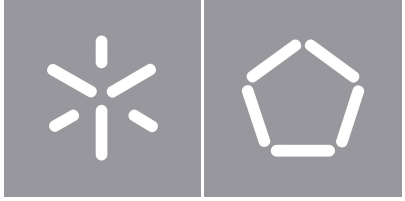
University of Minho
School of Engineering

Eva Miriam Pires de Castro

Security Data Analytics in 6G Open Networks

october 2024



University of Minho
School of Engineering

Eva Miriam Pires de Castro

Security Data Analytics in 6G Open Networks

Master's Dissertation in Telecommunications and Computer Engineering

Dissertation supervised by
Professora Doutora Maria João Nicolau

october 2024

Copyright and Terms of Use for Third Party Work

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

This work can thereafter be used under the terms established in the license below.

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

License granted to users of this work:

[Caso o autor pretenda usar uma das licenças Creative Commons, deve escolher e deixar apenas um dos seguintes ícones e respetivo lettering e URL, eliminando o texto em itálico que se lhe segue. Contudo, é possível optar por outro tipo de licença, devendo, nesse caso, ser incluída a informação necessária adaptando devidamente esta minuta]

CC BY

`https://creativecommons.org/licenses/by/4.0/` *[Esta licença permite que outros distribuam, remixem, adaptem e criem a partir do seu trabalho, mesmo para fins comerciais, desde que lhe atribuam o devido crédito pela criação original. É a licença mais flexível de todas as licenças disponíveis. É recomendada para maximizar a disseminação e uso dos materiais licenciados.]*

CC BY-SA

`https://creativecommons.org/licenses/by-sa/4.0/` *[Esta licença permite que outros remis-*

turem, adaptem e criem a partir do seu trabalho, mesmo para fins comerciais, desde que lhe atribuam o devido crédito e que licenciem as novas criações ao abrigo de termos idênticos. Esta licença costuma ser comparada com as licenças de software livre e de código aberto «copyleft». Todos os trabalhos novos baseados no seu terão a mesma licença, portanto quaisquer trabalhos derivados também permitirão o uso comercial. Esta é a licença usada pela Wikipédia e é recomendada para materiais que seriam beneficiados com a incorporação de conteúdos da Wikipédia e de outros projetos com licenciamento semelhante.]

CC BY-ND

`https://creativecommons.org/licenses/by-nd/4.0/` *[Esta licença permite que outras pessoas usem o seu trabalho para qualquer fim, incluindo para fins comerciais. Contudo, o trabalho, na forma adaptada, não poderá ser partilhado com outras pessoas e têm que lhe ser atribuídos os devidos créditos.]*

CC BY-NC

`https://creativecommons.org/licenses/by-nc/4.0/` *[Esta licença permite que outros remisturem, adaptem e criem a partir do seu trabalho para fins não comerciais, e embora os novos trabalhos tenham de lhe atribuir o devido crédito e não possam ser usados para fins comerciais, eles não têm de licenciar esses trabalhos derivados ao abrigo dos mesmos termos.]*

CC BY-NC-SA

`https://creativecommons.org/licenses/by-nc-sa/4.0/` *[Esta licença permite que outros remisturem, adaptem e criem a partir do seu trabalho para fins não comerciais, desde que lhe atribuam a si o devido crédito e que licenciem as novas criações ao abrigo de termos idênticos.]*

CC BY-NC-ND

`https://creativecommons.org/licenses/by-nc-nd/4.0/` *[Esta é a mais restritiva das nossas seis licenças principais, só permitindo que outros façam download dos seus trabalhos e os compartilhem desde que lhe sejam atribuídos a si os devidos créditos, mas sem que possam alterá- los de nenhuma forma ou utilizá-los para fins comerciais.]*

Acknowledgments

The conclusion of this dissertation reflects the dedication and hard work invested over several years, particularly during my master's degree. Achieving this milestone would not have been possible without the support of many.

First and foremost, I want to thank Ivo for always supporting me and providing strength when I needed it most.

I am grateful to my family — my sister, for always believing in me, and my parents, for providing the means to pursue and complete my academic journey.

I extend my sincere thanks to Professor Doctor Maria João Nicolau for being a great advisor and for all the guidance and support throughout the development of this dissertation.

I also wish to thank José and everyone at Optare Solutions for making this dissertation possible and for contributing to build my professional career.

Finally, I would like to express my appreciation to everyone who, in some way, helped me overcome this challenge. Your support has been invaluable.

Funding

This work has been partially funded by the "Ministerio de Asuntos Económicos y Transformación Digital" and the European Union-NextGenerationEU in the frameworks of the "Plan de Recuperación, Transformación y Resiliencia" and of the "Mecanismo de Recuperación y Resiliencia" under reference 6G-OPENSEC SECURITY (TSI-063000-2021-58).

Universidade do Minho

optare solutions

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

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

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

University of Minho, Guimarães, october 2024

Eva Miriam Pires de Castro

Abstract

The transition of mobile networks from rigid, proprietary architectures to open and flexible approaches, known as Open Networks, presents new security challenges. Traditional networks, dominated by "vendor islands" with specialized "black-box" hardware and software, are being replaced by "white box" hardware with open interfaces, fostering innovation and competition. The adoption of Open Networks, driven by initiatives like the O-RAN Alliance, is crucial for accelerating the deployment of 6G technology. However, this transition introduces significant security risks due to the increased complexity and involvement of multiple vendors.

This dissertation addresses these challenges by developing a system to autonomously detect and classify DDoS attacks using Machine Learning (ML), aligning with the principles of Zero-Touch Network & Service Management (ZSM) architecture. The system integrates into the Security Manager of the 6GOPENSEC-SECURITY project, specifically within the Security Closed-Loop Automation (SCLA) framework, enhancing the security of network slices in a 6G environment.

A comprehensive study of existing systems revealed a gap in the application of ZSM principles and the need for a solution tailored to 6G networks. The developed system employs a Convolutional Neural Network (CNN) for detecting various DDoS attacks and a Random Forest classifier for attack type classification. Despite challenges such as the lack of a suitable 5G dataset, the CICDDoS2019 dataset was utilized due to its relevant characteristics.

The system provides low-latency detection, generates security reports, and includes a dashboard for monitoring network security. It has been tested, demonstrating high accuracy (99.85%) and integration with external systems. This system is ready for real-world testing in a 5G network slice environment and is expected to serve as a robust foundation for enhancing security in 6G open networks.

Keywords: 6G, 5G, Cybersecurity, Machine Learning, Deep Learning, Open Networks, Zero-Touch Networks

Resumo

A transição das redes móveis de arquiteturas rígidas e proprietárias para abordagens abertas e flexíveis, conhecidas como Redes Abertas, apresenta novos desafios de segurança. As redes tradicionais, dominadas por "ilhas de fornecedores" com hardware e software especializados, estão a ser substituídas por hardware "caixa branca" com interfaces abertas, promovendo inovação e competição. A adoção de Redes Abertas, impulsionada por iniciativas como a O-RAN Alliance, é crucial para acelerar a implantação da tecnologia 6G. No entanto, esta transição introduz riscos significativos de segurança devido ao aumento da complexidade e ao envolvimento de múltiplos fornecedores.

Esta dissertação aborda esses desafios desenvolvendo um sistema para detectar e classificar autonomamente ataques DDoS usando Machine Learning (ML), alinhado com os princípios da arquitetura *Zero-Touch Network & Service Management* (ZSM). O sistema integra-se no *Security Manager* do projeto 6GOPENSEC-SECURITY, especificamente no âmbito da *framework Security Closed-Loop Automation* (SCLA), para melhorar a segurança das *network slices* num ambiente 6G.

Um estudo abrangente de sistemas existentes revelou uma lacuna na aplicação dos princípios ZSM e a necessidade de uma solução voltada para redes 6G. O sistema desenvolvido emprega uma Rede Neural Convolucional (CNN) para detectar diversos ataques DDoS e um classificador Random Forest para a classificação dos tipos de ataque. Apesar dos desafios, como a falta de um conjunto de dados adequado para redes 5G, foi utilizado o conjunto de dados CICDDoS2019 devido às suas características relevantes.

O sistema fornece detecção de baixa latência, gera relatórios de segurança e inclui um painel para monitoriação da segurança da rede. O sistema foi testado, demonstrando alta precisão e integração com sistemas externos. Este sistema está pronto para testes em cenários reais num ambiente 5G *Netowrk Slicing* (NS) e espera-se que sirva como uma base robusta para melhorar a segurança em redes abertas 6G.

