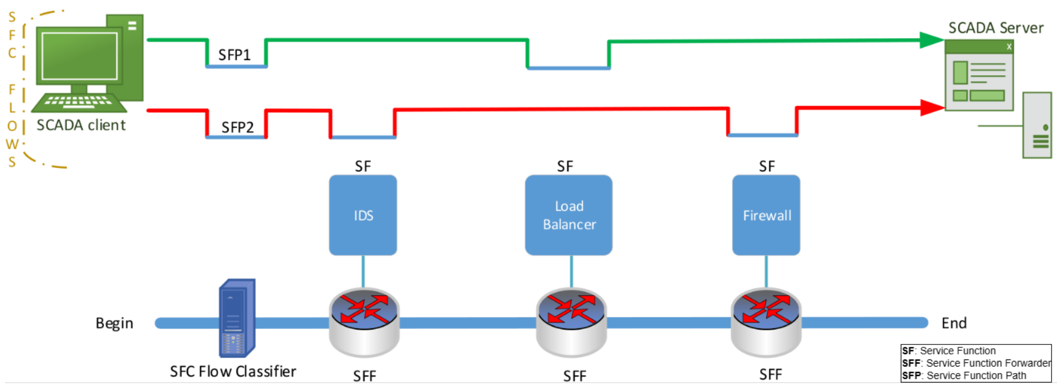


Figure 2.3: SFC Schema. Adapted from (CHEN et al., 2019).

devices to be configured in a given network, a large amount of work is required and the probability of error increases. Therefore, the SFC technique uses the best of SDN and NFV technologies to facilitate the work of configuring networks/devices through a central controller, as well as virtualizing services and machines. Some of the main components present in this solution are described below, according to RFC 7665 (HALPERN; PIGNATARO, 2015).

To take advantage of the dynamic flow change functionality, some of the key components must be present in the environment, such as Service Function (SF), SFC, Service Function Forwarder (SFF), Service Function Path (SFP), flow classifier, and Rendered Service Path (RSP). This scheme is shown in Figure 2.3.

The SF is a function responsible for performing a specific task on received packets. Such a function can operate in multiple layers of a protocol, such as the ISO/OSI layers. Some examples of SFs are firewalls, application acceleration, DPI, and load balancing.

A SFC is a structured sequence of abstract service functions and specific ordering constraints. These constraints define the precise order in which the functions must be applied to packets based on their classification. The first order of SFs must be satisfied because this technique provides the flexibility to change the order of the chain in real time.

SFF delivers the traffic to one or more service functions according to existing traffic forwarding policies. Another task that can be performed by SFFs is to deliver traffic to classifiers, if needed and supported, that transport traffic to another SFF.

The SFP is the specification of where the packets assigned to an SFP must go, i.e., after the classification, the SFC is instantiated by selecting instances of the constituent SFs, that result in a Service Function Path. In addition, the SFP supplies the notion of exactly which SFF/SFs the packet will visit when it traverses the network.

The Flow Classifier is the element that performs the matching traffic flows against policy for subsequent application of the service functions. For instance, in Figure 2.3, the SFC flow classifier forwarded the traffic throughout the IDS and Firewall by attack inspection in the path identified as SPF2.

The Rendered Service Path is related to the current sequence of SFFs and SFs visited by the packet until reaching the current component.

Additional SFC components can be used in specific situations. They are described in RFC 7665 (HALPERN; PIGNATARO, 2015). Another important aspect of NFV and SDN is that both technologies have been defined as key drivers in the design of 5G network architecture (ALLIANCE, 2015).

2.3.4 Advantages of Using SFC

It is especially important to highlight the choice we made for the SFC technique to increase the security level in industrial environments. Thus, the main contributions of using SFC are listed below:

- **Topological independence of the network**: SFC combines the best of SDN and NFV. Together, they provide a flexible, topology-agnostic environment.
- **Different paths with a unique topology**: Unlike DiD, SFC can use a unique topology and supply different paths for different applications. In this way, it is possible to have several levels of security by making use of the combination of Service Functions.
- **Specific protection layers**: Using SFC, it is possible to adjust intermediate security layers in a more specific way because it will not be seen by attackers, becoming harder for them to control the path (i.e., scale privileges). But we know that the attack surface can be larger, given there are larger network services infrastructures (YOON et al., 2017).
- **Machine Learning**: It is possible to use and integrate ML techniques into security solutions.

Another important technique used in this work is machine learning. A set of tools was used to evaluate the security level of the solutions and to generate the dataset and the testbed. Therefore, the primary concepts of machine learning are explained in the following section.

2.4 Machine Learning

The purpose of machine learning is to develop algorithms and models that enable computers to learn from data and make predictions or take actions without being explicitly programmed. The goal is to enable machines to automatically analyze and interpret patterns, extract meaningful insights, and make accurate decisions or predictions based on the available data (DOMINGOS, 2012). Machine learning aims to enhance the performance and capabilities of computer systems by enabling them to learn, adapt, and improve their performance over time through experience.

In the machine learning process, the availability and quality of the dataset play a crucial role. Particularly in supervised machine learning, the dataset consists of a collection of labeled examples or instances that are used to train and evaluate machine learning models. It serves as the foundation for the learning process, providing the necessary information for the algorithms to generalize patterns and make predictions (BISHOP; NASRABADI, 2006).

The dataset is typically divided into two main subsets: the training set and the test set. The training set is used to train the machine learning model by exposing it to labeled examples and allowing it to learn the underlying patterns and relationships. The test set, on the other hand, is used to evaluate the model's performance by assessing its ability to accurately predict the labels of unseen instances (AGGARWAL et al., 2015).

During the classification process, a ML algorithm employs extracted features from each dataset instance to create a predictive model or assign class labels. This involves applying statistical techniques and mathematical algorithms to analyze data patterns and relationships. Consequently, the model, which represents the learned knowledge, can classify new, unseen instances based on the learned knowledge (AGGARWAL et al., 2015).

The classification process involves different steps, such as feature selection or extraction, model training, and prediction (DOMINGOS, 2012). Feature selection aims to identify the most relevant and informative features from the dataset that contribute to the classification task. Model training involves feeding the selected features and their corresponding labels into the machine learning algorithm, allowing it to adjust its internal parameters and optimize its performance. Once the model is trained, it can be used to predict the labels of new instances based on their extracted features, providing valuable insights and facilitating decision-making processes.

By leveraging the power of datasets and the classification process, machine learning enables computers to automatically analyze complex data, detect patterns, and make accurate predictions or classifications, thereby enhancing the capabilities and performance of computer systems in a wide range of applications.

2.4.1 Machine Learning Classifiers

In this study, we employed a set of ML classifiers, namely Neural Network, Decision Tree (DT), Random Forest (RF), Gradient Boosting (GB) and Gaussian Naive Bayes (GNB) to analyze and classify the dataset. These classifiers are widely recognized and extensively used in the field of machine learning for their ability to handle complex datasets and make accurate predictions.

The Neural Network classifier is particularly noteworthy. It is inspired by the structure and functioning of the human brain and consists of interconnected nodes organized into layers. These networks excel at capturing intricate patterns and dependencies in data, making them well-suited for tasks requiring deep learning and understanding of complex relationships (HASTIE et al., 2009).

The Decision Tree classifier is a popular algorithm that builds a hierarchical tree-like structure to represent decisions based on feature values. It offers interpretability and transparency in decision-making processes, making it suitable for understanding the underlying patterns and relationships within the dataset (HASTIE et al., 2009).

The Random Forest classifier, on the other hand, leverages an ensemble of decision trees to make predictions. By combining multiple decision trees, it reduces the risk of overfitting and improves the overall accuracy and robustness of the classification model. The Random Forest algorithm is known for its ability to handle high-dimensional data and capture complex interactions between features (AGGARWAL et al., 2015).

Another powerful classifier utilized in this study is Gradient Boosting. It operates by sequentially adding weak learners to form a strong predictive model. Gradient Boosting is particularly effective in handling large datasets and has demonstrated superior performance in various machine learning tasks. It excels in capturing intricate patterns and generating accurate predictions by minimizing errors iteratively (HASTIE et al., 2009).

The GNB classifier is a probabilistic algorithm that assumes feature independence and works well with continuous input features following a Gaussian distribution. It calculates the likelihood of data points belonging to each class and uses Bayes' theorem to determine the posterior probability of each class. The GNB classifier is computationally efficient and has found applications in text categorization and medical diagnosis tasks (HASTIE et al., 2009).

By incorporating these well-established machine learning classifiers into our analysis, we aim to leverage their strengths and exploit their unique characteristics to achieve accurate and reliable classification results. The combination of Decision Tree, Random Forest, Gaussian NB and Gradient Boosting classifiers provides a diverse and comprehensive approach to effectively handle the complexities and nuances of our dataset, enabling us to make informed decisions and draw meaningful insights from the data.

2.4.2 Late-Fusion Classification

Late-fusion classification is a technique in machine learning where the outputs or decisions from multiple classifiers are combined at a later stage to make a final prediction or classification. It involves leveraging the results from different classifiers to enhance the accuracy and reliability of the classification process. Instead of relying solely on one classifier, late fusion classification takes advantage of the diverse perspectives and strengths of multiple classifiers (GUNES; PICCARDI, 2005).

The outputs from these classifiers are merged or integrated to produce a consolidated prediction. This approach helps to mitigate individual classifier limitations and biases, leading to improved overall performance in classification tasks. Late fusion classification has found applications in various domains, including computer vision, natural language processing, and anomaly detection, where it has shown effectiveness in enhancing classification accuracy and handling complex data patterns.

In the landscape of cybersecurity, where the power of machine learning unfolds, providing a dynamic shield against evolving threats, another crucial dimension is the strategic use of honeypots.

2.4.3 Honeypot

Low interaction honeypots are cybersecurity decoys designed to mimic vulnerable systems or services with minimal interaction capabilities (BRINGER; CHELMECKI; FUJINOKI, 2012). Unlike high-interaction honeypots that emulate real systems and engage more deeply with potential attackers, low-interaction honeypots limit the risk by providing only a surface-level simulation.

Essentially, low-interaction honeypots emulate specific vulnerabilities or services without actually running the full application or operating system. They often respond to known attack patterns or behaviors, providing a controlled environment for monitoring and investigating malicious activity. These honeypots are typically less resource-intensive, reducing the likelihood of compromising the security of the entire network (BRINGER; CHELMECKI; FUJINOKI, 2012).

One of the advantages of low-interaction honeypots is their ease of deployment and maintenance. Their limited functionality and reduced complexity make them more straightforward to set up and manage compared to their high-interaction counterparts. While they might not provide the depth of interaction seen in high-interaction honeypots, they still serve as valuable tools for early detection, data collection, and analysis of common attack techniques without exposing the network to significant risks.

For ICS environments, versatile, low-interaction server-side honeypots are available. It offers various industrial control protocols, providing the basis for custom systems that emulate complex infrastructures and create a convincing environment for potential adversaries. Custom HMI and artificially delayed service responses enhance its deceptive capabilities. It can also support access with production HMIs and can incorporate real hardware, making it a valuable tool that benefits from collaboration with notable contributors in the field (CONPOT, 2023).

2.5 Chapter Discussion

This chapter has provided a comprehensive theoretical foundation essential to understanding the context of this research. It encompassed essential concepts including Industrial Control Systems, the Defense in Depth approach, SDN, NFV, SFC, and its core components. In addition, a brief view of the field of Machine Learning was also presented. The next chapter presents the works related to this thesis.

3 Related Works

This chapter discusses the main related works on two topics: (i) the use of SFCs to enhance security in systems, applications, and environments, and (ii) challenges in improving the security level of SFCs. It also covers (iii) existing IDSs developed for SCADA environments and (iv) the use of machine learning techniques to enhance IDS security.

3.1 SFC as a Framework to Solve Security Problems

A common issue in the field of security is the Distributed Denial of Service (DDoS) attack, which aims to render services or systems unavailable. Therefore, it is crucial to find effective solutions to mitigate such attacks. Four papers were found that address this problem and utilize the SFC to either detect or mitigate it.

Bauer, Heseding & Flittner (2017) propose a new architecture to detect and mitigate DDoS. In addition, the authors used the concepts of SDN, NFV, SFC, and Python algorithms to implement the architecture in a tool they named EarlyDrop. It is a new defense mechanism against DDoS attacks where the network operator can adjust its parameters according to the cost of attacks and the number of resources required for mitigation.

It implemented a prototype of the EarlyDrop framework, which was tested under a simulated environment (using Mininet network), Ryu SDN, LXC (Containers), Python scripts SFC, and Suricata IDS). The results show that the proposed architecture and framework can reduce the mitigation costs (when compared to other approaches) and provide the network operator with control over the number of resources used for the mitigation task.

Similar to the previous work, Shameli-Sendi et al. (2016) proposed a new architecture and a framework named SDPAP based on distributed security policies. These policies are considered for the placement (which the authors call optimal placement) of security functions that let the nodes less overloaded, thus the entire system becomes more resilient to attacks like DDoS.

In addition, tests have been conducted in a simulated environment of up to 69,000 nodes, demonstrating acceptable performance and providing a better and more tailored level of security for data center applications.

The work of Bondan et al. (2017) aims to improve the mitigation of potential attacks related to time consumption, resources, or other factors by utilizing collaboration among multiple domains. To achieve this, the authors proposed a framework that employs SDN and SFC concepts to improve collaboration among multiple domains. The framework manages and allocates resources for collaboration.

Collaboration is highly relevant in mitigating DDoS attacks as it requires a considerable amount of resources. By creating a distributed mitigation environment, other high-scale attacks can also be mitigated. Preliminary tests were conducted in a real environment, demonstrating the viability of this approach for high-scale attacks.

Another work related to DDoS was done by Alqahtani & Gamble (2015) and its purpose was to detect this kind of attack. This approach uses four detection layers: service-level detection, tenant-level detection, application-level detection, and cloud-level detection. Each layer has different techniques for the detection of DDoS attacks, in which the first layer sends the result to the second layer; the second layer sends the result to the third layer, and so on. Finally, the last layer is responsible for analyzing all results to infer what services are under attack and what services are applying the attack. Therefore, each application has a service chain to detect possible DDoS attacks. The authors report successful detection of DDoS attacks, which presents new possibilities for mitigation.

One of the initial steps taken by attackers is to map target ports. This can be accomplished through various methods, including TCP scanning, NULL, TCP connects, SYN, FYN, XMAS, and ACK. While many of these techniques can be easily detected by security solutions, attackers often employ more advanced techniques such as distributed port scanning and slow port scanning, which require greater sophistication to detect.

Sanz et al. (2017) proposed an architecture and a prototype capable of detecting various types of port scanning, including distributed (horizontal or vertical) or slow scans. According to the authors, the prototype was able to detect all types of port scanning with high accuracy. However, it generated a high number of false positives in distributed port scanning.

Hu & Yin (2017) discusses the importance of configuring security levels based on the specificities of each application. Different applications require different security levels. The authors proposed a framework that utilizes the SFC technique to provide dynamically customized security and adaptive services for current applications. The framework was developed using a base application for managing special vehicles, including ambulances, fire department cars, and others. The analyzed security features were related to security policies.

The authors claim that their framework can enhance vehicle management and application security. However, they acknowledge that the new components added to the framework may introduce security challenges in the SFC environment and require further analysis.