Palavras-chave: 6G, 5G, cibersegurança, Machine Learning, Deep Learning, Redes Abertas, Redes *Zero-Touch*

Contents

List of Figures

List of Tables

Acronyms

2D	Two-Dimensional.
3D	Three-Dimensional.
4G	Fourth-Generation.
5G	Fifth-Generation.
6G	Sixth-Generation.
AI	Artificial Intelligence.
ANN	Artificial Neural Network.
AUC	Area Under ROC Curve.
AWS	Amazon Web Services.
CI/CD	Continuous Integration/Continuous Delivery.
CLA	Closed-Loop Automation.
CLI	Command Line Interface.
CNN	Convolutional Neural Network.
DDoS	Distributed Denial-of-Service.
DL	Deep Learning.
DNS	Domain Name System.
DoS	Denial-of-Service.
E2E	End-to-End.
ETSI	European Telecommunications Standards Institute.
FN	False Negative.
FNR	False Negative Rate.

FP False Positive.

FPR False Positive Rate.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

HIDS Host Intrusion Detection Systems.

ID3 Iterative Dichotomiser 3.

IDS Intrusion Detection Systems.

IoT Internet of Things.

IP Internet Protocol.

IT Information Technology.

JSON JavaScript Object Notation.

LSTM Long Short Term Memory.

LUCID Lightweight, Usable CNN in DDoS Detection.

MITM Man-in-the-Middle.

ML Machine Learning.

MSE Mean Squarred Error.

NFV Network Functions Virtualization.

NGMN Next Generation Mobile Networks.

NIDS Network Intrusion Detection Systems.

NS Network Slicing.

O-RAN Open Radio Access Network.

PCAP Packet Capture.

QoS Quality of Service.

ROC Receiver Operating Characteristic Curve.

SCLA Security Closed-Loop Automation.

SDA Security Data Analytics.

SDN Software Defined Networks.

SQL Structured Query Language.

SVM Support Vector Machine.

TN True Negative.

TNR True Negative Rate.

TP True Positive.

TPR True Positive Rate.

UE User Equipment.

V2X Vehicle To Everything.

ZSM Zero touch network & Service Management.

Chapter 1

Introduction

This chapter contextualizes the shift from traditional closed architectures to open and flexible paradigms in mobile networks field. Emphasizing the challenges posed by multi-vendor, disaggregated networks, the chapter explores the motivation for enhancing security measures. It outlines the dissertation objectives, focusing on the development and implementation of security mechanisms within the 6GOPENSEC-SECURITY project. The chapter finalizes with an overview of the dissertation's structure.

1.1 Context

Mobile networks are transitioning from inflexible architectures that rely on specialized "black-box" hardware with proprietary software to more open and flexible approaches known as Open Networks. Traditionally, these networks have depended on hardware and software designed and patented by a limited number of vendors, resulting in "vendor islands" where each network segment relies on specific proprietary solutions.

In recent years, there has been a significant shift towards replacing these traditional architectures with open solutions. This involves the use of "white box" hardware with open interfaces, allowing for the integration of software from any vendor [1]. This trend promotes the decoupling of hardware and software vendors, fostering greater diversity and competition within the telecommunications market.

One of the most notable initiatives driving this transition is the Open Radio Access Network (O-RAN) Alliance[1]. Open networks not only encourage innovation and the entry of new providers but also have the potential to accelerate the deployment of Sixth-Generation (6G) technology in a more competitive and cost-effective manner, reducing reliance on incumbent providers.

However, the adoption of open networks introduces significant security challenges. The presence of multiple providers in a complex environment increases the risk of new vulnerabilities [2]. In the context of Fifth-Generation (5G) networks, a new architecture called Network Slicing (NS) has emerged. A slice

[1] https://www.o-ran.org/

is a logical network offering specific capabilities and features tailored to different market scenarios. This architecture enhances the openness of mobile networks.

Security in network slices is paramount, encompassing user authentication, transaction accounting, and the detection of active security threats. With the advent of 6G networks, these security concerns are intensifying, particularly in an open, multi-vendor environment.

Given these factors and the new security challenges presented by the next generation of mobile networks, there is a pressing need for more agile, responsive, and autonomous security mechanisms that differ from traditional approaches.

1.2 Motivation

The cybersecurity landscape in the context of next-generation communication networks faces significant challenges. 6G networks, with their open and multi-vendor architectures, introduce additional complexities that existing security solutions are not adequately equipped to handle [3]. Traditional security approaches exhibit notable deficiencies in addressing the complex demands and threats impacting 6G communication infrastructure [4]. These deficiencies include an inability to respond rapidly to emerging attacks and a lack of adaptability to the dynamic and open environments of modern networks.

To address these security challenges in open and disaggregated 6G networks, the European project 6GOPENSEC-SECURITY[2] focuses on the design and implementation of an intelligent and autonomous security manager. This manager is responsible for the management of network slices with specific security requirements in multi-vendor 6G networks. A key component of this project is the implementation of a Security Closed-Loop Automation (SCLA), a proactive and adaptive strategy designed to ensure network resilience through continuous monitoring, threat identification, and the automatic deployment of countermeasures, capable of meeting the challenges posed by the next generation of communication networks.

1.3 Main Aims

The primary aim of this dissertation is to contribute to the overarching objectives of the 6GOPENSEC-SECURITY project by focusing on the design and implementation of the Security Data Analytics (SDA) component, part of the SCLA. This work plays a role in enhancing the overall security posture of 6G networks, leveraging Machine Learning (ML) techniques to detect security threats in real-time.

The general objectives of this dissertation are:

[2] https://www.cttc.cat/project/secure-network-slice-manager-for-open-and-disaggregated-6g-networks/

- Develop robust methodologies for analyzing security data to detect attacks in 6G networks.
- Contribute to the creation of an intelligent and autonomous security manager capable of operating with minimal human intervention.
- Align the development of the SDA component with the broader objectives of the 6GOPENSEC-SECURITY project, ensuring seamless integration within the overall architecture.

The specific objectives are:

- Design and implement the SDA as part of the SCLA following the Zero touch network & Service Management (ZSM)[3] architecture principles, developed by European Telecommunications Standards Institute (ETSI).
- Utilize ML models to analyze large volumes of network data, identifying patterns and abnormal behaviors indicative of security threats.
- Promote the automation of security processes, reducing the need for human intervention through automated threat detection mechanisms.
- Ensure that the SDA component operates seamlessly within the SCLA to provide continuous, adaptive security management.

By achieving these objectives, this dissertation aims to develop a system to detect security attacks with high accuracy and low latency, for efficient attack detection, automate all processes to achieve a zero-touch approach, and integrate this system into the SCLA component of the security manager of the 6GOPENSEC-SECURITY project to enhance the security of network slices and 6G networks.

It is important to note that the system will be developed to detect specifically Distributed Denial-of-Service (DDoS) attacks, once it was the test case proposed for the 6GOPENSEC-SECURITY project.

1.4 Main Contributions

The research and work achieves with this dissertation was aligned with the Research and Development project, 6GOPENSEC-SECURITY, as already mentioned, in a collaboration with *Optare Solutions* (Spain), and other partners.

[3] `https://www.etsi.org/technologies/zero-touch-network-service-management`

This research included: the survey of 5G and 6G networks and its respective security state; the study and application of ML mechanisms for DDoS detection; specification and development of an intelligent applications to enhance security.

This tasks resulted in the development of a component capable of analyzing network traffic and detecting the presence of DDoS attacks, leveraging ML mechanisms. This component is going to be tested in a real 5G environment to justify its applicability in the future 6G networks.

Additionally, this research also resulted in scientific publications:

Enhancing Network Slicing Security: Machine Learning, Software-Defined Networking, and Network Functions Virtualization-Driven Strategies [5]

José Cunha, Pedro Ferreira, Eva M. Castro, Paula Cristina Oliveira, Maria João Nicolau, Iván Núñez, Xosé Ramon Sousa, and Carlos Serôdio. Future Internet 16, no. 7: 226. 2024.

Security and Trust in Open and Disaggregated 6G networks [6]

Pol Alemany, Raúl Muñoz, R. Vilalta, Ll. Gifre, R. Martínez, R. Casellas, Eva M. Castro, P. Ferreira, D. Moreira, J. García, J. Cunha, I. Núñez, G. Gómez, S. Castro, A. Pastor and D. López. 24th International Conference on Transparent Optical Networks (ICTON). 2024.

1.5 Dissertation Structure

This dissertation is structured into six chapters, each serving a distinct purpose in the exploration and development of this dissertation:

In the first chapter, the focus is on introducing the dissertation's theme, providing context, discussing the motivation behind the chosen topic, outlining the main objectives, and giving a brief overview of the dissertation's organization.

The second and third chapters explore key theoretical concepts fundamental to the dissertation's progression. Chapter two specifically addresses 5G and 6G networks, highlighting their security challenges. Chapter three focuses on intrusion detection systems, discussing relevant research and existing work in the field. Both chapters provide essential background knowledge.

Chapter four is dedicated to describing the architecture of the solution, outlining its underlying principles, and detailing its integration within the overarching project framework.

The fifth chapter concentrates on the development of the solution. It explores the decisions made during the implementation process and outlines the methodology employed.

In the sixth chapter, case studies and testing procedures are presented. This section describes the conducted case studies, outlines the testing methodologies applied, and provides an analysis and discussion of the obtained results.

Finally, the seventh chapter concludes the dissertation by summarizing the findings and insights gained from the research. It also discusses future perspectives and potential avenues for further development within the scope of the project.

Chapter 2

Study of 5G and 6G networks

This chapter gives a thorough summary of the most recent advances and trends in telecommunications, with a particular emphasis on the development in 5G networks and the expected transition to 6G technology. This chapter attempts to contextualize the continuous evolution in the telecommunications landscape by analyzing the capabilities of 5G, such as increased data rates and decreased latency, in addition to the potential features of 6G, with the focus on the security paradigm of these networks.

2.1 5G Networks

This section presents an overview of the 5G networks, it will be provided a vision of the evolution from Fourth-Generation (4G) to 5G in terms of radio performance and new approaches for 5G architecture and deployment. Also, it will be presented what is NS and its importance, and the security as well as the security and privacy challenges associated with this technology..

2.1.1 Overview

The innovative fifth generation of cellular networks, or 5G, is designed to provide far more than just traditional cellular services. [7]. 5G was design to bring new capabilities such as higher data rates, lower latency, and multiple device connectivity [8], once due to the increasing mobile traffic and the rapid growth of communication infrastructures, 4G is no longer able to meet users' actual needs [9].

Therefore, in comparison of its antecedent, 5G communications are compromised to have [8]:

- data rate 10 times faster.
- latency 10 times slower.
- higher bandwidth and spectrum efficiency.

- low cost.
- much more connected devices.

Figure 1 synthesizes the evolution of the performance requirements from 4G to 5G, also showing some of the technologies used.

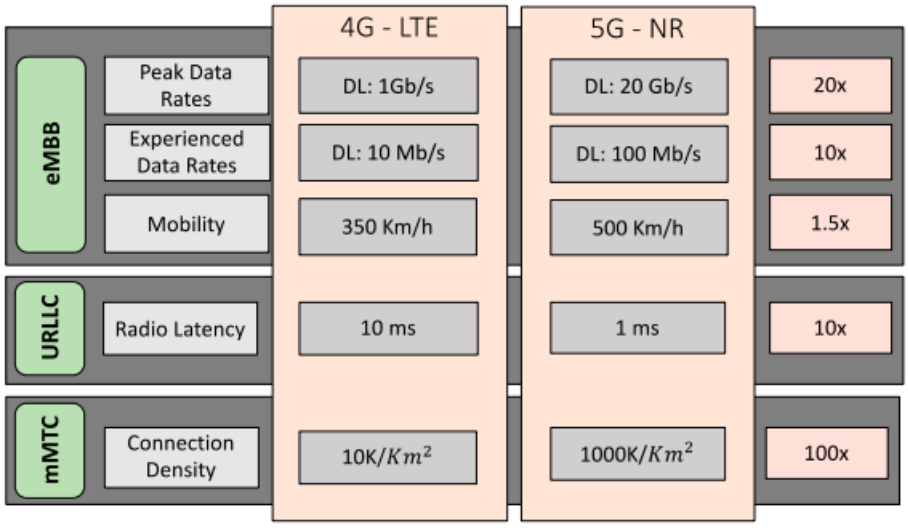


Figure 1: 5G and 4G comparison of performance requirements [7].

Beyond its improved radio capabilities, 5G's revolutionary potential comes additionally from its flexibility. Until now, cellular network deployments have relied on 'black-box' methodologies, where hardware and software are plug-and-play devices with minimal or no reconfiguration options [1]. This architecture is thought to be lacking in the scalability and flexibility required to effectively handle a wider range of business needs, each with distinct requirements for availability, scalability, and performance [10]. This rigid, monolithic infrastructure, are unable to accommodate the tight needs of 5G applications and the heterogeneity and variety of 5G scenarios [1]. So, the transition in mobile networks involves moving away from traditional inflexible architectures, characterized by dedicated hardware and proprietary firmware/software, towards disaggregated setups using open source software [1]. This new approach can be achieved by the use of technologies such as NS, Software Defined Networks (SDN), Network Functions Virtualization (NFV), etc. Furthermore, a highly flexible and scalable 5G network is required, as numerous use cases are anticipated to be active concurrently in operator networks [10]. This way 5G networks can effectively adjust to the specific needs of different use cases, making its flexibility evident, guaranteeing a smooth and responsive infrastructure to support the wide range of applications, going from smart cities and homes, to healthcare and smart transportation [7]. Figure 2 provides a visual representation of some of the applications of 5G.

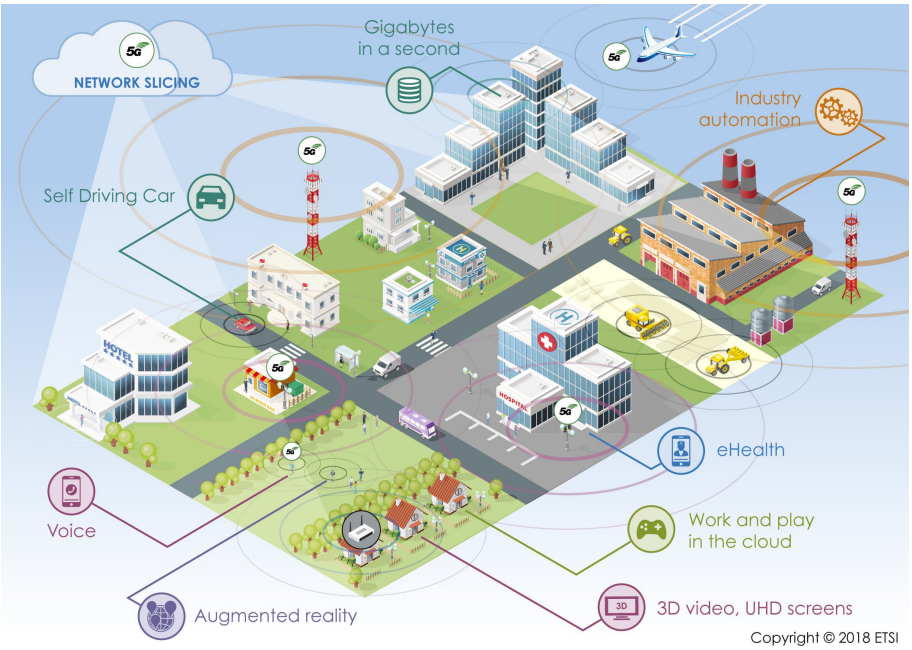


Figure 2: 5G applications [11].

2.1.2 Network Slicing

As mentioned earlier, NS is one of the key technologies used by 5G, to achieve its flexibility, symbolizing a break from conventional, monolithic network topologies by bringing a highly configurable framework.

In very simple terms, NS refers to the creation of End-to-End (E2E) isolated, tailored logical networks on top of a common underlying network infrastructure, using technologies like SDN and NFV [12], With a decided-upon service-level agreement, it is customized for a specific service type [13]. NS, a foundational concept in 5G technology, revolutionizes connectivity by implementing virtualization principles. This is a crucial technological and commercial enabler for 5G in addition to making service customisation, isolation, and multi-tenancy support easier [14].

The essence of NS lies in dividing a physical network into isolated logical networks, each dedicated to different services based on their unique requirements and characteristics [15]. It separates network functions from hardware and software components by using virtualization techniques. Users are then shown these abstracted functionalities as separate virtual networks, each with its own set of resources and functioning independently [1]. Network operators can provide unique solutions for a range of market situations [15]. This method maximizes resource usage and creates new economic opportunities by enabling service differentiation, leasing of unused resources, and real-time adaptability to traffic demand [1].

So, NS is important and revolutionary, since it may offer specialized network services to various user and application types while guaranteeing that every slice satisfies unique performance, security, and pri-

vacy requirements.

Figure 3 gives a visual representation about what is NS and some applications areas.