The main purpose of the work of DePhillips, Katramatos & Bhattacharyya (2017) was to develop a mechanism for analyzing packet content (on-the-wire analysis) to enhance security in future high-bandwidth networks. The authors utilized a SFC environment for communication between two hosts exchanging messages. Subsequently, they conducted simple tests to analyze the payload of the packets and identify potentially malicious flows. The task was successfully completed. The authors report a significant finding regarding the low cost of wire analysis. However, they acknowledge the need for further testing to assess scalability.

3.2 SFC in Specific environments

Some works have utilized SFC for specific environments, including industrial, scientific, and federated networks (FYSARAKIS et al., 2017), (PETROULAKIS et al., 2018), (ANANTHA; RAMAMURTHY, 2017).

The works of Fysarakis et al. (2017) and Petroulakis et al. (2018) were developed by the same research group and focused on industrial networks in a real wind park.

The main idea was to analyze network traffic and classify it as either legitimate or malicious. To achieve this, we developed an architecture and a reactive security framework. The framework continuously monitors network traffic and performs detailed analysis to detect possible attacks.

The analysis is performed through the following security functions: Firewall, IDS, and DPI, which are chained using SFC. Once the traffic is classified as legitimate or malicious, legitimate traffic is directed to its destination in the network after bypassing additional security functions. Malicious traffic, on the other hand, is sent to a honeynet, which is a network designed to analyze the behavior of attackers.

A proof of concept was conducted within an operational wind park to evaluate the framework's performance and overhead. The results suggest that the framework is suitable for operational wind parks, as the additional delay for critical services was acceptable.

Anantha & Ramamurthy (2017) proposed a new framework called ScienceSDS, which aims to facilitate the use and chaining of complex security functions for scientific data sent in SDN environments. These data are susceptible to security analysis, which can selectively filter and route them according to monitoring and inspection rules.

The authors claim that their framework provides a secure way to transfer data between endpoints and dynamically compose services. They implemented the framework and evaluated its performance, measuring the time required to configure a Service Function Chain (SFC) and steer traffic through SFCs of varying sizes.

The results demonstrate that the framework generates minimal overhead, allows for flexible security service composition using appropriate functions, and is adaptable to changes in traffic and security attacks.

Some papers use SFC to enhance security in federated environments, as described in this subsection.

Massonet et al. (2016) proposed to improve security in federated network environments using NFV and SFC. These techniques evaluate communication among different networks to determine which are trusted and which are untrusted. If an untrusted network is detected, all communication with that network must be encrypted. The authors utilized the Secure Socket Layer (SSL) tool and its public key algorithms for encryption.

The approach relies on a service manifest that specifies the global network security policy. Through this manifest, the configuration of the security functions for different clouds of the federation is generated. Tests of the approach were performed in a simple environment (with five virtual machines) successfully, however as future works the authors intend to perform tests in more complex environments.

Another paper published by the same authors (MASSONET et al., 2018) complements the previous work with the use of global security policies to automate the deployment and configuration of network security functions through different cloud federation networks. This is the major contribution of the work because performing this task manually can take a long time and is error-prone.

The same case study of the previous work was performed using the global security policies for encrypting the communication among clouds (there are three clouds: two of them trusted and one untrusted) when the destination cloud is untrusted, otherwise, the encryption is not necessary.

The authors did not consider the possible existence of local policies (it could conflict) and local service chains (compatibility) to simplify the environment. Thus, this work presents a high-level approach to security customization in federated networks.

3.3 Proposed architectures

This section describes two works proposing theoretical solutions, such as architectures.

As discussed earlier, Chou et al. (2016) discusses the importance of SDN and NFV concepts in the implementation of new techniques, such as SFC. As Network security management and information risk control bring challenges to the existing networks, authors propose new security service on-demand architecture to allow service providers to offer their clients flexible and secure dynamic services over an SDN network.

The main contribution of the work is to enable service functions to provide diverse security services for different users, also improve some processes of manual installations, build rapidly and flexibly security services, and optimize the use of resources. However, the contribution is limited to the proposed architecture, i.e., no tests were performed either in simulated or real environments.

Zhou et al. (2018) presents a comprehensive analysis of the SCADA system architecture is proposed and delineates its fundamental characteristics, and constructs an attack model. Following the basic DiD model, a refined five-layer defense architecture is introduced and subjected to simulation with multiple attack scenarios. Empirical results show that the proposed DiD model significantly increases the complexity of attacks and robustly defeats threats, especially those related to system infiltration and data tampering. This security is important, but lacking integration with others is concerning.

3.4 IDS Security Improvements

Some works were developed to improve the security of existing IDSs, such as Snort, Suricata, and Bro, to be used in SCADA environments. Also, new IDS approaches were proposed.

Yang, Cheng & Chuah (2019) proposed a network intrusion detection system making use of machine learning and deep learning. This solution aims to protect SCADA networks in ICS environments against SCADA-specific attacks, as well as more general attacks. Convolutional Neural Networks were used to find attack patterns in SCADA traffic and identify time windows where may be an attack. Presented results show that the approach used reached a high detection accuracy and provide the capability to handle emerging threats..

Lai, Zhang & Liu (2019) proposed a solution for the detection of traffic anomalies in ICS, as well as the classification of attacks. For this purpose, Convolutional Neural Networks Convolutional Neural Networks (CNNs) are used to represent the detection model. According to the authors, the method can automatically extract characteristics considered critical and provide a precise classification. For the evaluation of the presented model, real attack data to a SCADA system were used. The results presented a method related to attack detection and classification in a SCADA environment. A limitation of the solution is that new attacks or variations are not detected by the model.

Another work that uses the idea of anomaly detection using CNNs is presented by Zhang et al. (2019). In this work, the authors collected and analyzed network traffic from control systems in power generation and substation environments. This way, they proposed a new specific model which can detect a possible attack in power environments. Results showed that the model is effective and has high accuracy when compared with other CNN works.

Radoglou-Grammatikis et al. (2020) proposed a specific IDS for the Modbus/TCP protocol to detect possible DoS attacks on the environment. For this purpose, there are two main modules, the first is responsible for capturing traffic on subnets, and the second is to analyze whether such traffic can be related to DoS attacks. Machine learning techniques were used to detect attacks, with an accuracy of 81%. Their model was able to significantly improve the system's accuracy when compared to traditional techniques. However, the protocol-specific nature of the scheme may lead to a lack of generalization, thus, unreliability to be used in production environments.

Similarly, Alem et al. (2023) proposed a more specific intrusion detection scheme for ICS aiming PLC intrusion detection. The authors used a neural network that evaluates PLC-related messages. Although the authors assessed the quality of their scheme in a real industrial environment, the applicability to other industrial assets was not evaluated.

Mieden & Beltman (2020) developed an IDS to analyze the traffic of a water treatment plant that also uses machine learning to detect different types of attacks. The model with deep neural networks developed was able to identify malicious behavior within the industrial network with a high success rate, however, configurations and adaptations are necessary for use in industrial environments, since the time of traffic increases.

Rajesh & Satyanarayana (2021) focuses on enhancing the security of SCADA networks, against cyber-attacks through ML techniques. The authors create a new dataset by capturing real-time SCADA test bed traffic containing both normal and attack data. Four machine learning algorithms are assessed using various performance metrics. The study compares the performance of these algorithms under different scenarios, achieving a ROC value of 99.96%. Such a IDS could be integrated into the architecture proposed in this thesis.

In general, the approaches proposed in the literature prioritize efforts to enhance the system's accuracy. For instance, Ahakonye et al. (2023) proposed a feature selection technique for ML-based NIDS in SCADA systems. The authors improve their false-positive rates when proactively selecting their model features in outdated NIDS datasets. Unfortunately, the applicability of their proposed model in real-world ICS is overlooked.

Similarly, Ouyang et al. (2021) proposed NIDS implemented using a few-shot learning scheme for SCADA systems. Their proposed scheme improved detection accuracy compared to other approaches on a SCADA-related dataset. The impact of motivated attackers on circumventing their proposed model is not evaluated.

3.5 Chapter Discussion

The works presented and discussed in this chapter focus on improving the security level of SFC architectures, proposing new security solutions in various environments, and the new IDS implementation that makes use of Machine Learning and Deep Learning techniques to improve results. However, among all the papers presented, none of them presents a solution for SCADA environments that performs the dynamic classification of flows in an integral way, i.e., a solution aimed at preventing privilege escalation, thereby preventing attackers from gaining control over the layers and maintaining the integrity of the DiD, as vulnerability exploitation often involves privilege escalation techniques.

In this thesis, we propose an architecture based on Defense in Depth with different security layers. It is important to emphasize that each of the layers has one or more security tools, as well as different Operating System (OS), aiming for diversity as a whole in the ICS environment.

Some of the works presented in this chapter, such as IDSs or specific security tools for SCADA and ICS environments, possibly can be integrated into our testbed as a specific security layer. Therefore our proposal, which consists of different security layers, can be an interesting starting point when thinking about improving the security level in Industrial Environments, and subsequently, selecting the best tools for each specific environment. Finally, Table 3.1 highlights the most important characteristics of each related work presented in this chapter.

Table 3.1: The main characteristics of related works.

Author	Focus on ICS	Reactive architecture	Defense in Depth	SFC to improve security	ML-based IDS	ICS Anomaly detector	OS diversity	ICS Functional prototype
(BAUER; HESEDING; FLITTNER, 2017)		✓		✓				
(SHAMELI-SENDI et al., 2016)		✓		✓				
(BONDAN et al., 2017)				✓				
(ALQAHTANI; GAMBLE, 2015)		✓		✓				
(SANZ et al., 2017)		✓		✓				
(HU; YIN, 2017)				✓				
(DEPHILLIPS; KATRAMATOS; BHATTACHARYYA, 2017)				✓				
(FYSARAKIS et al., 2017)	✓	✓	✓	✓				✓
(PETROULAKIS et al., 2018)	✓	✓	✓	✓				✓
(ANANTHA; RAMAMURTHY, 2017)				✓				
(MASSONET et al., 2016)				✓				
(MASSONET et al., 2018)				✓				
(CHOU et al., 2016)		✓		✓				
(ZHOU et al., 2018)	✓		✓					
(YANG; CHENG; CHUAH, 2019)	✓				✓	✓		
(LAI; ZHANG; LIU, 2019)	✓				✓	✓		
(ZHANG et al., 2019)	✓				✓	✓		
(RADOGLOU-GRAMMATIKIS et al., 2020)	✓				✓			
(ALEM et al., 2023)	✓				✓	✓		
(MIEDEN; BELTMAN, 2020)	✓				✓	✓		
(AHAKONYE et al., 2023)	✓				✓	✓		
(OUYANG et al., 2021)	✓				✓	✓		
(RAJESH; SATYANARAYANA, 2021)	✓				✓	✓		

4 ICS Integral Defense in Depth Architecture

ICS/SCADA systems are often targeted by cyber-attacks and are critical systems due to the potential for irreparable losses resulting from an attack or malfunction. These systems also store and transmit sensitive data for companies and their customers, making them attractive targets for attackers seeking to profit illegally from potential vulnerabilities. As a result, ensuring the security of these systems is of utmost importance.

This work presents a Machine Learning-Based Integral Defense in Depth Security solution for ICS, which is capable of detecting possible attacks on SCADA systems and performing mitigation actions dynamically and immediately, to ensure that attackers do not even know that they are dealing with a set of security tools embedded within the environment in question.

Since it is not possible to state that a system is 100% secure, regardless of its nature, and industrial systems have become increasingly attractive targets to attackers, it is important to note that there are increasingly effective security mechanisms. Therefore, this thesis proposes an architecture and a security solution based on Defense in Depth, SFC and Machine Learning to mitigate possible attacks aimed at exploiting existing vulnerabilities in ICS environments.

In Section 2.2, we discussed the potential susceptibility of DiD environment to manipulation by attackers aiming to exploit vulnerabilities within the security tools operating at each layer. For example, attackers could gain control of one layer and use it as a starting point to elevate their privileges in subsequent layers, allowing them to advance through the defense layers.

One of the main goals and defining features of this thesis pertains to the concept of "Integral" (within the context of the Integral DiD Security Model). This term emphasizes the solution proposed in this work, which hides the layers of defense. This design prevents attackers from gaining control or exploiting potential vulnerabilities within the secure environment. Even if malicious activity is detected, the attacker's data stream is redirected to a controlled environment, such as a honeypot.

The implementation of an ICS environment with Integral Security DiD mechanism has been achieved through the use of SFC. This technique provides greater control over the flow of data in the environment, allowing for the determination of which security tools are used to analyze traffic when a flow passes through the environment (SFP). The path for the flow can be predefined or dynamically changed based on the analysis performed.

For instance, when network traffic can be directed towards various hosts, each necessitating different security levels, it becomes possible to configure a path that traverses more security layers for specific hosts compared to others. This adaptability allows us to tailor the security level to the specific requirements of different domains.

The capability to dynamically adjust traffic flows is enabled by a machine learning model integrated into the SFC flow classifier developed in this study.

4.1 Architecture Overview

In this chapter, a qualitative approach to intrusion detection and prevention is presented in an architecture that reinforces its robustness through the strategic implementation of DiD and SFC. Visually represented in Figure 4.2, the architecture encompasses different security layers that can be customized according to the environment. At the core of this structure, the SFC Flow Classifier stands out a component that utilizes ML-based heuristics for flow forwarding in the SFC flow-classifier to assess traffic and subsequently route it to the security layer best prepared to discover the nature of the traffic. This dynamic approach empowers the architecture to adapt to different types of threats, providing a precise response to potential attacks.

To complement this approach, an additional component has been inserted

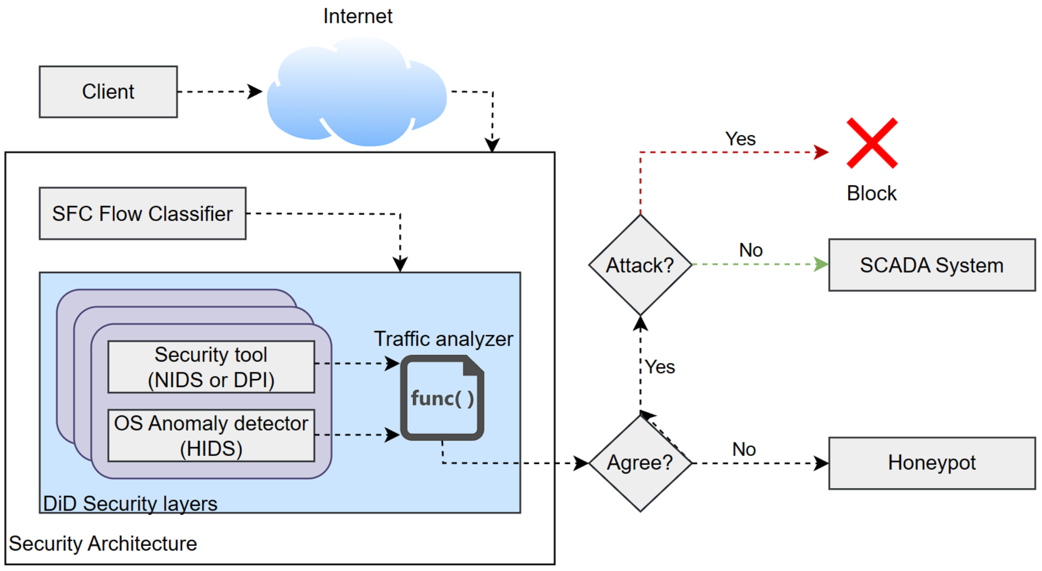


Figure 4.1: Architecture Overview.

within each layer of DiD through the quantitative approach. This component focuses on identifying anomalies in the Operating System parameters. Thus, it is possible to detect attacks and attempts to escalate DiD, as attackers often target the most vulnerable components of the architecture.

In the outlined security architecture, traffic enters in the ICS environment, and to reach the SCADA system, this traffic is directed through the proposed architecture. The first element of this architecture is the SFC Flow Classifier, responsible for directing traffic to three possible security layers. These include a combination of NIDS + HIDS or DPI + HIDS.

Within these layers, a thorough analysis of traffic is performed to discern malicious activity. Subsequently, the responses of these two tools are evaluated. If both agree that the traffic is considered an attack, the system immediately blocks this traffic, preventing any further compromise. In the case of a consensus on the normality of the traffic, it is forwarded to the SCADA system. However, if there is disagreement between the tools, the traffic is redirected to a specific honeypot.

To address new threats, suspicious traffic is directed to a specialized honeypot as previously mentioned. In this scenario, each incident is meticulously recorded, categorized, and analyzed. This information is then used for continuous improvement of the detection model, enhancing the effectiveness of defense against future attacks. This approach not only increases the system's

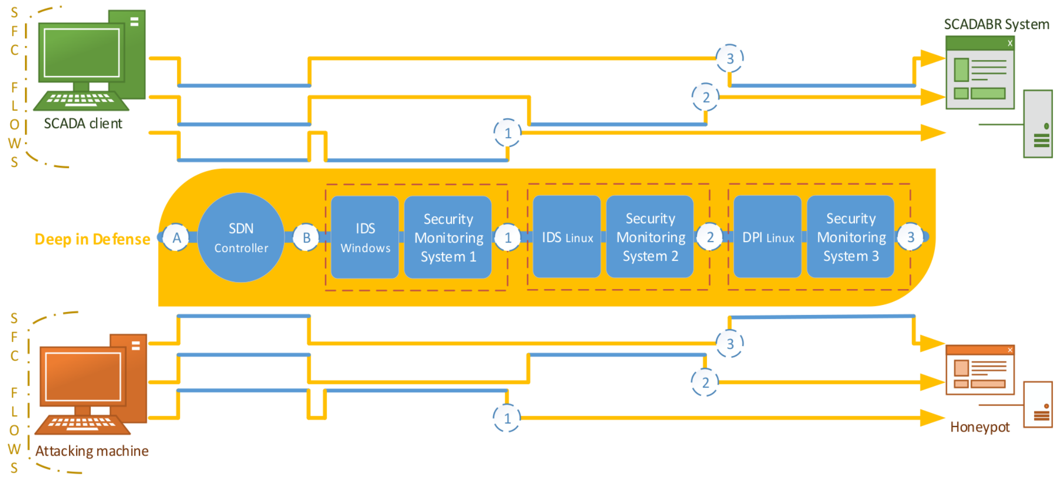


Figure 4.2: Security Solution Overview.

resilience to emerging threats but also contributes to the ongoing evolution of cybersecurity, ensuring an agile and adaptive response.

4.2 Machine Learning-Based Defense in Depth to Provide Security in ICS Systems

The following paragraphs present the architecture developed and its general idea, as the main goal of this work is to enhance the security level of ICS environments.

Thus, the security enhancement of this thesis is compounded by an SFC-based DiD implementation with an intelligent SFC flow classifier. In each DiD layer (represented as 1, 2, and n in the figure), there is a Security Tool and a Security Monitoring System, as shown in Figure 4.2. The SFC Flow Classifier and the Security Monitoring System components are described in the following sections.

It is crucial to acknowledge that this is a general architecture, and customization may be necessary for each specific environment.

We assume that normal and attack traffic is routed through the DiD based on SFC, that we have shown a split in Figure 4.2 as SFC flows for ease of understanding. In SFC flows, the blue/dark line (identified as 1, 2, and n) means that the traffic is sent from the SFC classifier located between A and B to a DiD layer, i.e., the SFP 1, for example, goes through the security tool 1 and

SMS 1 and then to the SCADA System or Honeypot.

As mentioned in the section 2.2, each environment utilizing this method must have a tailored level of security. Consequently, a variety of tools may be included in a particular layer of defense, depending on their intended use.

For instance, the Security tool can be a conventional Firewall, Intrusion Detection System, Deep Packet Inspection, etc. The SMS is an architectural system, specific to our proposal and after evaluation, the traffic is sent to the SCADA server (if the traffic is classified by the SMS as normal) or to honeypot (if the traffic is classified by the SMS as an attack). Of course, the SFPs represent various possibilities, but generally, only one of the paths will take place.

The network security mechanism comprises a conventional firewall, IDS, or DPI. The SMS is an architectural design specific to this work. After evaluation, traffic is directed to the SCADA system if it is deemed normal, or to the honeypot if it is not normal or an attack. The SFC flows represent different possibilities, but generally, only one will occur.

4.2.1 SFC Flow Classifier

The SFC flow classifier is one of the most important elements of this work, as it is responsible for dynamically selecting the best path to a packet. This is done by performing a similarity analysis of previous flows that have been labeled as attack traffic or normal traffic (i.e., coming from a real SCADA client).

The similarity analysis predicts the probabilities of the input features belonging to each category. Rather than returning a discrete class, the method returns the probabilities associated with each class. For example, a packet has a 90% chance of being related to the attack class. Based on this, the SFC flow Classifier determines the best security tool to evaluate the flow to reach a possible conclusion (i.e., the most specialized tool to analyze it), whether it is actually an attack or not.

After reviewing the selected security tool, there may still be uncertainty about the type of flow. Therefore, the SFC Flow Classifier must repeat the same steps to resolve the issue. This event occurs until the packet is forwarded to the SCADA server (if it is classified as normal traffic), or to the honeypot (if it is classified as an attack, or if it cannot be determined that it is normal traffic).

The SFC flow classifier brings to this solution very important features towards the goal of mitigating the layers attack surface (YOON et al., 2017), because it

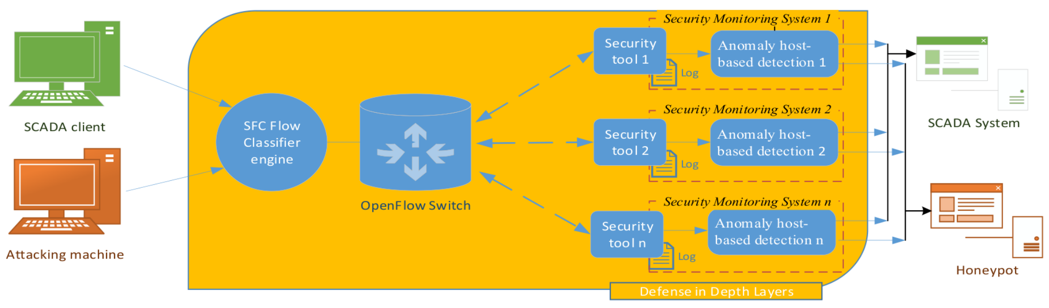

SCADA client
Attacking machine
SFC Flow Classifier engine
OpenFlow Switch
Security Monitoring System 1
Security tool 1
Log
Anomaly host-based detection 1
Security Monitoring System 2
Security tool 2
Anomaly host-based detection 2
Security Monitoring System n
Security tool n
Anomaly host-based detection n
Defense in Depth Layers
SCADA System
Honeypot

Figure 4.3: Security Monitoring System Details.

does not allow an attacker to explore a vulnerability in one layer, control it and after attacking another layer in DiD, i.e., beyond the security enhancement imposed by the DiD, a motivated attacker cannot scale up the DiD arrangement and reach the SCADA system, taking control of each layer individually. This is particularly important in ICS systems, as IEC 62443 recommends the use of DiD in these systems.

Therefore, this work employs an intelligent SFC flow classifier that uses a ML-based engine and selects the best security tool to analyze an attacker's content, since a dynamic classifier selection algorithm is applied to route a received content to a layer, thus the attacker loses control over which layer he will face in the DiD.

Consequently, the attacker has no control over the number of layers of defense in the environment or the next layer that will be accessed if they manage to bypass one. This is because, regardless of the arrangement, each time content reaches the SFC flow classifier engine, it can be routed to another security tool without the attacker's knowledge. Figure 4.3 illustrates this scenario.

Even considering the worst case, where the content is similar to the content previously seen by the SFC flow classifier engine, and the content travels through the same flow to reach the security tool, our proposal adds an anomaly host-based detection engine (i.e., SMS in Figure 4.3), which will reduce the impact of an attack, as we will explain in the next section.

4.2.2 Security Monitoring System

This module is composed of two components, a **Security tools**, and **Anomaly host-based detection**.

- **Security Tools**: This module identifies market tools and mechanisms designed to detect known attacks reported in Common Vulnerabilities and

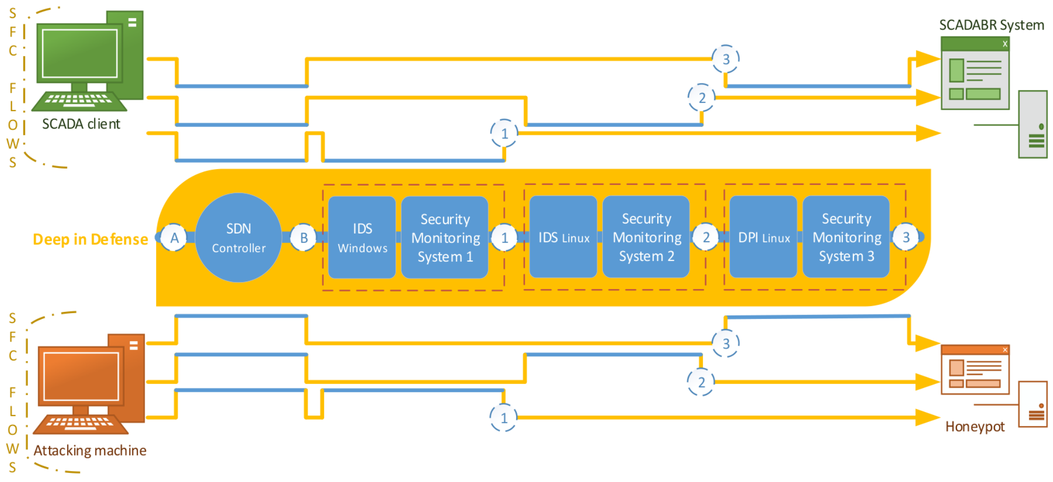


Figure 4.4: Scenario composed of security tools and anomaly host-based detection.

Exposures (CVE). These tools are commonly used for offensive and defensive strategies by red, blue, and purple teams. The SMS monitors the log for each content forwarded to this security tool and alerts if a breach or attack is detected.

- **Anomaly host-based detection**: This module aims to monitor attacks using resources available in the operating system. The rationale behind this is that if a security tool is attacked, it will cause noticeable changes in processes and system parameters that the ML-based model can detect at the host level. Thus, assuming the worst case scenario that a security tool can be individually controlled, in this additional module it will be noticed by a SMS administrator, but will not be considered as a resource to be bypassed by an attacker. In this case, we use OS diversity to catch a possible vulnerability in the same tool but on a different platform (GARCIA et al., 2014), (BULLE et al., 2020).

To evaluate the combination of the two modules mentioned above, a scenario was created as shown in Figure 4.4. This architecture allows for modifications, such as adjusting the number of layers and tools, to ensure adaptability to various types of ICS environments while maintaining an appropriate level of security. The tools and implementation details are presented in the following section.

4.3 Prototype and Result Analysis

This section presents the results evaluating the use and feasibility of the tools used in the development process to build the proposed solution. It also presents the tools, their specifications, specifications and libraries used in the testbed implementation. Two evaluations are conducted, one for performance and one for security.

The testbed created in this work is a client/server architecture using three security layers (a number that can be changed at any time, according to the required security level) and theSFC flow classifier implemented through the SDN controller.

Several tests were performed to verify that the selected tools and components could improve the security level of the ICS environment. The entire environment was built on a machine running Ubuntu Linux OS. All the tools listed below were installed and configured.

4.3.1 VirtualBox (Host machine)

It is a widely known and open source cross-platform virtualisation software (VIRTUALBOX, 2023). Our environment consisted of 9 virtual machines, each of them with different functions (Linux attack, Windows attack, normal traffic, defense layer 1, defense layer 2, defense layer 3, ScadaBR, ModbusPal, and Conpot honeypot). This set of virtual machines was created and managed using Virtualbox software on a Linux machine.

4.3.2 Linux Attack Machine

The Linux attack machine consisted of a set of tools responsible for part of the Penetration Testings (Pentests) performed. Such a machine used the standard Kali Linux OS installation (LINUX, 2023).

Kali Linux

Kali Linux is an open source Debian-based Linux distribution designed for various information security tasks such as Pentest, security research, computer forensics and reverse engineering (LINUX, 2023).

The attacking machine was updated with manually installed Pentest tools and is presented below.

Nessus

This is a well-known vulnerability analysis tool used for pentests. In this work, the essential version was used.

Nessus (NESSUS, 2023) was used to scan for possible vulnerabilities in ScadaBR (presented later), but it was necessary to create a specific attack profile to characterise it as an advanced vulnerability scanner. This profile was created by a professional Pentest analyst.

Nmap

Nmap (Network Mapper) is another tool used for penetration testing (NMAP, 2023). It has several purposes, but in this paper it was only used for port scanning.

Smod

Smod is a tool composed of several modules with offensive functions to test for possible vulnerabilities specific to the Modbus protocol (widely used in ICS/SCADA environments) (SMOD, 2023).

As the smod is an open source tool, it was possible to make some adaptations necessary for the automation of the Pentests, such as the continuous execution of Modbus attacks, which in their normal version require several manual requests.

4.3.3 Windows Attack Machine

This machine had the same role as the previous one, but with different attack tools.

Acunetix

The other attack machine running Windows was updated with security tools to run Pentest on the testbed environment. Accunetix (ACUNETIX, 2023) was one of the tools manually installed on such a machine.

The Acunetix is a complete web application security testing solution that can be used both stand-alone and as part of complex environments (ACUNETIX, 2023). This work was only used as a web vulnerability scanner, focusing on the SCADA web server.

This tool also required the creation of a specific attack profile. This profile was also developed by a professional Pentest analyst.

Arachni

Arachni is an open source web application security scanner framework (ARACHNI, 2023). It was also used to test the SCADA web server. Like Acunetix, the Arachni tool required some customization for the Cross-Site Script (XSS) and Structured Query Language (SQL) injection attacks.

4.3.4 SCADA Client

This is the machine responsible for sending normal traffic to the outside world. For this purpose, a standard Ubuntu Linux was used as the OS. TcpReplay software was also added (REPLAY, 2023), which is briefly introduced below.

4.3.5 Security Layer 1 (SMS 1)

This machine also used the standard Ubuntu Linux OS and two security tools added later, Snort IDS (SNORT, 2023) and SysStat (GODARD, 2015).