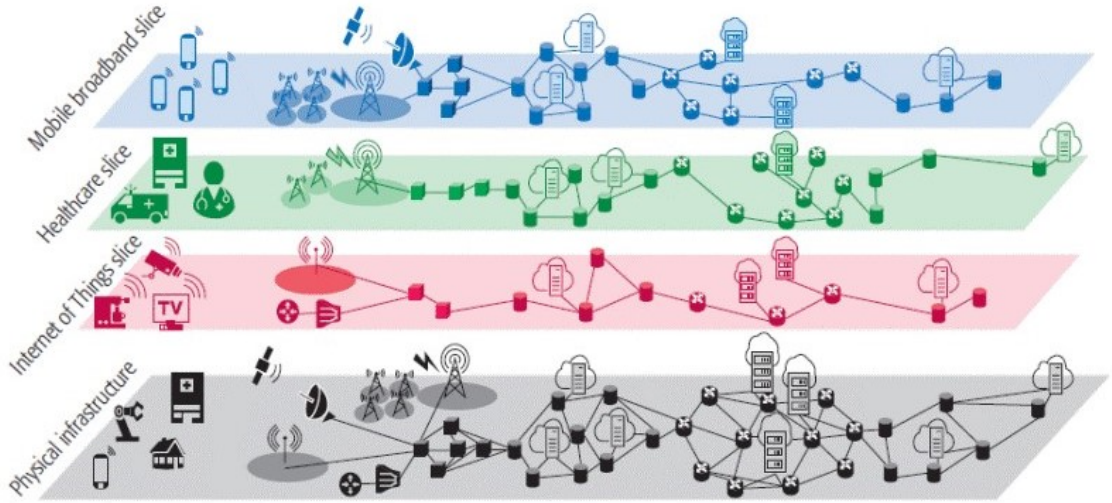


Figure 3: Network Slicing in 5G [12].

However, effective orchestration is required for network slices [15]. Orchestration is the coordination of various network operations that are designed to generate, manage, and provide services in a slicing context. In order to provide strong isolation and enable the functioning of parallel slices on a common underlying substrate, the orchestration process also involves defining relevant policies and mechanism [12]. Also, it plays a crucial role in meeting Service Level Agreements and fortifying resilience against failures and outages [1]. Its purpose is resource allocation and the management of network slices to cater to the varied requirements of distinct services [9]. This includes ensuring performance isolation to meet particular service needs and E2E performance regardless of load and performance in other slices [12].

NS has several benefits, however it poses serious security and privacy concerns. Because every slice is independent and has particular characteristics, maintaining user privacy and putting robust security measures in place across these slices present complex challenges. The Next Generation Mobile Networks (NGMN) Alliance [1], wrote some security recommendations about 5G NS, presented in [10].

The challenges in terms of privacy and security that arise from NS are complex and involve many important factors. NS creates complexities in orchestrating slices across several proprietary virtual platforms, each with distinct security characteristics, as well as inter-slice security concerns. Network sharing between slices belonging to various tenants can give rise to security issues, which is why it is important to use extra parameters to differentiate the necessary security levels for each slice [14].

Strong isolation techniques must be put in place to overcome these issues and prevent potential

[1] https://www.ngmn.org/

attacks or faults in one slice from influencing other slices. Each slice necessitates independent security functions to thwart unauthorized access to slice-specific information, emphasizing compartmentalization at each virtualization level [12]. Therefore, it is crucial to guarantee the security and isolation of every slice in order to prevent unwanted access or interference between slices [1]. *You et al.* consider the slice isolation the most crucial NS property [13].

Robust actions must be taken to avoid data breaches or unauthorized access in order to protect user privacy and their data within each slice. The problem also arises with shared infrastructure components, where extra caution is required to avoid jeopardizing the confidentiality and privacy of each slice. This necessitates a comprehensive strategy that includes robust security measures and privacy-preserving methods tailored to the specific requirements of NS to overcome these challenges [1].

In result, addressing the complex security and privacy issues raised by NS in 5G requires an extensive multi-level security framework [12]. This framework should include components such as software integrity, remote attestation, dynamic threat detection and mitigation, user authentication, and accounting management, essential to maintaining the dependability and credibility of NS in the dynamic telecommunications environment. Effectively dynamic managing of network slices is essential to accommodate diverse services while addressing challenges related to isolation, security, and scalability [13]. Also, ML can be a great ally in achieving security in the 5G NS environment. Due to the growing number of connected devices and exchanged data, traditional threat detection is becoming obsolete, and there is a need to introduce ML based solutions, this way ML turns out to be a powerful solution for addressing security challenges in 5G [9].

2.2 6G Networks

This section will address a vision of the 6G networks with an emphasis on security issues. An overview of 6G networks and their improvements over 5G networks will be provided. Similarly, the importance of security in the context of 6G, the notion of openness, and the idea of ML as a major facilitator of 6G functionalities and security, should all be understood.

2.2.1 Overview

The introduction of 5G was a major step forward in building a high-performance network, completely changing how we engage with the digital world. With the advent of NS, 5G networks open the door to supporting enormous numbers of end devices and many logical networks [3]. Despite all of these

advancements, 5G networks might not be able to keep up with the future demands [16], once it is expected that a large volumes of data will be produced as the actual world gradually transitions to full digitization [3]. 5G networks won't be able to offer an entirely automated, intelligent network [17], once it is expected that mobile communications will be far more prevalent in our daily lives than they are now. So, the next generation of mobile networks, or 6G, will enable us to address the issues that may arise in 2030 and beyond [3].

While 6G represents an evolution in mobile technology, it inherits many security challenges from 5G. Issues such as vulnerabilities in network architecture, data privacy concerns, and the complexities of managing numerous connected devices will not only persist but are expected to intensify with the rollout of 6G. As the number and diversity of connected devices increase, alongside the anticipated full adoption of open network concepts and NS related technologies, the potential attack vectors will also expand, resulting in a more complex security landscape.

6G's development is being driven by the growing need for advanced wireless connectivity that can accommodate a range of changing needs for new services and applications [18]. This requirement is brought on by the exponential increase in mobile traffic and subscriptions, which intensifies the demand for ongoing network efficiency improvements. Simultaneously, the 6G system aims to fulfill the requirements of existing services while creating opportunities for disruptive innovations [19].

6G is positioned to offer a significant increase in coverage, peak data rate, user experience rate, system capacity, and connectivity density, while meeting strict criteria in latency, dependability, mobility, and security [19]. 6G's global reach is a key focus, with goals including increased intelligence, security, and resilience along with improved spectral, energy, and cost efficiency [13]. Some of the performance requirements and application areas of 6G are shown in Figure 4.

Additionally, in order to increase network management and automation, 6G will need to integrate Artificial Intelligence (AI) and ML technologies, this will allow for the dynamic coordination of networking, caching, and computing resources, which will boost the 6G network systems [13].

In this way, 6G, as the next generation of wireless communication technologies, is crucial for addressing advanced security and privacy challenges, meeting the needs of emerging technologies, and supporting a constantly evolving application environment. However, it is vital to critically analyze the security challenges that 6G will face, as these challenges — many of which stem from its predecessor, 5G — must be addressed to ensure the robustness and reliability of future networks [21].

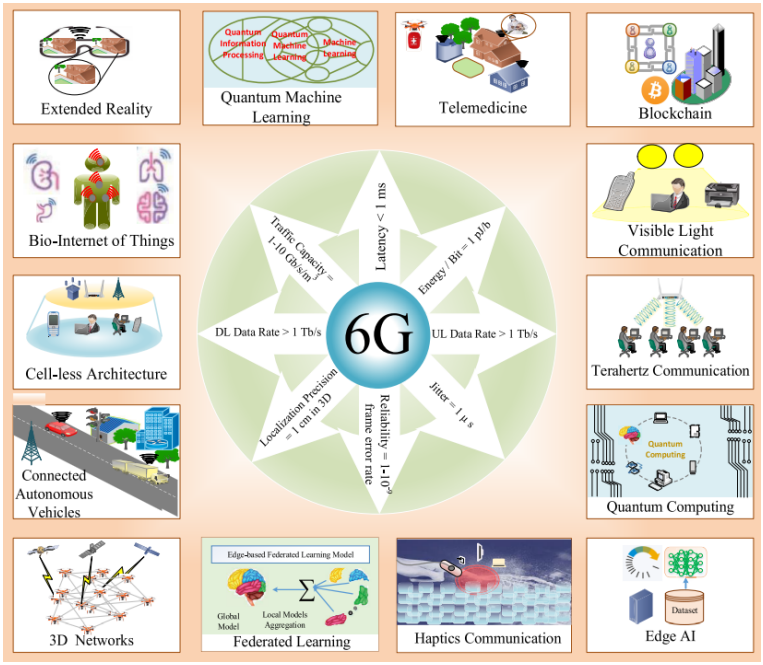


Figure 4: Expected requirement and application of 6G networks [20].

2.2.2 The Need for Security

The introduction of 6G technology presents until now unprecedented challenges, especially in the field of security. As noted by Bernardos et al. [3], ultrahigh levels of security are required to maintain trust and data privacy given the massive volume of data handled by 6G systems across a variety of critical applications. Innovative approaches like slicing and zero-touch micro-segmentation are needed to overcome these challenges. Because of the wide range of applications that 6G will offer, they demand sophisticated network and security requirements, specially with the increase of skilled attackers and malicious activity [21]. Figure 5 gives a vision of the security threat panorama in 6G.

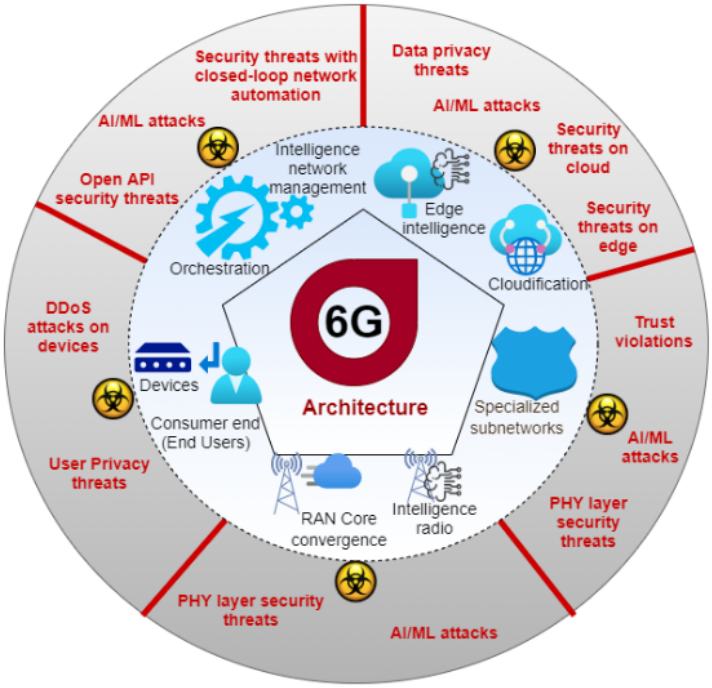


Figure 5: 6G security threat panorama [21].

According to the European Vision of 6G [3], 6G will pass to a multi-vendor paradigm that can potentially increase the threat surface for malicious attacks. The integration of open networks concepts, such as ORAN[2], in 6G networks, along with the disaggregation and proliferation of standardized and open interfaces, contribute for the need of higher levels of security. The openness of 6G networks will be discussed further.

Furthermore, NS was suggested in 5G as a crucial networking technology enabler [20], but its full implementation is anticipated in 6G. More sophisticated NS techniques in 6G might potentially expose the network to new threats [22].

Given this, it is expected that security challenges associated with 6G systems will be more complex than those affecting current 5G systems [19]. In today's linked world, maintaining privacy and security is still a major concern [23]. Thus, the development of innovative and all-encompassing security and privacy solutions is required for communication networks.

2.2.3 Openness

The landscape of 6G networks is rapidly changing, and one prevalent idea that has emerged is openness. This common vision for the innovative network architecture highlights the necessity for new interfaces,

[2] https://www.o-ran.org/

allowing a wider range of hardware and software providers to enter the telecom market [18].

In line with the vision for openness, and according to Zhou et al. [24], openness at both the network architecture and interface levels, will turn 6G a more flexible and intelligent network. On the one hand, a crucial idea in the network architecture's openness is NS, it enables service providers and vertical industries to quickly launch new services, on top of the same common physical infrastructure, enabling the decoupling of hardware and software. On the other hand, open interfaces are essential to vendor connectivity, teamwork, and a strong supplier ecosystem. The ownership of physical infrastructure is expanding beyond mobile providers as 6G connects with vertical sectors. Smooth interoperability requires open interface designs and standardization.

Despite the many perks, the 6G ecosystem becomes more complex and risky due to the integration of multi-vendor and open source software [2]. While the openness of the network in 6G enhances flexibility in network management and resource utilization, it amplifies the prominence of security issues, necessitating robust measures to safeguard against potential threats and vulnerabilities in the evolving network landscape [24].

In essence, the 6G network architecture's pursuit of openness promises a revolutionary change in the telecom sector that encourages cooperation and creativity. But as the network becomes more capable, maintaining a careful balance between security and flexibility is crucial to preserving the robustness and dependability of 6G networks.

2.2.4 The Role of ML

ML turns out to be a major force behind the development of 6G networks, being essential to many facets of network functionality and security [20]. Network management and intelligent orchestration are intimately related to 6G, as a result, AI/ML plays a crucial role in the 6G paradigm [25].

According to the European Vision for 6G Networks [3], AI/ML will be used in a wide range of areas, for example, to automate processes and network functions, to achieve a zero-touch approach, and also for the optimization of the physical layer. We can therefore assume that AI/ML will play a major role in the development of the next generation of 6G mobile networks. [25].

Furthermore, higher peak rates and the expected massive volume of data generated in the 6G networks, will encourage the integration of AI/ML in the 6G network security design [13], thus the use of ML techniques becomes imperative to achieve security.

ML can be used as part of an intelligent and flexible security mechanism, capable of predicting, detecting, containing, mitigating, and preventing threats and active attacks, thereby limiting the spread of

vulnerabilities [21, 22], having a crucial role in the detection of new types of attacks [2].

The vision for 6G security, highlights the necessity for security automation by emphasizing the integration of AI, especially ML [21]. The merge of ML with concepts like virtualization and security function softwarization, plays a vital role in the automated security [17], strengthening security measures across the entire network infrastructure, including E2E network security [22].

In summarized form, ML is a key component of the security architecture of 6G networks. It enables the automation of complex tasks, such as intelligent orchestration and autonomous adaptation of security systems [2]. Helps to provide automated security, flexible defenses, and creative solutions to deal with the constantly changing security issues in these next-generation networks. Ensuring security is essential to making the 6G vision a reality. Intelligent and trustworthy security solutions are offered by AI-enabled network security [25].

Regardless of all the promising applications of AI in 6G networks, there are some issues that should be addressed. Siriwardhana et al. [26] talks about four issues areas: security, privacy, ethical and the use of AI to launch intelligent attacks.

- **Security** - Specially ML systems face security threats like poisoning attacks, evasion attacks, and API-based attacks.
- **Privacy** - AI's large-scale data analysis and automation needs in 6G networks can compromise privacy. Insecure IoT devices and model inversion attacks on ML can target data theft and privacy violations, making it crucial to protect user data.
- **Ethical** - AI in 6G networks reduces the need for human intervention, yet computers are not as ethically conscious as humans. Although AI systems can operate in accordance with their training, unlike humans, they are not capable of acting against logic in some circumstances.
- **Intelligent Attacks** - AI can be used to identify patterns in large data volumes, potentially exposing network vulnerabilities

Without doubt, AI/ML will revolutionize the future of networks, however it is necessary to have in mind the challenges it can bring.

Chapter 3

Security in 5G Networks

This chapter delves into the description of Intrusion Detection Systems (IDS). It begins by outlining the fundamental concepts of security, followed by an examination of the various IDS technologies. Additionally, the chapter explores the integration of ML techniques in IDS to enhance detection accuracy and response times. It further reviews related work in the field, showcasing existing research and methodologies for implementing ML-based IDS in 5G networks. The goal of this comprehensive review is to provide readers an accurate understanding of the state-of-the-art in IDS for 5G networks.

3.1 Security Framing Concepts

This section aims to provide some explanation and definition of some important concepts that will be crucial for the understanding of the next sections and chapters.

3.1.1 Vulnerabilities

A vulnerability in an Information Technology (IT) system is a weakness that attackers can exploit to carry out successful attacks. Vulnerabilities can come from flaws in design or implementation, misuse of intended features, or user errors [27]. Next, it is presented some kinds of vulnerabilities:

- **Flaws** - unintended functionalities resulting from design or implementation mistakes, and they may go undetected for a significant period.
- **Zero-day vulnerabilities** - vulnerabilities discovered before being mitigated. Attackers actively sought after these kind of vulnerabilities and exploited them, posing big a risk to systems.
- **Features** - intended functionalities that can be misused by attackers. Although features can improve user's experience or system's efficiency, they can be wrongly exploited.

- **User errors** - such as choosing weak passwords or leaving devices unattended. Users are a great source of vulnerabilities, turning well-designed systems insecure.

3.1.2 Anomalies

Any deviation from the established regular communication patterns inside a network is referred to as an anomaly in the context of network communication. On one hand, the anomalies could be deliberate interruptions meant to jeopardize the security of the network, such as malware invasions and cyberattacks. On the other hand, anomalies may result from technical issues with the network architecture, such corrupted data packets or changes in communication patterns brought on by equipment malfunctions, capacity constraints, or network issues [28].