Snort

Snort is open source software that can work like IPS or IDS, i.e. to prevent and/or detect attacks. It is capable of real-time traffic analysis as well as logging according to preconfigured rulesets.

In this work, Snort was configured using the community ruleset available from the official Snort IDS website. In addition, Modbus protocol-related settings were made, such as enabling the Modbus preprocessor.

As mentioned earlier, Snort can log possible attacks related to the configured ruleset. This is a very important aspect of this tool in this work, as any packet detected as a possible attack is logged for subsequent ML model generation.

SysStat

Sysstat aims to monitor and log various activities that occur within the environment. Such a tool is important because, in the event of an attack, there may be anomalies in the activity of the OS. Thus, SysStat is not a security focused tool, its usual function is to evaluate performance parameters. However, in an attack scenario, these parameters can provide valuable security information.

Table 4.1 shows all the metrics analysed by SysStat. Like the Snort IDS, SysStat also logs possible anomalies in these parameters. In this way, it also generates a log file for use in building the ML model.

Metric ID	**Event**
rxpck	rxpck/s: packet receiving rate (unit: packets/second)
txpck	txpck/s: packet transmitting rate (unit: packets/second)
rxkB	rxkB/s: data receiving rate (unit: Kbytes/second)
txkB	txkB/s: data transmitting rate (unit: Kbytes/second)
rxcmp	rxcmp/s: compressed packets receiving rate (unit: Kbytes/second)
kbmemfree	Amount of free memory available in kilobytes
kbmemused	Amount of used memory in kilobytes. This does not take into account memory used by the kernel itself
memused	Percentage of used memory
kbbuffers	Amount of memory used as buffers by the kernel in kilobytes
kbcached	Amount of memory used to cache data by the kernel in kilobytes
kbcommit	Amount of memory in kilobytes needed for the current workload. This is an estimate of how much RAM/swap is needed to guarantee that there never is out of memory
commit	Percentage of memory needed for current workload in relation to the total amount of memory (RAM+swap). This number may be greater than 100% because the kernel usually overcommits memory
kbactive	Amount of active memory in kilobytes (memory that has been used more recently and usually not reclaimed unless absolutely necessary)
kbinact	Amount of inactive memory in kilobytes (memory that has been less recently used. It is more eligible to be reclaimed for other purposes)
kbdirty	Amount of memory in kilobytes waiting to get written back to the disk
kbanonpg	Amount of non-file backed pages in kilobytes mapped into userspace page tables

kbslab	Amount of memory in kilobytes used by the kernel to cache data structures for its own use
kbkstack	Amount of memory in kilobytes used for kernel stack space
kbpgtbl	Amount of memory in kilobytes dedicated to the lowest level of page tables
kbvmused	Amount of memory in kilobytes of used virtual address space
usr	Percentage of CPU utilization that occurred while executing at the user level (application). Note that this field does NOT include time spent running virtual processors
nice	Percentage of CPU utilization that occurred while executing at the user level with nice priority
iowait	Percentage of time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request
steal	Percentage of time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor
guest	Percentage of time spent by the CPU or CPUs to run a virtual processor
pswpin	The total number of swap pages the system brought in per second
pswpout	The total number of swap pages the system brought out per second
cswch	The total number of context switches per second

Table 4.1: SysStat metrics. Adapted from (GODARD, 2015)

4.3.6 Security Layer 2 (SMS 2)

Security layer 2 is similar to security layer 1, as both use a Snort IDS and a OS parameter analysis tool. However, the OS used in this layer is Windows 10, which results in a change in the parameter analysis tool as well, i.e. PerfMon (specific to Windows environments) was used in this layer.

Table 4.2: PerfMon Metrics. Adapted from (PERFMON, 2023)

Physical network interface card activity	**Idle time (Instantaneous)**
Network Interface	Sent packets/s
	Received packets/s
	Sent Bytes/s
	Received Bytes/s
Memory	Available KBytes
	% Confirmed Bytes in use
	Cache Bytes
Processor	% User time
% Hyper-V Processor	% Hyper-V execution time
	Context switching/s
	Monitor transition cost
Event logs	Events/s
Process	Virtual Bytes
	Thread number
	I/O Data operation/s
System	Context switching/s
	Processor queue length
	Threads
TCPv4	Active connections
	Connection fails
WinNAT	ICMP error packets dropped
	Sessions/s

Snort Windows

It was used in this layer like security layer 1, but the ruleset is different from the first, this factor aims to create diversity in the security layers.

PerfMon

As Perfmon is a similar tool to SysStat, its purpose is the same (PERFMON, 2023). However, the set of parameters analysed is different. The table 4.3 shows this set of parameters.

4.3.7 Security Layer 3

In this layer, the standard Ubuntu Linux OS was installed, and the security tool here is a DPI implemented based on Iptables.

IPTables

The IPTtables tool is used to set up, maintain and inspect the tables of IPv4 and IPv6 packet filtering rules in the Linux kernel. Many different tables can be defined, and each contains some built-in chains and can also contain user-defined chains (IPTABLES, 2023). This work used a set of rules specifically designed for use in SCADA systems, more specifically for the Modbus protocol (NIVETHAN; PAPA, 2016).

The testbed setup involved customization of the DPI implemented based on Iptables, which can analyse specific aspects of the payload, such as what is the Modbus function of a given packet. Specific rules have been created to detect possible attacks targeting the Modbus protocol for operations (presented in the Smod section of the table 4.3) and, consequently, the SCADA systems that use it. The standard TCP port used by Modbus is 502. Windows 10 OS was used on this machine with the installation of the SCADABr system.

4.3.8 SCADABr

SCADABr is a Brazilian open source project that aims to work as a SCADA web server for the control of industrial environments. It works as the core of an automation system, monitoring all devices and providing organised access to their controls and parameters (SCADABR, 2023).

As this is a supervisory system for industrial environments, it was necessary to create and configure the components to be monitored by the system. ModbusPal was chosen as the tool to perform this task. ModbusPal is presented in the next subsection.

4.3.9 ModbusPal

ModbusPal is a Modbus slave simulator, designed to provide a user-friendly interface with the capability to replicate complex and realistic ICS environments (MODBUSPAL, 2023).

In the ModbusPal simulator, three slave devices have been created which are accessed and monitored via the ScadaBR web server. The communication between the parties represents the normal traffic of Modbus packets in the environment.

4.3.10 Honeypot

A Honeypot is a system that aims to simulate the real operation of a specific target by providing data, applications, etc., to deceive possible criminals into thinking they are dealing with the real target, however, this is just a controlled environment for receiving the attacks. For this purpose, Conpot was used, a specific Honeypot for SCADA systems, which can simulate simple or more complex environments and distinguishes itself through its low-interaction approach (CONPOT, 2023).

This design involves minimal engagement with potential adversaries, creating a cybersecurity decoy that emulates vulnerable ICS systems with surface-level interaction capabilities. By strategically limiting interactivity, Conpot reduces the risks associated with more involved honeypot setups, making it an effective tool for detecting and analyzing potential threats within ICS environments.

4.3.11 Floodlight Controller

The Floodlight is an open source controller that uses the OpenFlow protocol (ATLASSIAN, 2023). This protocol is an open standard maintained by the Open Network Foundation. Based on OpenFlow, a remote controller (Floodlight) can modify the behavior of the network through a set of routing instructions. This allows the Floodlight controller to control ICS network flows through its set of routing rules.

4.3.12 OpenFlow Switch

An OpenFlow switch is nothing more than a switch that communicates via the OpenFlow protocol (NETWORKING, 2019) with other components and an external controller, in this case the Floodlight Controller. It consists of flow tables and group tables for searching and forwarding. The communication between the switch and the controller is therefore extremely important, as the controller manages the switch using the OpenFlow protocol.

This enables the controller to proactively and dynamically add, update and delete flow entries in the table.

4.3.13 NTP (Network Time Protocol)

After installing and configuring all the tools described above, a problem arose with the time synchronisation between the virtual machines present in the environment. It was therefore necessary to install the Network Time Protocol (NTP) tool (NTP, 2023) on all machines. Synchronisation was important because time was a highly relevant feature in log generation and subsequent analysis.

4.3.14 TCPDump

Tcpdump was used to capture all packets in the traffic during a time window corresponding to the execution time of an attack, which was approximately 2 hours each. All packets captured during the time window were saved to a capture file Packet Capture (PCAP), which was later processed by Flowtbag to generate flow statistics based on a set of standard features.

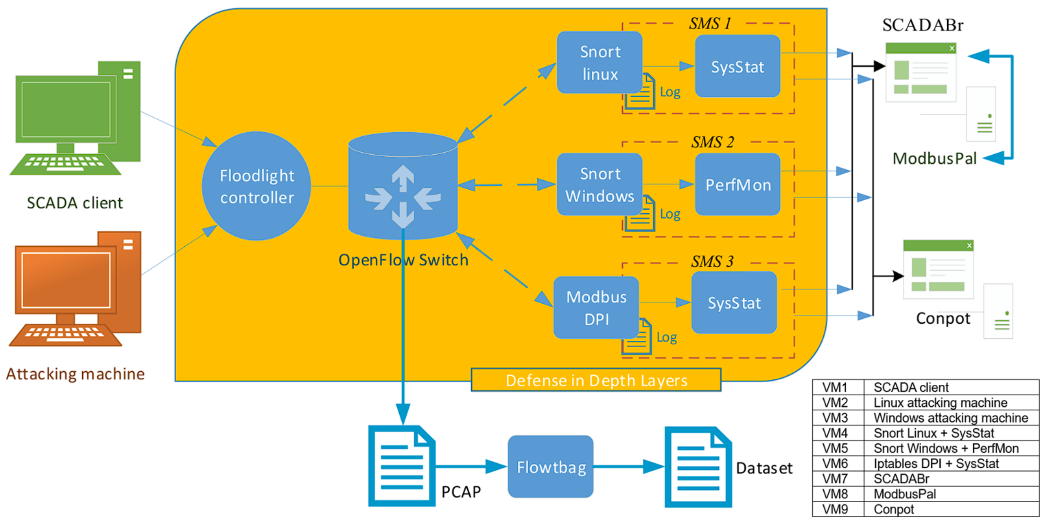
4.3.15 Flowtbag

Flowtbag is a tool that processes traffic files PCAP to extract flow statistics when running offline (ARNDT, 2023). These statistics were of great importance as the ML models were partly based on them.

An important aspect related to Flowtbag is that its standard version does not provide the timestamp of network flows in the output file, a key feature to find out which of the network flows were classified as an attack by detection tools such as Snort and DPI. Thus this flag was inserted into the source code, developed in the GO language, to ensure that the tool can now generate such data at the end of execution.

4.4 Testbed Scenario

To evaluate this work and run the testbed, the development was divided into two phases. The first phase was responsible for running the testbed and generating a dataset of traffic data, including normal flow and attack. The second stage involved constructing machine learning models using the dataset, which were subsequently utilized in the flow classifier.



VM1	SCADA client
VM2	Linux attacking machine
VM3	Windows attacking machine
VM4	Snort Linux + SysStat
VM5	Snort Windows + PerfMon
VM6	Iptables DPI + SysStat
VM7	SCADABr
VM8	ModbusPal
VM9	Conpot

Figure 4.5: The Testbed Created in This Work.

In the first stage, also during the execution of the attack tools and of the SCADA client for normal traffic, the logs of the security tools (Snort Linux, Snort Windows, DPI, SysStat, and Perfmon) were recorded. Note that SysStat and Perfmon are responsible for OS monitoring parameters for host-based anomaly detection. The logs generated were useful in verifying whether the tools identified each flow as an attack or normal traffic. The resulting dataset includes information on whether each security tool detected the flow as an attack or not, in addition to the final label indicating whether the flow is an attack or not.

The responsibility for managing traffic among virtual machines is the Open-VSwitch Switch (FOUNDATION, 2023), which makes use of the OpenFlow protocol to route all traffic to all virtual machines. This procedure was necessary for all security tools to receive all network traffic, regardless of the destination.

Figure 4.5 shows the whole scheme mentioned above. Note that the "end product" at this stage is the dataset that is generated and later manipulated to insert information about the security tools. The Python programming language and the pandas library were used to manipulate the dataset. The timestamp recorded in both the PCAP file and the tool logs was the key field to determine whether a tool detected a flow as an attack.

Table 4.3 presents the attacks executed during the testbed. Various attack types were used to exploit vulnerabilities in the ScadaBR system, which controls the ICS environment.

The attacks carried out by the Smod tool targeted ModbusPal since this tool

Table 4.3: Tools and attacks

Security Tool	Behavior (Tool)	Network Packets
Acunetix (Windows)	SQL injection	4k
	XSS	6k
Arachni (Windows)	Code injection	17k
Nessus (Linux)	Advanced scan	35k
Nmap (Linux)	Port scan	140k
Smod (Linux)	DoS Write single coils	1k
	DoS Write single register	2k
	Get function	2k
	Read coils	320k
	Read holding register	58k
	Read input register	150k
	Scanner UID	38k
	Write single coils	45k
	Write single register	296k
TCPReplay	Normal (Workload)	5.1B

works with the Modbus/TCP protocol. The ModbusPal mimics three slave devices that are accessed and monitored by the ScadaBR Web server. Communication between both represents normal Modbus packet traffic.

Penetration testing attack tools share common features, but each tool has a specific goal that distinguishes it from others. In addition, they address different Pentest techniques, which is important, since the security solution developed in this work must provide the highest possible level of security, regardless of the technique used by a potential attacker.

4.4.1 Dataset Description

No dataset was found in the literature that could provide all the data needed to perform this work. Therefore, we have developed a testbed that includes a variety of realistic behaviors found in SCADA production environments. In this environment, the SCADA server has been deployed through SCADABR, using a set of services typically mentioned in the literature (GHOSH; SAMPALLI, 2019).

Furthermore, the constructed testbed is composed of two possible environments, called normal and attack. The normal behavior is generated by 100 client machines continuously using HTTP, HTTPS, SSH, SMTP and MODBUS protocol communication with the SCADA server. The attack environment was performed using the tools presented in the previous chapter.

Table 4.3 shows the 14 types of attacks performed, as well as the normal traffic. It also shows the number of inputs generated by Flowtbag in our testbed, for each of the traffic generated. It is important to note that the execution time for each of them was 2 hours.

A total of approximately 20GB of data was generated by combining both normal and attack traffic, with an equal split of 10GB for each. This deliberate balance in the dataset, comprising both regular network activities and simulated attack scenarios, is crucial for training machine learning models. It ensures that the model is exposed to a representative mix of normal and potentially malicious patterns, enhancing its ability to accurately classify and distinguish between the two. This balanced dataset forms a robust foundation for the evaluation and optimization of ICS environments.

By categorizing packets into flows, the Flowtbag tool extracts 40 features to construct the data schema. These features include metrics such as the number of packets sent and received, minimum and maximum packet sizes, and several others. See Table 4.4 for the complete list of features.

The features were selected based on their value for model generation. The Random Forest algorithm was used for implementation with default parameters. Features with an information gain greater than 0, determined by a decision tree, were kept. Some features from the Perfmon and SysStat data were removed.

The files generated by SysStat no longer include the following features: 'txpck', 'txkB', 'rxcmp', 'txcmp', 'kbvmused', 'nice', 'steal', 'irq', 'guest', and 'gnice'. In Perfmon, the memory features were also removed. Additionally, it is worth noting that the undersampling technique was applied for data normalization.

4.5 Chapter Discussion

This chapter presents the objectives of each tool necessary for executing the testbed based on the proposed architecture. It also discusses configurations and modifications made to the source code of some tools to ensure proper execution of the environment and simplify certain tasks.

As discussed in subsection 2.2.1, if the security tools used in the environment have open CVEs, i.e., reported vulnerabilities that have not yet been fixed can

Table 4.4: The list of Flowtbag features. Adapted from (ARNDT, 2023)

Category	Name	Description
Identifier	srcip	Source ip address
Identifier	srcport	Source port number
Identifier	dstip	Destination ip address
Identifier	dstport	Destination port number
Identifier	protol	Application protocol TCP or UDP
Feature	total_fpackets	Total packets in the forward direction
Feature	total_fvolume	Total bytes in the forward direction
Feature	total_bpackets	Total packets in the backward direction
Feature	total_bvolume	Total bytes in the backward direction
Feature	min_fpktl	Size of the smallest forward packet
Feature	mean_fpktl	Mean size of forward packets
Feature	max_fpktl	Size of the largest forward packet
Feature	std_fpktl	Standard deviation from the mean of the forward packets
Feature	min_bpktl	Size of the smallest backward packet
Feature	mean_bpktl	Mean size of backward packets
Feature	max_bpktl	Size of the largest backward packet
Feature	std_bpktl	Standard deviation from the mean of the backward packets
Feature	min_fiat	Minimum amount of time between two forward packets
Feature	mean_fiat	Mean amount of time between two forward packets
Feature	max_fiat	Maximum amount of time between two forward packets
Feature	std_fiat	Standard deviation from the mean time between two forward packets
Feature	min_biat	Minimum amount of time between two backward packets
Feature	mean_biat	Mean amount of time between two backward packets
Feature	max_biat	Maximum amount of time between two backward packets
Feature	std_biat	Standard deviation from the mean time between two backward packets
Feature	duration	Duration of the flow
Feature	min_active	Minimum time that the flow was active before idle
Feature	mean_active	Mean time that the flow was active before idle
Feature	max_active	Maximum time that the flow was active before idle
Feature	std_active	Standard deviation from the mean time that the flow was active before idle
Feature	min_idle	Minimum time a flow was idle before becoming active
Feature	mean_idle	Mean time a flow was idle before becoming active
Feature	max_idle	Maximum time a flow was idle before becoming active
Feature	std_idle	Standard devation from the mean time a flow was idle before turn active
Feature	sflow_fpackets	Average number of packets in a forward sub flow
Feature	sflow_fbytes	Average number of bytes in a forward sub flow
Feature	sflow_bpackets	Average number of packets in a backward sub flow
Feature	sflow_bbytes	Average number of packets in a backward sub flow
Feature	fpsh_cnt	Number of PSH flags in forward packets
Feature	bpsh_cnt	Number of PSH flags in backward packets
Feature	furg_cnt	Number of URG flags in forward packets
Feature	burg_cnt	Number URG flags in backward packets
Feature	total_fhlen	Total bytes used for headers in the forward direction
Feature	total_bhlen	Total bytes used for headers in the backward direction
Feature	dscp	First set DSCP field for the flow
Label	class	Flow class

expose the environment. Therefore, the anomaly detector serves as an additional layer of security, which cooperates with the security tool within each layer. This makes the proposed architecture more robust and reliable. After configuring the entire environment, it was possible to thoroughly evaluate the proposed work. The evaluation will be discussed in the following chapter.

5 Evaluation

This thesis was financed in part by the Coordenação de Apereiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

After generating and preparing the normal and attack traffic datasets, this section evaluates the proposed ML-based Integral DiD to provide security in ICS environments by addressing three main research questions:

- (***RQ1***) *How does accuracy affect flow forwarding heuristics?*
- (***RQ2***) *Can the proposed SFC flow classifier be used to identify which security tool should be used?*
- (***RQ3***) *Does the chaining mechanism increase the confidence level of the security system?*

To answer these research questions, the proposed model and the selected operating point are used to create a chain to detect attacks. The impact of this chaining mechanism on the overall confidence of the security system is evaluated.

The following sections outline the construction of the models used in the evaluation and describe their performance in creating an attack detection chain.

5.1 Model Building

The models discussed in this section were built using the dataset previously mentioned, which include 6 different tools (Snort Linux, Snort Windows, Iptables, SysStat, Perfmon and Normal traffic). The training, testing, and validation steps were performed with a split of 56% for training, 14% for validation, and 30% for testing. Given that each tool was executed for 2 hours, these values were selected as the dataset was partitioned based on timestamps. The initial hour was allocated for training, the first half of the second hour for testing, and the latter half of the second hour for validation.

Four widely used classification models were evaluated: a Decision Tree, Random Forest, Gaussian Naive Bayes and Gradient Boosting. These algorithms were selected based on their widespread use and proven performance. The classifiers were trained using the training and test datasets, and their final accuracy was reported using the validation dataset. This approach ensures that the model's performance is evaluated on data it has not seen before, providing a more accurate assessment of its ability to generalize to new examples. By using separate datasets for training, testing, and validation, we can minimize the risk of overfitting and obtain a reliable estimate of the model's accuracy. A random undersampling without a replacement algorithm was applied to the training data to remove the imbalance among the classes.

Several important metrics were used to evaluate the results presented in this chapter. Precision measures the number of positive predictions that were made correctly (True Positives). Recall measures the number of positive cases correctly predicted out of all the positive cases in the dataset. F1-Score is a measure that combines both Precision and Recall and can be seen as the harmonic mean of the two.

5.2 SFC Flow Classifier Models

Figure 5.1 shows the SFC Flow Classifier model overview, comprising two main modules: Dynamic Selection and Forwarder. The Dynamic Selection module goal is to adjust the used security mechanisms based on the current OT network traffic. The model assesses the network traffic behavior and intelligently selects a subset of security mechanisms that are most suitable for classifying the incoming network traffic behavior. The main insight is to select which security tools should be used in a proactive manner to ensure that the system can maintain its classification reliability. Consequently, the proposed model can enhance the system's accuracy, even when facing dynamic attacker behavior. The forwarder is responsible to send the traffic to the selected layer.

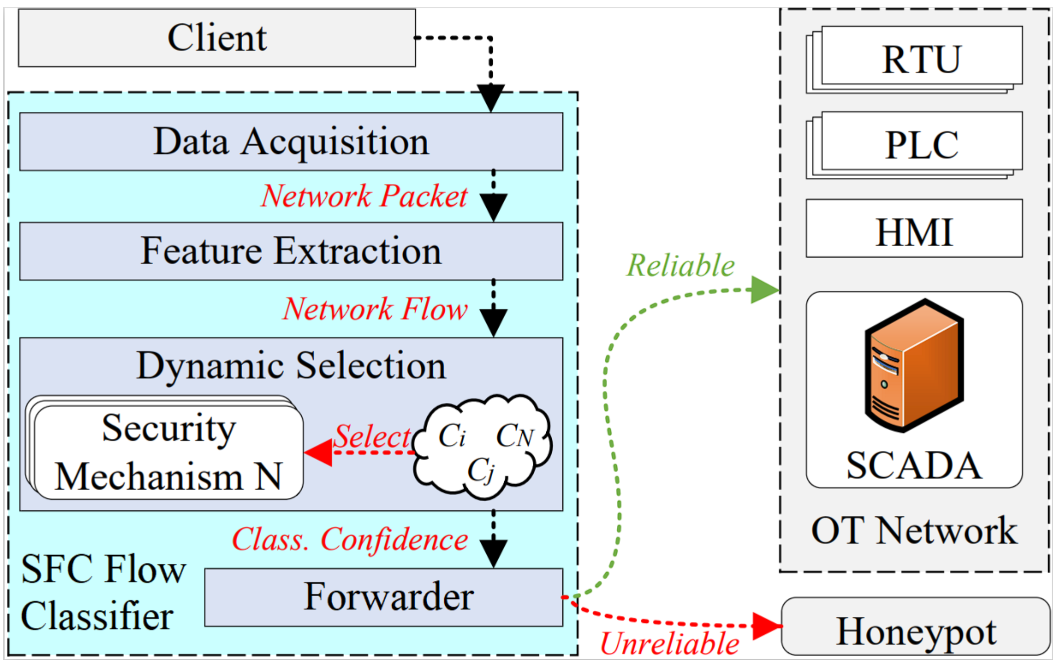


Figure 5.1: SFC Flow Classifier. The proposed scheme proactively selects the most suitable security mechanisms tailored for the current network traffic.

Regarding the SFC, a MultiLayer Perceptron (MLP) was considered to act as an engine to dynamically select the best path to a packet based on its characteristics. Functioning as a classifier, the MLP assesses packet features and makes dynamic decisions to route it according to established security policies.

The MLP was implemented using the scikit-learn v1.0.2 API, for Python v3.8.10. Two hidden layers were used, the first consisting of 500 neurons, and the second with 250 neurons. In addition, the hidden layers rely on the rectified linear unit (relu) activation function, with a learning rate of 0.001, the adam optimizer, and 200 epochs. For the output, 4 neurons were used. Three of them represent a tool that could be used to redirect a packet, while the remaining one is used to classify whether a packet could be an attack or not.

The following metrics were computed: True Positive (TP) - The number of correct positive predictions; True Negative (TN) - The number of correct negative predictions; False Positive (FP) - The number of incorrect positive predictions; False Negative (FN) - The number of incorrect negative predictions; Accuracy - The proportion of correct predictions (TP + TN) / (TP + TN + FP + FN); Recall - The proportion of actual positive predictions that were correctly predicted (TP) / (TP + FN); Precision - The proportion of positive predictions that were correct (TP) / (TP + FP).

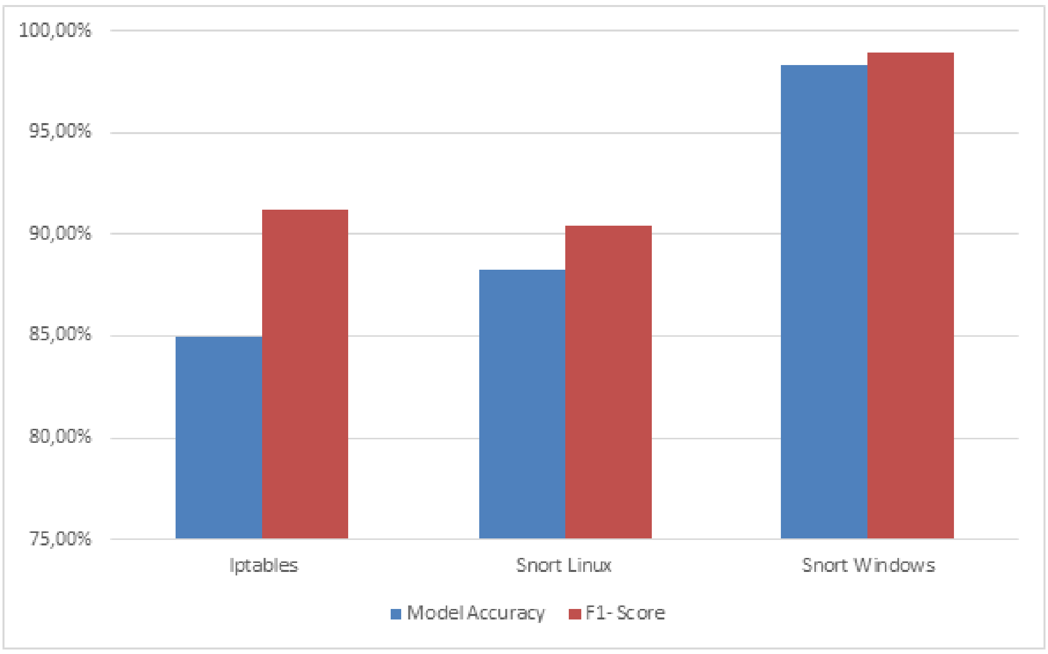


Figure 5.2: MLP Models for IPTables DPI, Snort Linux, Snort Windows.

To answer the research questions presented earlier in this chapter, it is necessary to analyze the data from each ML model created. Thus, Figure 5.2 therefore shows the data relating to the model generated from the data logged by IPTables, Snort Linux, and Snort Windows.

Of the three models, Snort Windows had the best overall accuracy in detecting all attacks, and normal traffic, with 98.29% and an F1-Score of 92.29%. This suggests that the model has an important ability to correctly identify positive instances and minimize false positives. It was also the best at detecting Acunetix attacks (SQL Injection and XSS), with 91%. Thus, such a model is the best candidate to be the first in the chain to analyze a possible low similarity packet, and for SQL injection and XSS attacks.

The lowest accuracy presented by Snort Windows was 81.89% for the XSS attack (even being the best at detecting such an attack).

The model generated for Iptables DPI achieved an accuracy of 84.99% and an F1-score of 91.21%. It only achieved the best accuracy for the specific attacks for SCADA environments launched by the Smod tool. This was to be expected since DPI was implemented focusing on such attacks. Therefore, a Modbus attack should be directed to this model since it has the highest similarity of the three.

When analyzing the IPTables model, the results suggest that the model performs well, with a high accuracy rate and a balance between precision and recall, which can be seen from the F1 score.

Snort Linux showed very good accuracies for SQL Injection (99.67%), Port Scan (99.88%), Code Injection (99.90%), Advanced Scan (99.82%), Get Function (99.88%) and Normal Traffic (99.78%). The weakness of this model is therefore the ModBus attacks generated by the Smod tool.

Snort Linux achieved an accuracy of 88.21% and an F1 score of 90.41%. Once again, the results indicate good performance, with a slightly higher accuracy rate than the previous model and an F1 score close to that of IPTables.

As shown in Figure 5.2, Snort Linux underperformed Snort Windows overall, but achieved the best accuracies, all above 99%, for the following traffic types: Arachni code injection, Nessus Advanced scan, Nmap port scan, and normal traffic. Thus, such attacks are more easily detected by this model.

It is important to note that, in general, IPTables and both Snorts present a diversity in their rules, which is an important aspect related to the detection of different types of attacks. Therefore, they complement each other in creating a robust security environment. The subsequent paragraphs will delve into specific aspects related to attacks.

The Acunetix tool was used for two types of attack, XSS and SQL Injection. First was detected by all three classifiers with an average accuracy of over 75%. The latter, on the other hand, was detected with the highest accuracy by both Snorts, with more than 99%. For the Iptables classifier, the SQL Injection accuracy was around 85%.

Port scanning using Nmap was detected over 99% of the time by both Snort classifiers. In Iptables, the accuracy was over 97%.

The code injection attack performed by the Arachni tool produced different results according to the different classifiers. The Iptables accuracy was 61,2%, Snort Linux 89,6%, and Snort Windows 99,9%.

For the advanced scanning attack performed by the Nessus tool, again both Snorts showed an interesting accuracy of over 91,6%, while Iptables had a result of around 68,5%.

The remaining attacks were all generated by the Smod tool (specific for Modbus environments). Of these, the DoS Write Single Coils attack was best detected by the Snort Linux classifier (98,4%). The other classifiers had an accuracy of around 86%.