3.1.3 Attacks

Cisco defines a cyberattack as a "malicious and deliberate attempt by an individual or organization to breach the information system of another individual or organization. Usually, the attacker seeks some type of benefit from disrupting the victim's network" [29]. In other words, we can classify a cyberattack as any intentional and unauthorized activity on a network, computer service or digital device that aims to breach its security, alter its operations, services and access confidential information, in order to extort money from the victims or stop the service. According to Cisco, the most common cyberattacks are [29]:

- **Denial-of-Service (DoS)** - DoS attacks are a tactic used by adversaries to interfere with expected device or network functionality. DoS attacks involve, for example, sending a request that the target device is unable to handle or flooding it with a large number of requests in a brief amount of time. The disrupted target may become unresponsive for a while, maybe until it can be rebooted [30]. Sometimes attackers can exploit vulnerabilities to perform a DoS, or they can do it by exhausting bandwidth, router processing capacity or network resources (network/transport-level), or even by exhausting the server resources (e.g., sockets, memory, disk/database bandwidth, etc.) [31]. Nowadays, it is common attackers to use more than one source to execute the attack, being called a DDoS.

- **Man-in-the-Middle (MITM)** - MITM is a kind of attack where a malicious actor secretly takes over the communication channel between two or more endpoints. The attacker has the capability to intercept, modify, alter, or substitute the communication traffic exchanged by the victims, setting

it apart from a mere eavesdropper. Victims remain ignorant to the presence of the intruder, believing that the communication channel is secure [32]. MITM attack aims to compromise:

  - **Confidentiality** - spying on the communication.
  - **Integrity** - intercepting the communication and manipulating messages.
  - **Availability** - destroying messages or modifying messages to cause one of the parties to cease communication.

- **Domain Name System (DNS) Tunneling** - DNS tunneling is a technique that uses the DNS protocol to transmit other types of data. Typically, DNS traffic is not blocked by network firewalls, allowing attackers to exploit this method to exfiltrate data from systems or establish remote control over them [33].

- **Malware** - The term "malware" describes a wide range of malicious software, such as ransomware, worms, trojans, spyware, bots, rootkits, and viruses. These programs are made to fulfill the harmful goal of attackers who seek to obtain sensitive personal data without authorization, interfere with system functions, and access computer systems and networks. Malware can also lead to overwhelm processes and affect system performance. Spyware is a type of malware that hides itself, steals important data from computers, and transmits it to attackers [34]. Ransomware is a malware that has becoming very popular in recent years, it encrypts victim's files and demands that a ransom is paid for the file's decryption.

- **Phishing** - The foundation of a phishing attack is social engineering, in which cybercriminals fabricate a fake communication that seems authentic and originated from a reliable source. Attackers deceive individuals into performing actions like installing malware, visiting a compromised website, or disclosing login credentials in order to steal money, critical data, credit card info, etc. They accomplish this by sending seemingly harmless emails or texts and appealing for humans emotions like fear and curiosity [35].

- **Structured Query Language (SQL) Injection** - In a SQL injection attack, the attacker attempts to manipulate SQL statements used by a web application. This can succeed due to inadequate input validation and incorrect SQL statement composition. Web applications frequently use database systems to provide backend functionality. User input is frequently used to dynamically generate SQL statements that communicate with databases in support of online applications. So, attackers try to pervert the application's original goal by sending SQL queries directly to the backend

database. The consequences of a successful SQL injection attack might be extensive, depending on the online application and how it handles the data supplied by the attacker before constructing a SQL statement [36].

- **Zero-day Exploit** - A zero-day exploit is a cyberattack method that capitalizes on an undisclosed security flaw in computer software, hardware, or firmware. The term 'zero day' indicates that the vendor has zero days to address the flaw, allowing malicious actors to exploit it immediately before a fix is available. This vulnerability may go unnoticed for an extended period until someone discovers it, either security researchers or malicious hackers. Once identified, the vulnerability becomes public knowledge, prompting a race between security professionals developing a fix and hackers creating a zero-day exploit to exploit the vulnerability [37].

3.2 Attack Detection Systems

As previously highlighted, one of the biggest concerns in today's world is cybersecurity owing to the increasing amount of digital technology being integrated into society. Because of the interdependence and reliance on digital infrastructures, cybersecurity is essential to the protection of information and systems. Establishing robust mechanisms for the rapid detection and prevention of malicious activities is crucial as cyber threats grow increasingly sophisticated. Putting in place efficient attack detection systems is the key to solving cybersecurity issues.

Emphasizing the possible consequences of any malicious infiltration or attack on computers, information systems, or network vulnerabilities is crucial. These kinds of events have the potential to trigger major catastrophes and, more importantly, they are violate the fundamental principles of computer security policy, which are represented by the Confidentiality, Integrity, and Availability (CIA) [38]. Thus, these systems are essential to maintaining these principles. This section explores the complex field of attack detection, illuminating some widely adopted strategies.Given the abundance of research and information available about Intrusion Detection Systems (IDS), these systems are emphasized throughout this section.

3.2.1 Intrusion Detection Systems

IDS become vital protectors in the never-ending war against cyberattacks, constantly searching for, evaluating, and identifying unwanted activity occurring within information systems. The main objective of IDS is to detect a range of security breaches, including external intrusions — attacks originating outside the organization — and internal intrusions, which stem from threats within the organizational perimeter [39].

An IDS is an early warning system that combines various tools, techniques, and resources to identify possible intrusions before they have a chance to compromise the security of critical system components [40].

The creation of advanced IDS becomes more important as the digital environment changes and the threat landscape gets more complex. The incorporation of ML techniques has been crucial in improving intrusion detection capabilities in recent decades, enabling IDS to respond more effectively and adapt to new cyber threats [41].

IDS can be divided in two major types, based on their scope of monitoring and the location at which they operate within network:

- **Host Intrusion Detection Systems (HIDS) -** Specifically focused on the security of a single host or device, it is a cybersecurity technique created to protect individual host computers by the monitoring and analysis of file and process activity within their software environment [39]. HIDS actively monitors incoming and outgoing traffic for a single host by living on it [42]. It seeks to detect and alert of any unusual or malicious activity that can compromise the host machine's availability, integrity, or confidentiality.

- **Network Intrusion Detection Systems (NIDS) -** NIDS is a cybersecurity solution that constantly monitors network traffic in order to detect and prevent intrusions [39]. It continuously processes and examines the packets traveling across a certain network link while operating at specified locations within a network architecture [43, 42]. NIDS improves network security by monitoring network traffic and instantly identifying malicious activity and security issues.

3.2.2 Detection methodology

Detection methods can be broadly classified into three types: anomaly-based, misuse-based, and hybrid. Each type of approach has its own advantages in detecting and combating security threats. Each one of these methods will now be discussed.

Anomaly Detection

As defined in [39], the basic principle of anomaly-based cybersecurity solutions is modeling normal network and system activity and identifying anomalies as deviations from established patterns. This strategy is highly desirable, because it is effective in identifying zero-day attacks and provide a proactive defense against unknown threats [40, 39].

Other advantages of this strategy are spotted by Buczak e Guven [39]: one is that typical activity profiles can be adapted per system, per application, or per network, which makes difficult for attackers to anticipate which actions might go unnoticed; other is that, the data that triggers alarms in anomaly-based techniques— which are frequently linked to new attacks— can be utilized to establish signatures for misuse detectors, thus improving the system/network security.

Although, a significant disadvantage is the possibility of high false alarm rates, as normal but until undetected system behaviors could be mistakenly classified as anomalies [39].

Within the realm of anomaly detection, it can be divided in two types of analysis:

- **Static Behavior Analysis:** This kind of analysis relies on the idea that the system being monitored has a static component that remains constant. Variations from the initial static configuration are marked as errors, suggesting that an unauthorized party may have accessed or altered the system [43].

- **Dynamic Behavior Analysis:** Dynamic anomaly detection utilizes audit records or monitored network traffic data to adjust to alterations in dynamic system behavior. This method provides real-time analysis of network traffic deviations from usual patterns by concentrating on events of interest that are captured in audit logs [43].

The ongoing development of anomaly detection techniques is highlighted by recent developments, such as the introduction of AI-based techniques that help distinguish anomalies in network traffic [44].

Misuse Detection

Misuse detection, also known as signature-based detection, is a cybersecurity technique designed that uses relies on the use of predefined signatures or profiles associated with previously known attacks that serve as reference points to identify and categorize potential threats based on recognized patterns [40]. This approach is especially effective at identifying known attack types without producing an excessive amount of false alarms [39].

By comparing current activities to a database of known attack scenarios, the misuse detection operational principle enables the system to identify and flag actions that correspond with recognized attack signatures. In this sense, the efficacy of misuse detection depends on a frequently updated database with rules and signatures of known attacks, suggesting that manual updates are necessary on a regular basis to keep the system up to date with the most recent threats [39].

The incapacity of misuse detection to identify new or zero-day attacks — those not included in its knowledge base — is an obvious drawback. Misuse detection is excellent at quickly and precisely recognizing existing attacks, but it has difficulty recognizing novel attack types that haven't been previously analyzed. Due to this limitation, it may not be possible to identify new threats that aren't included in the signature database [43, 40].

For example, the system may know the signature for a brute force password attack defined as "three failed login attempts within five minutes" or the signature of a known DoS attack, however, any modifications to the brute force password attack or the DoS attack may go undetected, underscoring the method's reliance on predefined signatures for recognition [40, 44].

Table 1 synthesizes the major differences between anomaly detection and misuse detection.

Table 1: Comparison of anomaly and misuse detection.

Detection Method	**Advantages**	**Disadvantages**
Anomaly Detection	• Able to identify unknown threats • Able to detect zero-day attacks • Can create attack signatures	• High false alarms • Hard to trace a normal behavior profile • Needs initial training
Misuse Detection	• Simple implementation • Minimum false alarms • Better for detecting known attacks	• Needs a database with attacks signatures • Constant update of the database • Unable to detect unknown threats and zero-day attacks

Hybrid Detection

Cybersecurity hybrid techniques combine anomaly detection methods with misuse techniques to enhance overall intrusion/attack detection. Their combined purposes are to lower False Positive Rate (FPR) associated with unknown threats and increase the detection rates of known attacks. Hybrid strategies allow a flexible defense against a variety of potential attacks by integrating the benefits of both approaches, providing a complete and efficient cybersecurity solution [39].

3.2.3 ML-Based Detection

The number of applications handled by network nodes and the size of networks have significantly increased in recent years, which creates enormous volumes of vital data that are exchanged throughout many nodes, putting in risk both the data and the nodes. So, researchers underline the importance for automated security techniques due to the dynamic nature of cyber threats [45]. Using ML techniques to identify new and unknown cyber threats is one potential strategy. ML is a valuable tool for addressing the challenges presented by modern cyber threats and an effective way to identify zero-day attacks due to its capacity to learn from past events and adapt to evolving attacks [45, 46].

Traditional IDS, in spite of decades of improvement, continue to face ongoing challenges in improving detection accuracy, reducing false alarm rates, and successfully recognizing new threats [47, 45].

One potential solution to get around the drawbacks of traditional IDS is to use ML techniques. With its ability to accurately and automatically distinguish between normal and abnormal data, ML techniques demonstrate a level of adaptability that is critical in the face of constantly changing cyber threats, and give IDS the capacity to identify unknown attacks, acting as a proactive protection against new and developing threats [47].

The capacity of ML-based detection systems to continuously learn and adapt is one of their main advantages. These systems are capable of successfully recognize both known attack variants and completely unidentified cyberthreats. This flexibility is especially important because attack frequency and sophistication are only going to rise.

Additionally, Khraisat et al. [41] highlights that ML-based IDS are capable of independently identifying trends and abnormalities in network traffic data, they help reduce the need for manual intervention. The move towards automation not only simplifies the detection process but also makes it possible for security systems to keep up with the volume of data generated and shared among network nodes.

Attack detection techniques are being revolutionized by ML, which provides increased speed, accuracy,

and flexibility while reducing the false alarm rate in response to evolving cyberthreats. With the rapid advancement of technology and with the emergence of 6G, network security must incorporate ML for strong defense against attacks.

Next, it will be described some of the ML algorithms and techniques that are being applied in the context of attack detection.

Supervised Learning

In supervised learning, a model is trained using labeled data and then tested using unlabeled data. The first steps in the process are gathering the dataset, dividing it into training and testing sets, preprocessing the sets, extracting features, and then putting the model into an algorithm to train it to identify the features linked to each label. After receiving test data, the model converts input data into output data based on a sample of input-output pairings. In short, the ML algorithms that require external aid are known as supervised algorithms [48]. There are two types of supervised learning: classification and regression.

Some of the most used supervised algorithms are:

- **Decision Tree** - A decision tree is a tree-shaped graphical representation of choices and their outcomes [48]. The structure of the tree is composed of a root, decision nodes (representing a feature), branches (the possible values for that attribute) and leaf nodes (final class - decision) [49]. Decision trees are mostly used in ML for classification tasks; the decision process goes in a sequential way from the root node to the leaf node, meanwhile features are evaluated and one branch is selected [50, 51]. Decision trees are known for being easy to use, having a straightforward prediction procedure, and being effective with unnormalized datasets [51]. However, this technique has a high computational cost [49]. Figure 6 gives a graphical example of a decision tree.

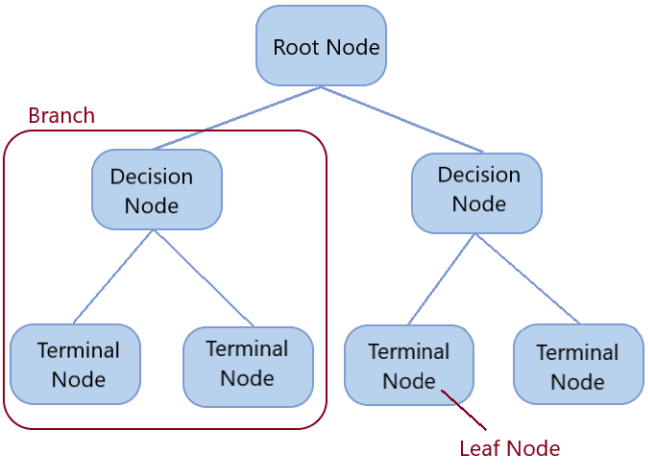


Figure 6: Graphical representation of Decision Tree algorithm.

- **Support Vector Machine (SVM)** [52, 49] - It is a method used for solving two-class problems, where the data can be separated by a hyperplane defined by support vectors, they are crucial for setting the boundary between the two classes. The space on either side of the hyperplane separating the two classes is like a margin. The goal is to reduce generalization errors by maximizing this margin, creating the widest possible distance between the separating hyperplane and the instances on either side of it. A key component of SVM is its ability to employ a variety of kernels, including Gaussian, Polynomial, and Linear ones. These kernels provide for flexibility in the mapping of data into different feature spaces. Figure 7 gives a simple visual representation of how SVM works separating two classes of data "A" and "B", using a linear approach.

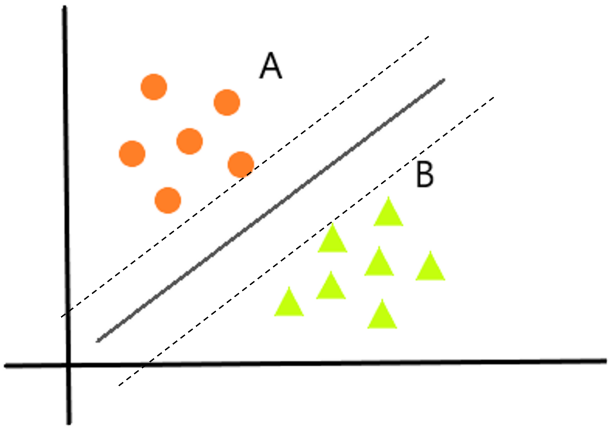


Figure 7: Graphical representation of SVM algorithm.

- **Random Forest** [53] - A Random Forest is an ensemble learning method used for classification and regression tasks. It works by building multiple decision trees during training and merging their results for a more accurate and stable prediction. The process begins with data sampling, where random subsets of the training data are created using a method called bootstrapping. For each subset, a decision tree is constructed. Unlike traditional decision trees, each node in these trees considers a random subset of features when splitting the data. For classification tasks, each tree votes for a class, and the most common class is chosen. For regression tasks, the average of the predictions from all the trees is taken. Figure 8 provides a visual representation of how this algorithm classifies a dataset instance.