For the DoS Write Single Register attack, the Iptables and Snort Windows classifiers yielded the best results compared to Snort Linux, with an accuracy rate of approximately 95.8%.

For the attacks on Read Coils, Read Holding Register, and Read Input Register, both Iptables and Snort Windows achieved accuracies of over 96% in most cases. However, Snort Linux performed poorly, with accuracies close to 50%.

For the Write Single Coils attack, the Snort Linux classifier presented the best accuracy (98.4%), while the Iptables and Snort Windows classifiers had around 86%.

The Write Single Register attack was better detected by Iptables and Snort Windows classifiers, with 95.8% of accuracy, and Snort Linux with 85.9%.

For the Get Func attack, Snort Linux performed well, with a 99.8% of accuracy, while the others presented around 88.5%.

The UID Scanner attack was better detected by the Snort Linux classifier, with 98,6% of accuracy. The others had an accuracy of around 88,5%.

And for detecting normal traffic, all three classifiers performed well, with Snort Linux and Snort Windows standing out with a 99.8% accuracy.

Thus, it is evident that the classifiers generated by these tools complement each other in detecting various types of attacks. If one classifier is not effective in detecting a particular type of attack, another classifier can compensate for this deficiency.

The Iptables classifier demonstrates noteworthy performance in detecting Smod tool attacks due to its rules being specifically designed for this purpose. In contrast, the results of Snorts varied depending on the type of attack and tool used. It is worth noting that Snort Linux had poor accuracy in detecting specific Modbus attacks using the Smod tool.

The Receiver Operator Characteristic (ROC) curve for the MLP containing IPTables, Snort Windows, and Snort Linux is presented in Figure 5.3.

5.3 Anomaly Host-based Detection Models

The current module comprises the detection of anomalies in a host using ML. The features considered for the current step were obtained from SysStat and PerfMon logs. For each feature set, a RF was used as the classifier, using 100 decision trees as the base estimators, with the gini as the split criterion.

Four different classifiers were used to build the anomaly host based model,

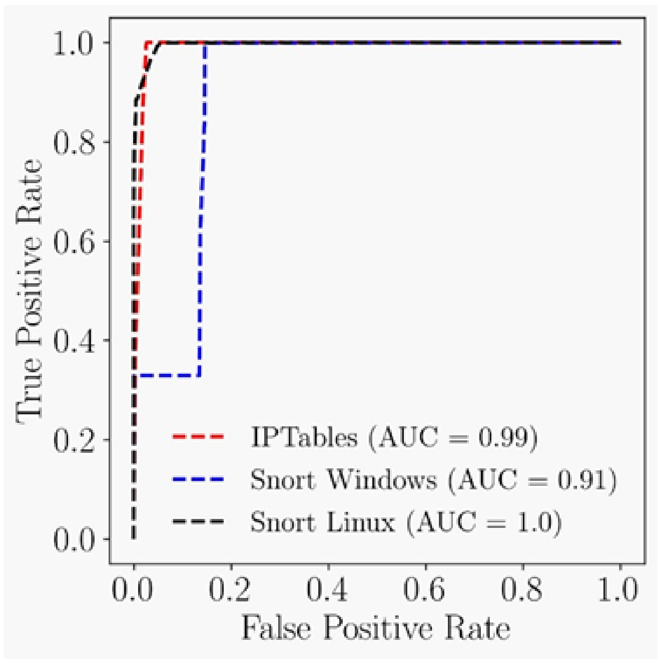


Figure 5.3: ROC curve for the MLP created based on IPTables, Snort Windows, and Snort Linux.

DT, GNB, RF, and GB. In this way, different results were achieved according to each classifier.

5.3.1 SysStat Model

Figure 5.4 presents the results of the model created for SysStat according to the four classifiers mentioned above. When analyzing the results obtained in Linux (SysStat), the four classifiers presented most of the accuracies between 90% and 100%.

The Decision Tree only performed worse on attacks related to the Acunetix tool, with 85.7% accuracy. For the others, the result was between 93.7% and 100%.

The Gaussian NB classifier had 100% accuracy in all attacks except normal traffic, which had an accuracy of 0.

Random Forest showed great results with approximately 100% accuracy for all attacks, except for the Acunetix tool, with 91.22%, and normal traffic with 75.95%.

Gradient Boosting had similar accuracy to Random Forest, but the result for Acunetix was 84.18% and normal traffic was better than the previous one, with

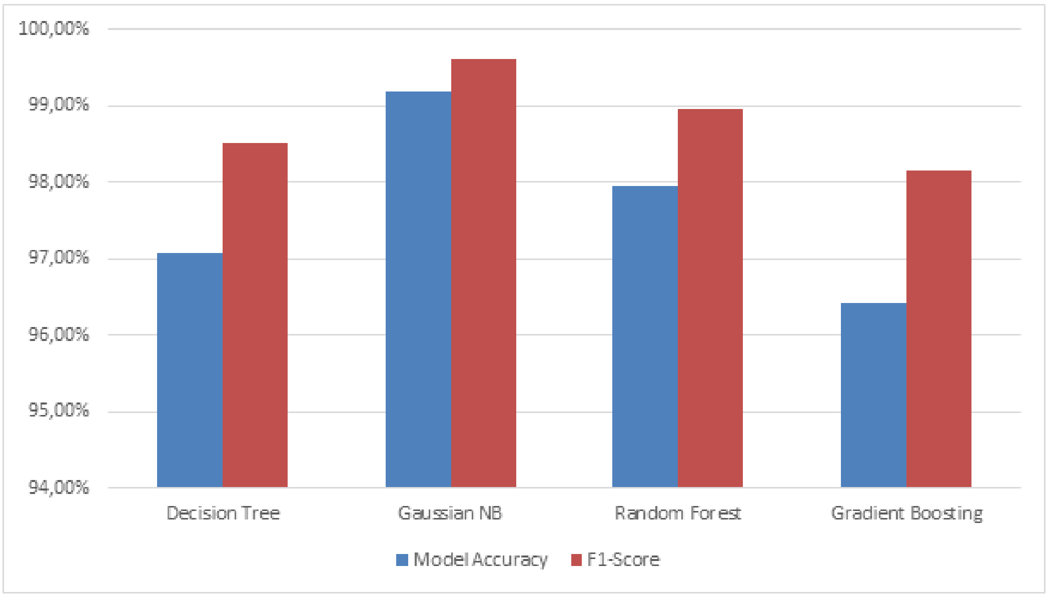


Figure 5.4: Models for SysStat.

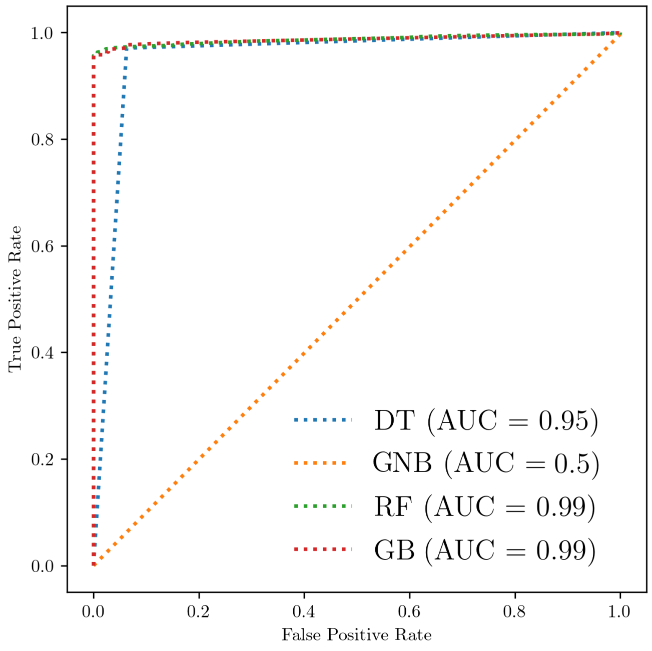


Figure 5.5: ROC curve for the SysStat model with four classifiers.

97.18%.

The ROC curve for the SysStat model is presented in Figure 5.5 below. Note that only one of the classifiers had a low AUC, the GNB (50%).

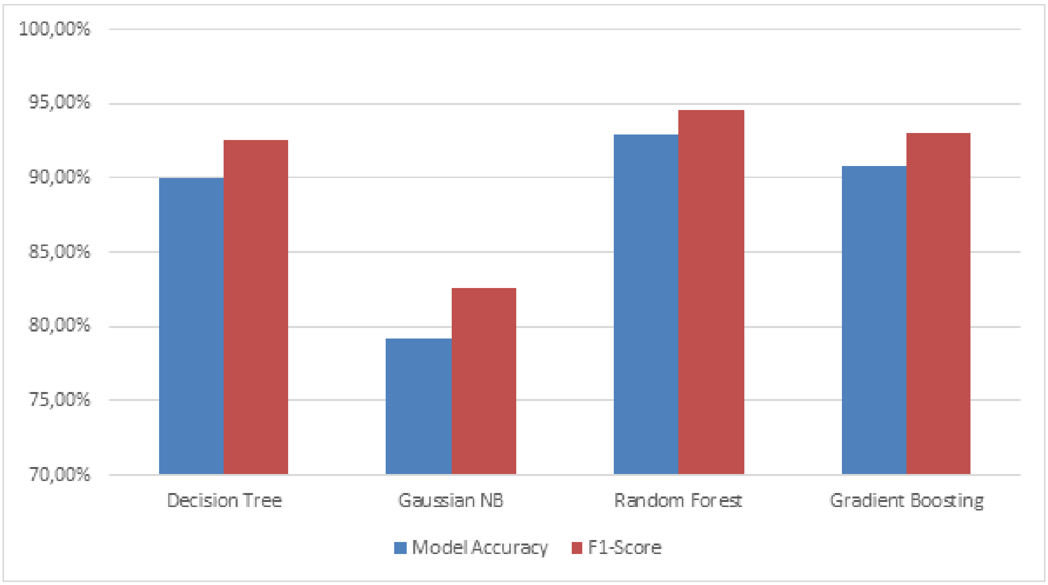


Figure 5.6: Models for PerfMon.

5.3.2 Permon Model

Figure 5.6 presents the results obtained in the Perfmon (Windows) model using the same four classifiers used in the SysStat models.

Analyzing the results obtained in the Perfmon model, the four classifiers presented most of the accuracies between 88,6% and 98%. Decision Tree varied between 89,1% and 92,6% of accuracy.

Gaussian NB did not perform well in detecting attacks generated by Arachni, which had an accuracy of only 57.3%. The others ranged from 88.6% to 93.7%.

The Random Forest classifier performed the best result in normal traffic, with an accuracy of 98.08%.

Gradient Boosting presented an accuracy between 89,8% and 91,7% for all types of traffic.

The memory-related features were removed from the PerfMon model generation as they did not provide any information gain. Thus, we can conclude that the analysis of memory features was not relevant for the attacks performed in this work.

Figure 5.7 shows the ROC curve for the Perfmon model. The evaluated classifiers demonstrated similar accuracy rates based on their OS feature sets.

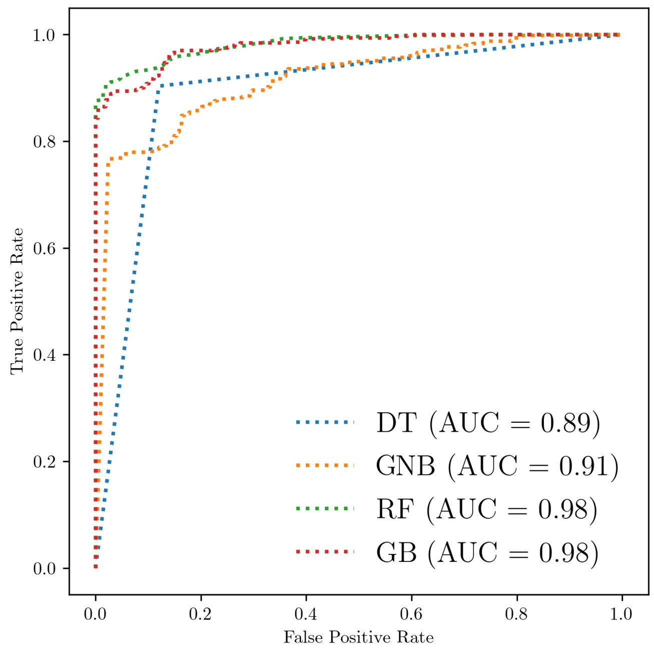


Figure 5.7: ROC curve for the PerfMon model with four classifiers.

5.4 Security Monitoring System Discussion

Among the three models generated for Iptables DPI, Snort Linux, and Snort Windows, the latter had the best average accuracy in detecting all attacks, and normal traffic, with 92.75%. In addition, Snort Windows was also the best at detecting Acunetix attacks (SQL Injection and XSS), with 91%. Thus, such a model is the best candidate to be the first in the chain to analyze a possible low similarity packet. The same can be said for SQL injection and XSS attacks.

As for Snort Linux, it performed very well against almost all types of attacks. Only the SCADA-specific attacks, which were sent by the Smod tool showed low accuracy rates. For the Arachni code injection attacks, the Nessus Advanced scan, the Nmap Port scan, and the normal traffic, the model built for Snort Linux showed the best accuracy rates, all above 99%. This means that such attacks are more easily detected by this model.

The model generated for Iptables DPI achieved only the best accuracy for the specific attacks for SCADA environments launched by the Smod tool. This was to be expected since DPI was implemented focusing on such attacks. Therefore, a Modbus attack should be directed to this model as it has the highest similarity of the three.

Regarding the host-based anomaly models of SysStat (Linux) and Perfmon (Windows), the former obtained the best overall accuracy among all classifiers used, with 92.7%. It is also important to note that the classifier with the best average accuracy for SysStat was Gradient Boosting, with 96.7%. The Decision Tree classifier had a similar result, with 96.5%.

Among the classifiers created for Perfmon, the Random Forest had the best accuracy, approximately 97%.

In response to RQ1, the impact of accuracy on flow forwarding heuristics is crucial in optimizing resource allocation in the security environment to achieve the highest level of confidence. This process involves carefully selecting the most appropriate tools to analyze each packet, taking into account their respective capabilities. Thus, the heuristics efficiently guide traffic, ensuring that the selected tools can accurately determine whether a packet has malicious origins. This approach greatly enhances effectiveness in the security environment, supported by well-founded and precise decision-making.

5.5 Intelligent Defense in Depth

The generated Machine Learning models were inserted within to mitigate possible attacks. Also, they were inserted into the Floodlight Controller to classify and target new flows in real-time.

At this stage, following the analysis flow of Algorithm 1, the Floodlight Controller, responsible for dynamically selecting classifiers, achieved optimal routing for each packet that entered the Ethernet interface and passed through the SFC flow classifier, which uses similarity calculation.. The pseudocode that implements the classifiers decision is shown in Algorithm 1.

To address RQ2, we analyze the previously presented algorithm that performs a similarity analysis between the input packet and those used in model training. This allows the packet to be sent to the most appropriate tools for accurate classification. The Floodlight controller implements the flow classification algorithm, which identifies the best tool or chain of tools to detect potential attacks or normal traffic.

The determination of the path a packet should take is customizable. This task can be divided into three steps: similarity calculation, classification level 1 (tools), and classification level 2 (anomaly).

Algorithm 1 Intelligent Defense in Depth (Prototype)

```
1: **Initial setup**
2: **procedure** GETPACKET(packet)
3:     $flowClassifier \leftarrow$ similarity($packet$)
4:     **if** $flowClassifier$ = SnortWindows **then**
5:         send($packet$;
6:         $classificationSW \leftarrow$ SnortWindowsClassifier($packet$);
7:         $AnomalySW \leftarrow$ SnortWindowsAnomaly($packet$))
8:         **if** ($classificationSW$ = 'Attack') $\wedge$ ($AnomalySW$ = 'Attack') **then**
9:             **block**(packet)
10:        **else if** ($classificationSW$ = 'Normal') $\wedge$ ($AnomalySW$ = 'Normal') **then**
11:            send($packet, SCADABr$)
12:        **end if**
13:    **else if** $flowClassifier$ = SnortLinux **then**
14:        send($packet$;
15:        $classificationSL \leftarrow$ SnortLinuxClassifier($packet$);
16:        $AnomalySL \leftarrow$ SnortLinuxAnomaly($packet$))
17:        **if** ($classificationSL$ = 'Attack') $\wedge$ ($AnomalySL$ = 'Attack') **then**
18:            **block**(packet)
19:        **else if** ($classificationSL$ = 'Normal') $\wedge$ ($AnomalySL$ = 'Normal') **then**
20:            send($packet, SCADABr$)
21:        **end if**
22:    **else if** $flowClassifier$ = DPItool **then**
23:        send($packet$;
24:        $classificationDPI \leftarrow$ DPI($packet$);
25:        $AnomalyDPI \leftarrow$ DPIAnomaly($packet$))
26:        **if** $classificationDPI$ = 'Attack' **then**
27:            **block**(packet)
28:        **else if** $classificationDPI$ = 'Normal' **then**
29:            send($packet, SCADABr$)
30:        **else**
31:            send($packet, honeypot$)
32:        **end if**
33:    **end if**
34: **end procedure**
```

Table 5.1: Whole System Accuracy.

Classifier	Flow label	Accuracy	F1-score	Blocked	Scada	Honeypot
DT	Win	92.08%	94.42%	1.19%	70.23%	28.57%
	Linux	97.10%	98.52%	1.19%	70.23%	28.57%
GNB	Win	83.50%	87.37%	1.19%	70.23%	28.57%
	Linux	97.17%	98.55%	1.19%	70.23%	28.57%
GB	Win	92.08%	94.69%	1.21%	70.22%	28.57%
	Linux	97.89%	98.93%	1.21%	70.22%	28.57%
RF	Win	87.13%	91.79%	1.23%	70.20%	28.57%
	Linux	99.56%	99.78%	1.23%	70.20%	28.57%

The initial task is to calculate similarity using the predict_proba function from the scikit-learn library. This function utilizes the results obtained from the previously presented DPI and Snorts models to select the most suitable candidate for analyzing the newly arrived package.

After the calculation is complete, the packet is sent to either Snort or DPI and then to either SysStat or Perfmon for analysis. These tools determine whether the packet is normal traffic or a possible attack.

During the third step, the level 2 classifier identifies the packet's destination as either SCADABr, the Honeypot, or blocking it.

The algorithm 1 presents all the possibilities according to the previously chosen path for sending the packet and analysis by one of the tools, as well as the OS anomaly detector SMS.

Currently, a conservative approach is employed. If the indicated tool and the anomaly detector do not agree that the traffic is normal, it may be sent to the Honeypot or blocked if both indicate that the traffic is an attack. This approach can be tailored to various scenarios and environments.

The algorithm 1 was implemented using the four ML algorithms previously used in SysStat and PerfMon. Thus, Table 5.1 presents the results obtained. It is important to note that all algorithms used a threshold of 0.05 as the operating point. These values were obtained from the F1-Score metric.

This table displays accuracy values ranging from 83% to 99.5%. These values indicate whether the packet was labeled as an attack or normal by the developed SFP system.

For example, when analyzing the DT algorithm for the Linux OS, we have an accuracy of 97.1% and F1-score of 98.5%. This means that after the two classifications performed, packets were correctly sent to SCADABr, Honeypot, or correctly blocked correctly 97.1% of the time. Such a statement can be made with 95% confidence since the operating point is 0.05.

In general, the four classifiers obtained satisfactory results, with Random Forest standing out on Linux OS, with 99.5% accuracy. For the Windows OS, the best results were obtained from DT and GB, with 92.07%.

When analyzing the F1-score, we have values always above 90%, which indicates that very bad cases in the scenario of this work, such as high rates of false negatives is low. Thus, the solution presented here becomes a relevant solution for ICS environments, with high accuracy and low overhead.

Since DPI is responsible for performing a comprehensive analysis of network packets, its computational requirements are typically higher than other tools. As a result, DPI operates independently in this capacity and is therefore not included in the Table 5.1.

When analyzing the last three columns of Table 5.1, it can be observed that, across all classifiers, approximately 1.2% of attacks were blocked, 70.2% were forwarded to SCADA, and 28.6% were directed to the Honeypot. The traffic sent to the latter is one of the major contributions of this study, as if the problem were approached using traditional DiD techniques, some of the attacks within the 28.6% sent to the Honeypot could be treated as normal traffic and sent to SCADABr. The total number of packets used in this test was 832,258.

Figure 5.8 presents the ROC curve for the Linux OS. The evaluated classifiers demonstrated similar accuracy rates based on their OS feature sets, with an AUC of 1.00 for all classifiers.

For the Windows OS, the ROC curve is shown in Figure 5.9. Of the four classifiers shown in the figure, DT has the lowest AUC at 0.80 compared to the others.

Regarding the models generated for Windows and Linux paths, the choice of the best classifier to be used can be made by analyzing the average classification time for each of them. The execution times of DT, RF, and GB ranged from 0.002 to 0.005 seconds. Only GNB had a higher time of 0.0237 and 0.0249.

This characteristic is very important (VIEGAS et al., 2016), because a high latency time can make a good security solution unusable for ICS.

Of all the classifiers, the GNB is the slowest of all. This is a disadvantage when compared to the others, which have a similar time for both OS (in seconds). The table 5.2 presents the elapsed time of all classifiers.

Finally, we show that the security level of an ICS environment can be in-

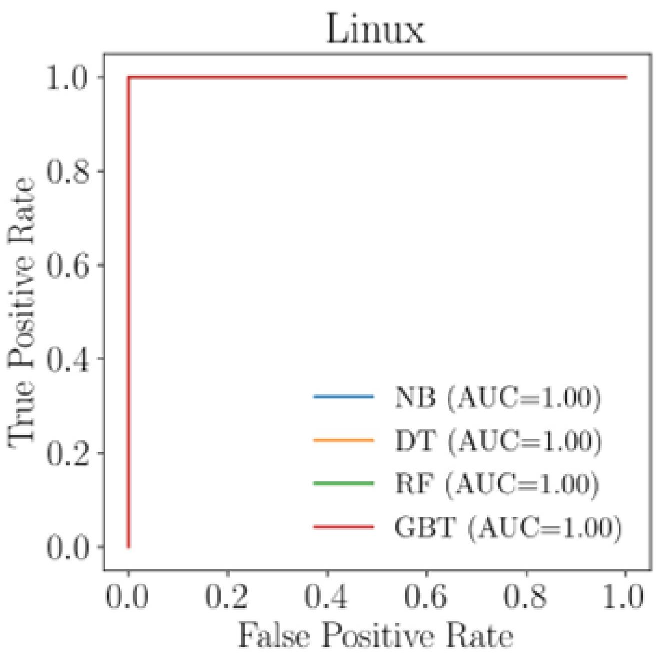


Figure 5.8: ROC curve for the Linux OS model with four classifiers.

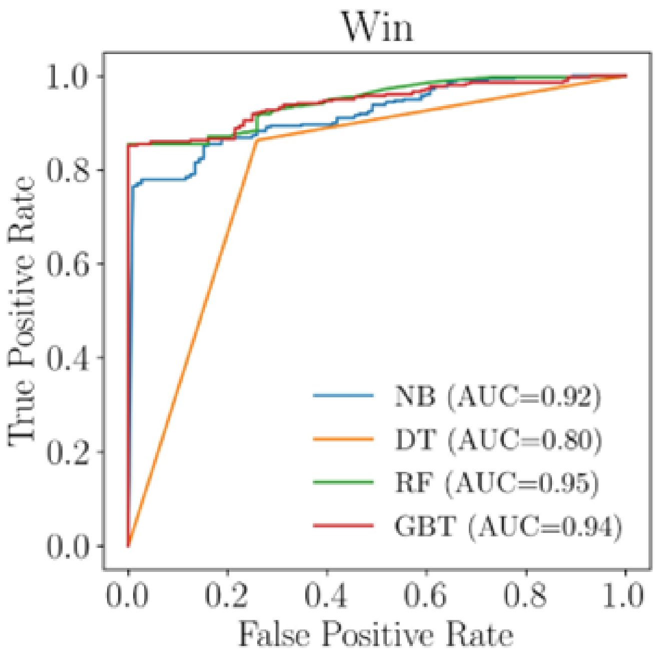


Figure 5.9: ROC curve for the Windows OS model with four classifiers.

creased by using ML models, together with the concatenation of security functions. This combination can optimize DiD routing techniques and specific security tools by utilizing packet classification and dynamic routing through the SFC flows. Therefore, we can answer RQ3 based on the results presented here.

Table 5.2: Elapsed Time (in seconds) for all Classifiers in Linux and Windows.

Classifier	Flow label	Average (seconds)	Std Deviation (seconds)
DT	Windows	0.0003	0.0005
	Linux	0.0003	0.0005
GNB	Windows	0.0249	0.0081
	Linux	0.0237	0.0054
RF	Windows	0.0002	0.0005
	Linux	0.0002	0.0005
GB	Windows	0.0004	0.0005
	Linux	0.0005	0.0008

5.6 Limitations

The use of different types of attacks in relation to those used to create the ML models in the classifiers may be considered a limitation. This is because the detection of new attack categories may have lower accuracy than the values presented here. However, the use of Honeypot helps to mitigate this problem by collecting data to retrain the models and continuously improve the security environment.

It is important to note that the flow classifier plays a central role in this work. If it is compromised, the entire system can be seriously impacted. One potential solution to this issue is to implement a clustered solution, which would allow for the replication of key components, including the classifier used in this work.

The honeypot's collected data requires processing for model insertion and updates. However, labeling a large amount of data is a limitation. However, labeling a large amount of data is a limitation. To address this, a semi-supervised learning technique could be employed with expert support.

Finally, determining when machine learning models are outdated can be a critical aspect of this work. Therefore, it is possible to establish a policy for updating models based on specific intervals. Additionally, other policies can be developed and put into practice.

5.7 Chapter Discussion

Relevant results have been obtained regarding the detection of various types of attacks against ICS/SCADA environments. It has been demonstrated that a well-designed set of attack detection tools can effectively mitigate potential attacks with high accuracy, in conjunction with the ML models developed in this study.

Diversity among security tools is crucial to creating a defense environment with multiple layers, each with its own specialty. This approach contributes to an intelligent and modular defense system that minimizes potential bottlenecks.

The latency generated by a security solution must be minimal for it to be feasible to implement in an ICS environment. Therefore, this work highlights the feasibility of implementing the presented models, including dynamic classifier selection. The relevance of this aspect is demonstrated by the previously presented classification times.

When a packet is sent to the honeypot, it indicates that even after being classified by specific machine learning models, it is impossible to unambiguously determine whether the analyzed packet is an attack or normal traffic. This means that the classification decision is unknown.

To enhance the security level of the proposal during model retraining, it is crucial to analyze the behavior of potential new attacks and traffic when classification decisions are unknown. This analysis will help update the generated models and improve overall security as traffic changes over time.

The use of classifiers, particularly in the heuristics of the flow classifier, is crucial in limiting an attacker's ability to compromise Defense in Depth (DiD) or take control of this security structure. This approach increases complexity by requiring the intruder to not only overcome each individual layer of the DiD but also to understand and neutralize the automatic decisions of the flow classifier. This strengthens the integrity and resilience of the security architecture.

6 Conclusion and Future Works

This work introduces a novel architecture and implementation of DiD within an ICS environment, with a focus on intelligent selection of security tools based on incoming packets. The approach involves dynamic traffic routing within the flow classifier, ensuring that each access attempt can be directed through different SFC flows. Consequently, this intricacy in flow routing complicates an attacker's ability to control DiD, adding an additional layer of defense against potential intrusion.

Another aspect of this work is its flexibility in terms of the security tools used in our prototype. Our architecture allows for easy addition or removal of security tools, making it adaptable to changing needs.

Anomaly host-based detection has been integrated with security tools to evaluate packets in DiD flows. Both classifiers determine whether the packet content is an attack or normal, improving the reliability of traffic classification for ICS by 28.6% compared to the current state of the art in DiD, which does not incorporate an anomaly detector based on OS parameters.

From the results obtained, we can state that the solution proposed in this work is a significant alternative to the standard DiD technique, which is state of the art, as it presents improvements concerning the best security function chain to detect a possible attack, and has diversity among the tools, working with an anomaly detection system based on OS monitoring. In addition, the machine learning models created and used in this work present better accuracy when compared to off-the-shelf security tools.

In addition to the aforementioned advantages, the use of a honeypot is another significant contribution. The honeypot receives inconclusive (unknown) classifier decisions and serves as a controlled environment for collecting attack data to adjust the ML models created here in the future.

Regarding the overhead introduced by machine learning models, it is important to note that the additional cost is low. This makes the solution scalable and viable in ICS, where time is often highly relevant.

6.1 Future Works

As future work, a more significant number of security tools and, consequently, different sets of flows can be tested. A more comprehensive range of Pentests can be applied to obtain more of the results previously presented with new types of attacks and consequently can improve this work. Furthermore, testing this work in a real ICS scenario would be very interesting as possible new features to be implemented.

This work suggests potential research areas for future studies, including optimizing flow forwarding heuristics, enhancing the flow classifier through advanced machine learning methods, and exploring the architecture's interoperability with existing security standards. Additionally, improving anomaly detection in industrial environments is a relevant area for subsequent research. Advanced metrics and practical considerations regarding security policies and compliance should be included to offer significant contributions to the ongoing advancement of industrial cybersecurity.

References

ACUNETIX. *Acunetix*. 2023. Disponível em: <https://www.acunetix.com>.

AGGARWAL, C. C. et al. *Data mining: the textbook*. [S.l.]: Springer, 2015. v. 1.

AHAKONYE, L. A. C.; NWAKANMA, C. I.; LEE, J.-M.; KIM, D.-S. Agnostic ch-dt technique for scada network high-dimensional data-aware intrusion detection system. *IEEE Internet of Things Journal*, v. 10, n. 12, p. 10344–10356, 2023.

ALANAZI, M.; MAHMOOD, A.; CHOWDHURY, M. J. M. Scada vulnerabilities and attacks: A review of the state-of-the-art and open issues. *Computers & Security*, Elsevier, p. 103028, 2022.

ALEM, S.; ESPES, D.; NANA, L.; MARTIN, E.; De Lamotte, F. A novel bi-anomaly-based intrusion detection system approach for industry 4.0. *Future Generation Computer Systems*, v. 145, p. 267–283, 2023. ISSN 0167-739X.

ALLIANCE, N. 5g white paper. *Next generation mobile networks, white paper*, v. 1, n. 2015, 2015.

ALQAHTANI, S.; GAMBLE, R. F. Ddos attacks in service clouds. In: IEEE. *2015 48th Hawaii International Conference on System Sciences*. [S.l.], 2015. p. 5331–5340.

ANANTHA, D. N.; RAMAMURTHY, B. Sciencesds: A novel software defined security framework for large-scale data-intensive science. In: *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. [S.l.: s.n.], 2017. p. 13–18.

ARACHNI. *Arachni*. 2023. Disponível em: <https://www.arachni-scanner.com>.

ARNDT, D. *Flowtbag*. 2023. Disponível em: <https://github.com/DanielArndt/flowtbag>.

ATLASSIAN. *Floodlight Controller*. 2023. Disponível em: <https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview>.

AUTOMATION, I. *What is Scada*. 2018. Disponível em: <https://inductiveautomation.com/resources/article/what-is-scada>.

BAUER, R.; HESEDING, H.; FLITTNER, M. Earlydrop: A trade-off driven ddos defense mechanism for software-defined infrastructures. In: IEEE. *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. [S.l.], 2017. p. 207–210.

BISHOP, C. M.; NASRABADI, N. M. *Pattern recognition and machine learning*. [S.l.]: Springer, 2006. v. 4.

BONDAN, L.; WAUTERS, T.; VOLCKAERT, B.; TURCK, F. D.; GRANVILLE, L. Z. A framework for sfc integrity in nfv environments. In: SPRINGER INTERNATIONAL PUBLISHING. *Security of Networks and Services in an All-Connected World: 11th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2017, Zurich, Switzerland, July 10-13, 2017, Proceedings 11*. [S.l.], 2017. p. 179–184.

BRINGER, M. L.; CHELMECKI, C. A.; FUJINOKI, H. A survey: Recent advances and future trends in honeypot research. *International Journal of Computer Network and Information Security*, Modern Education and Computer Science Press, v. 4, n. 10, p. 63, 2012.

BULLE, B. B.; SANTIN, A. O.; VIEGAS, E. K.; SANTOS, R. R. dos. A host-based intrusion detection model based on os diversity for scada. In: IEEE. *IECON 2020 The 46th annual conference of the IEEE industrial electronics society*. [S.l.], 2020. p. 691–696.

CERT, I. *Attack statistics*. 2023. Disponível em: <https://ics-cert.kaspersky.com/statistics>.

CHEN, X.; ZHANG, D.; WANG, X.; ZHU, K.; ZHOU, H. P4sc: Towards high-performance service function chain implementation on the p4-capable device. In: IEEE. *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. [S.l.], 2019. p. 1–9.

CHOU, L.-D.; TSENG, C.-W.; HUANG, Y.-K.; CHEN, K.-C.; OU, T.-F.; YEN, C.-K. A security service on-demand architecture in sdn. In: IEEE. *2016 International Conference on Information and Communication Technology Convergence (ICTC)*. [S.l.], 2016. p. 287–291.

COMMISSION, I. E. et al. Iec 62443-4-1: 2018: Security for industrial automation and control systems-part 4-1: Secure product development lifecycle requirements. *International Electrotechnical Commission*, 2018.

CONPOT. *Conpot Honeypot*. 2023. Disponível em: <https://hub.docker.com/r/honeynet/conpot>.

DANEELS, A.; SALTER, W. What is scada? 1999.

DEPHILLIPS, M.; KATRAMATOS, D.; BHATTACHARYYA, S. Making a case for high-bandwidth monitoring-a use case for analysis on the wire. In: IEEE. *2017 New York Scientific Data Summit (NYSDS)*. [S.l.], 2017. p. 1–6.

DOMINGOS, P. A few useful things to know about machine learning. *Communications of the ACM*, ACM New York, NY, USA, v. 55, n. 10, p. 78–87, 2012.

FOUNDATION, L. *OpenVSwitch*. 2023. Disponível em: <https://www.openvswitch.org>.

FOUNDATION, O. N. *OpenFlow Switch Specification*. 2015. Disponível em: <https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>.

FYSARAKIS, K.; PETROULAKIS, N. E.; ROOS, A.; ABBASI, K.; VIZARRETA, P.; PETROPOULOS, G.; SAKIC, E.; SPANOUDAKIS, G.; ASKOXYLAKIS, I. A reactive security framework for operational wind parks using service function chaining. In: IEEE. *2017 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.], 2017. p. 663–668.

GARCIA, M.; BESSANI, A.; GASHI, I.; NEVES, N.; OBELHEIRO, R. Analysis of operating system diversity for intrusion tolerance. *Software: Practice and Experience*, Wiley Online Library, v. 44, n. 6, p. 735–770, 2014.

GERRITZ, C. *Depth is a Flawed Cyber Strategy*. 2018. Disponível em: <https://www.cyberdefensemagazine.com/special-report-defense-in-depth-is-a-flawed-cyber-strategy/>.

GHEORGHE, A.; MASERA, M.; WEIJNEN, M.; VRIES, L. D. Critical infrastructures at risk. *Securing the European electric power system*, Springer, 2006.

GHOSH, S.; SAMPALLI, S. A survey of security in scada networks: Current issues and future challenges. *IEEE Access*, IEEE, v. 7, p. 135812–135831, 2019.

GODARD, S. *SYSSTAT utilities home page*. 2015. Disponível em: <http://sebastien.godard.pagesperso-orange.fr>.

GROUP, N. B. D. P. W. et al. *NIST Special Publication 1500-1-NIST Big Data Interoperability Framework: Volume 1, Definitions. NIST Spec Publ [Internet]. 2015; 1: 32*. 2018.

GUNES, H.; PICCARDI, M. Affect recognition from face and body: early fusion vs. late fusion. In: IEEE. *2005 IEEE international conference on systems, man and cybernetics*. [S.l.], 2005. v. 4, p. 3437–3443.

HALPERN, J.; PIGNATARO, C. *RFC 7665: service function chaining (sfc) architecture*. [S.l.]: RFC Editor, 2015.

HAN, B.; GOPALAKRISHNAN, V.; JI, L.; LEE, S. Network function virtualization: Challenges and opportunities for innovations. *IEEE communications magazine*, IEEE, v. 53, n. 2, p. 90–97, 2015.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H.; FRIEDMAN, J. H. *The elements of statistical learning: data mining, inference, and prediction*. [S.l.]: Springer, 2009. v. 2.

HEMSLEY, K. E.; FISHER, E. et al. *History of industrial control system cyber incidents*. [S.l.], 2018.

HU, Z.; YIN, Y. A framework for security on demand. In: IEEE. *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. [S.l.], 2017. p. 378–383.

IPTABLES. *IPTables*. 2023. Disponível em: <https://www.netfilter.org/projects/iptables/index.html>.

KASPERSKY. *Threat landscape for industrial automation systems. Statistics for H2 2022*. 2023. Disponível em: <https://ics-cert.kaspersky.com/publications/reports/2023/03/06/threat-landscape-for-industrial-automation-systems-statistics-for-h2-2022/>.

KAYAN, H.; NUNES, M.; RANA, O.; BURNAP, P.; PERERA, C. Cybersecurity of industrial cyber-physical systems: a review. *ACM Computing Surveys (CSUR)*, ACM New York, NY, v. 54, n. 11s, p. 1–35, 2022.

LAI, Y.; ZHANG, J.; LIU, Z. Industrial anomaly detection and attack classification method based on convolutional neural network. *Security and Communication Networks*, Hindawi Limited, v. 2019, p. 1–11, 2019.

LANGNER, R. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, IEEE, v. 9, n. 3, p. 49–51, 2011.

LINUX, K. *Kali Linux*. 2023. Disponível em: <https://www.kali.org>.

LIU, C.-G. A novel moving target defense scheme with physical unclonable functions-based authentication. *IEEE Access*, IEEE, v. 10, p. 23051–23062, 2022.

LYONS, B. Applying a holistic defense-in-depth approach to the cloud. 2011.

MASSONET, P.; DUPONT, S.; MICHOT, A.; LEVIN, A.; VILLARI, M. Enforcement of global security policies in federated cloud networks with virtual network functions. In: IEEE. *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*. [S.l.], 2016. p. 81–84.

MASSONET, P.; DUPONT, S.; MICHOT, A.; LEVIN, A.; VILLARI, M. A motivating case study for coordinating deployment of security vnf in federated cloud networks. In: SPRINGER. *Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2016, Vienna, Austria, September 5–7, 2016, Revised Selected Papers 5*. [S.l.], 2018. p. 34–42.

MELL, P.; SHOOK, J.; HARANG, R. Measuring and improving the effectiveness of defense-in-depth postures. In: *Proceedings of the 2nd Annual Industrial Control System Security Workshop*. [S.l.: s.n.], 2016. p. 15–22.

MIEDEN, P.; BELTMAN, R. Network anomaly detection in modbus tcp industrial control systems. *Tech. Rep.*, University of Amsterdam, 2020.

MILLER, B.; ROWE, D. A survey scada of and critical infrastructure incidents. In: *Proceedings of the 1st Annual conference on Research in information technology*. [S.l.: s.n.], 2012. p. 51–56.

MODBUSPAL. *ModbusPal*. 2023. Disponível em: <https://github.com/zeelos/ModbusPal>.

MÖLLER, N.; HANSSON, S. O.; HOLMBERG, J.-E.; ROLLENHAGEN, C. *Handbook of safety principles*. [S.l.]: John Wiley & Sons, 2018. v. 9.

NESSUS. *Nessus*. 2023. Disponível em: <https://pt-br.tenable.com/products/nessus>.

NETWORKING, O. *Software Defined Networking*. 2019. Disponível em: <https://www.opennetworking.org/sdn-definition>.

NGUYEN, V.-G.; BRUNSTROM, A.; GRINNEMO, K.-J.; TAHERI, J. Sdn/nfv-based mobile packet core network architectures: A survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 19, n. 3, p. 1567–1602, 2017.

NIVETHAN, J.; PAPA, M. On the use of open-source firewalls in ics/scada systems. *Information Security Journal: A Global Perspective*, Taylor & Francis, v. 25, n. 1-3, p. 83–93, 2016.

NMAP. *Nmap*. 2023. Disponível em: <https://nmap.org/book/man.html>.

NTP. *NTP.br*. 2023. Disponível em: <https://ntp.br>.

OUYANG, Y.; LI, B.; KONG, Q.; SONG, H.; LI, T. Fs-ids: A novel few-shot learning based intrusion detection system for scada networks. In: *ICC 2021 - IEEE International Conference on Communications*. [S.l.: s.n.], 2021. p. 1–6.

PERFMON. *Perfmon*. 2023. Disponível em: <https://learn.microsoft.com/pt-br/windows-server/administration/windows-commands/perfmon>.

PETROULAKIS, N. E.; FYSARAKIS, K.; ASKOXYLAKIS, I.; SPANOUDAKIS, G. Reactive security for sdn/nfv-enabled industrial networks leveraging service function chaining. *Transactions on Emerging Telecommunications Technologies*, Wiley Online Library, v. 29, n. 7, p. e3269, 2018.

RADOGLOU-GRAMMATIKIS, P.; SINIOSOGLOU, I.; LIATIFIS, T.; KOUROUNIADIS, A.; ROMPOLOS, K.; SARIGIANNIDIS, P. Implementation and detection of modbus cyberattacks. In: IEEE. *2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST)*. [S.l.], 2020. p. 1–4.

RAJESH, L.; SATYANARAYANA, P. Evaluation of machine learning algorithms for detection of malicious traffic in scada network. *Journal of Electrical Engineering & Technology*, Springer, p. 1–16, 2021.

REPLAY, T. *TCP Replay*. 2023. Disponível em: <https://tcpreplay.appneta.com>.

RUNNELS, G. M. Implementing defense in depth at the university level. *GSEC Practical*, n. 1.4, 2002.

SANZ, I. J.; LOPEZ, M. A.; MATTOS, D. M. F.; DUARTE, O. C. M. B. A cooperation-aware virtual network function for proactive detection of distributed port scanning. In: IEEE. *2017 1st Cyber Security in Networking Conference (CSNet)*. [S.l.], 2017. p. 1–8.

SCADABR. *ScadaBR*. 2023. Disponível em: <https://www.scadabr.com.br>.

SECURITY, D. of H. *Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies*. 2016. Disponível em: <https://www.cisa.gov/uscert/sites/default/files/recommended_practices/NCCIC_ICS-CERT_Defense_in_Depth_2016_S508C.pdf>.

SHAMELI-SENDI, A.; JARRAYA, Y.; POURZANDI, M.; CHERIET, M. Efficient provisioning of security service function chaining using network security defense patterns. *IEEE Transactions on Services Computing*, IEEE, v. 12, n. 4, p. 534–549, 2016.

SHENG, C.; YAO, Y.; LI, W.; YANG, W.; LIU, Y. Unknown attack traffic classification in scada network using heuristic clustering technique. *IEEE Transactions on Network and Service Management*, IEEE, 2023.

SMOD. *Smod*. 2023. Disponível em: <https://github.com/theralfbrown/smod-1>.

SNORT. *Snort IDS*. 2023. Disponível em: <https://www.snort.org>.

THOMAS, G. Introduction to the modbus protocol. *The Extension*, v. 9, n. 4, p. 1–4, 2008.

TRENDMICRO. *Industrial Control System*. 2023. Disponível em: <https://www.trendmicro.com/vinfo/us/security/definition/industrial-control-system>.

VIEGAS, E.; SANTIN, A. O.; FRANCA, A.; JASINSKI, R.; PEDRONI, V. A.; OLIVEIRA, L. S. Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems. *IEEE Transactions on Computers*, IEEE, v. 66, n. 1, p. 163–177, 2016.

VIRTUALBOX. *Virtualbox*. 2023. Disponível em: <https://www.virtualbox.org>.

YANG, H.; CHENG, L.; CHUAH, M. C. Deep-learning-based network intrusion detection for scada systems. In: IEEE. *2019 IEEE Conference on Communications and Network Security (CNS)*. [S.l.], 2019. p. 1–7.

YOON, C.; LEE, S.; KANG, H.; PARK, T.; SHIN, S.; YEGNESWARAN, V.; PORRAS, P.; GU, G. Flow wars: Systemizing the attack surface and defenses in software-defined networks. *IEEE/ACM Transactions on Networking*, IEEE, v. 25, n. 6, p. 3514–3530, 2017.

ZHANG, L.; LV, Z.; ZHANG, X.; CHEN, C.; LI, N.; LI, Y.; WANG, W. A novel approach for traffic anomaly detection in power distributed control system and substation system. In: SPRINGER. *Network and System Security: 13th International Conference, NSS 2019, Sapporo, Japan, December 15–18, 2019, Proceedings 13*. [S.l.], 2019. p. 408–417.

ZHOU, X.; XU, Z.; WANG, L.; CHEN, K.; CHEN, C.; ZHANG, W. Construction and evaluation of defense-in-depth architecture in scada system. In: EDP SCIENCES. *MATEC Web of Conferences*. [S.l.], 2018. v. 173, p. 01012.