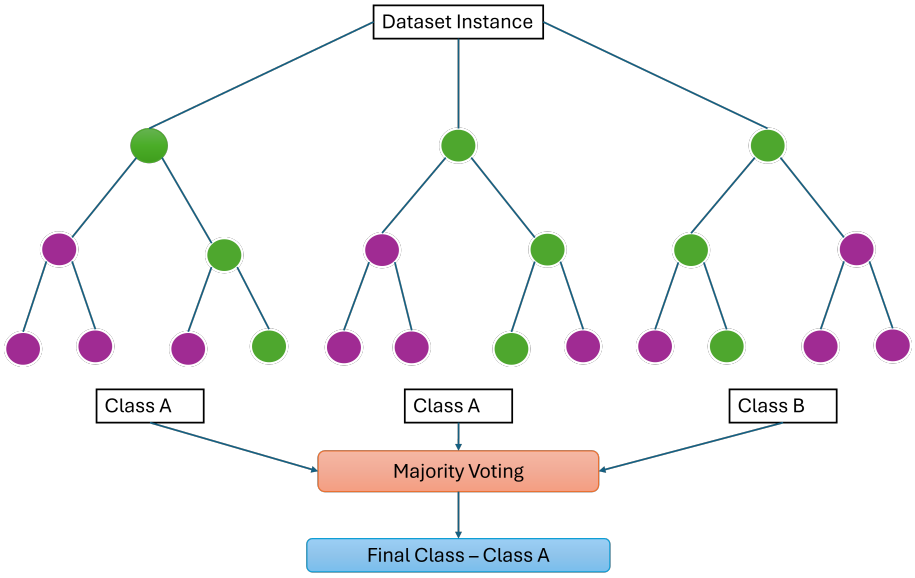


Figure 8: Graphical representation of Random Forest algorithm.

Unsupervised Learning

In unsupervised learning the machine simply receives inputs but it doesn't obtains supervised target outputs [54], in other words data instances are unlabeled. When new data is introduced, the previously learnt features are used to identify the data's class [48]. A very used technique for this type of learning is clustering, and a very used algorithm is K-means.

Deep Learning

As cyber threats get more complex, traditional ML techniques are becoming incapable to detect threats and attacks efficiently [55]. ML methodologies have difficult to handle the increasing security concerns due to the the introduction of new technologies, increased network traffic, and the production of large-scale

and multi-dimensional data, as well as the sophistication of attack scenarios [56]. This way, researchers are exploring more and more the use of Deep Learning (DL) in these systems, to suppress the limitations of traditional ML.

DL is a branch of ML and refers to a class of Artificial Neural Network (ANN) specifically designed to handle large-scale and high-dimensional datasets effectively. With such data, classic ML techniques find it difficult to retain efficacy and accuracy; in this situation, DL algorithms provide strong alternatives [57].

Put simply, ANNs are computational models that learn patterns and relationships within data by utilizing interconnected layers of artificial neurons. These models are inspired by the form and operation of biological neural networks [58]. Among many others, a widely used type of ANN is Convolutional Neural Network (CNN). CNNs [59] are designed especially to analyze visual data, such as pictures. Its design imitates the way the human visual system works by dividing images into smaller, easier to understand pieces and teaching the user to recognize patterns within them. CNNs are capable of creating increasingly complex representations of the visual world through the use of a technique called convolution, which involves swiping small filters across the image to extract features like edges, shapes, and textures. This technique allows CNNs to perform tasks like object recognition, image classification, and even image generation. Figure 9 shows hows CNNs can classify an image.

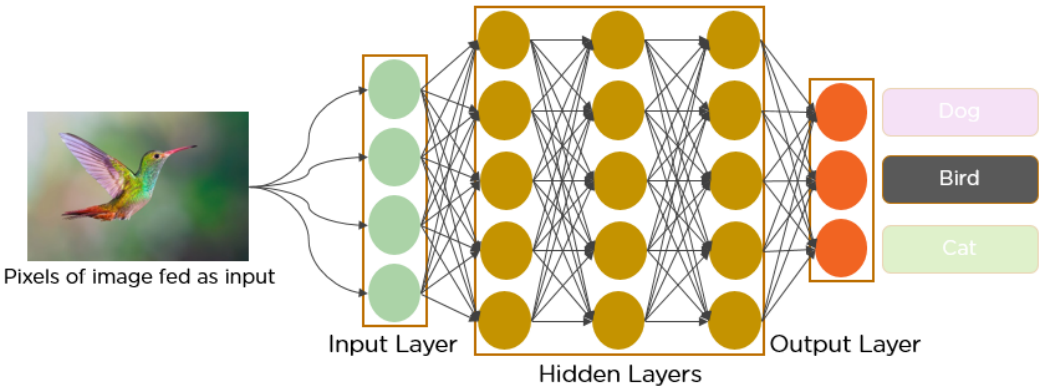


Figure 9: CNN image classification representation [60].

3.2.4 Challenges

In this section, it will be examined the challenges that attack detection techniques encounter, focusing on intrusion detection. These challenges highlight the complexity of this techniques, from handling False Positive (FP) and False Negative (FN) to coping with the ever-changing nature of cyber threats. By addressing these issues, we can better understand the continuous work needed to strengthen defenses against a constantly changing set of security threats.

3.2.4.1 False Alarm Rate

This kind of challenge is mainly seen in anomaly detection based systems, as previously referenced. It happens because anomaly detection assumes that intrusive behavior is anomalous. It is reasonable to expect that a significant portion of attacks would result in anomalous behavior. This method, however, has the potential to mistakenly classify a lot of harmless behavior as malicious just because it deviates from the norm.

Finding the ideal balance between FP and FN is essential to the efficacy of intrusion and attack detection. FN happen when actual malicious activity is not detected, and FP happen when harmless activity is inadvertently reported as malicious. To guarantee a trustworthy and accurate IDS, security administrators frequently have to choose between reducing FP and FN [38]. The difficult part of system design is minimizing both kinds of faults while taking high security precautions during implementation [61].

3.2.4.2 Resource Intensiveness

Keeping profiles that specify typical behavior for all relevant entities and resources—such as users, apps, files, and systems—is essential for anomaly-based detection systems.

Adeleke [62] addresses the problem of computer overhead in intrusion/attacks detection, being a problem that affects mainly systems based in anomaly detection. In the article it is explained that, in order to compare recent activity sets with anticipated usage models, systems must also monitor current activity and analyze it using appropriate algorithms, and have to undertake regular recalculations to generate new models as usual behavior changes. Also, it is underlined that the majority of ML techniques for anomaly detection are neural network-based, which have long processing times for the initial training set of data, calling for enough processing and storage capacity. When balancing security and computational overhead, designers must take into account variables such as the frequency of recomputation, computing and storage capacity, and comparison time resolutions.

However, in order to keep all of the attack signatures, signature-based detection also requires a large amount of storage capacity; the more signatures the system is aware of, the more effective it will be.

To give a specific example, IDS can be used to reduce attacks and threats in edge computing; however, because edge nodes have limited resources (e.g., in terms of processing and storage capacity), it can be difficult to allocate resources within an IDS in an effective and fair manner [63]. The increasing popularity of edge computing, being one of the enabling technologies of 6G [13], highlights the importance of addressing these issues and emphasizes the requirement for IDS solutions that are flexible enough to accommodate the resource limitations that come with this computing paradigm, or in other systems.

3.2.4.3 Complexity in Network Environments

The growing number of devices connected to the internet emphasizes how important it is for detection systems to handle data efficiently. As mentioned in earlier sections, 5G is capable of supporting a large number of connected devices and services; as traffic volume increases, 6G will be even more capable of supporting them. Thus, the issue of creating efficient and scalable solutions is made more challenging by the constantly shifting network dynamics, which necessitate adaptive attack detection systems to counter emerging intrusion and attack techniques [64].

In addressing the particular context of Internet of Things (IoT) systems, for example, [65] highlights the significant growth in devices as compared to traditional systems. Due to the constraints of current centralized techniques, this surge requires scalable alternatives.

So, overcoming this challenge is very important to ensure that new strategies align with the intricate demands of the new generation of networks, namely 5G and 6G.

3.2.4.4 Zero-Day Attacks

A common problem with signature-based detection is that it is difficult to identify previously unknown attacks, which prevents it from identifying new threats. However, because new vulnerabilities and exploits surface on a daily basis, a successful anomaly detection system needs to be able to adapt to unknown attacks in order to be able to recognize threats without well-established signatures [64].

Although anomaly-based detection performs better at identifying zero-day attacks, it still struggles to detect carefully planned zero-day attacks that fit into expected usage models [62].

IDSs' ability to adapt is essential for effectively fighting against a constantly evolving variety of cyber-security threats.

3.2.4.5 Privacy Concerns

In the realm of IDS, the issue of privacy becomes a critical concern for a long time. As outlined in [66], when log files, often containing personal and sensitive information, are employed for auditing events, the personal integrity of users is at risk.

In recent times, privacy legislation across the world has highlighted the importance of protecting citizens' right to privacy, as clarified on in [67]. Several countries have passed legislation related to data protection and privacy, such as the United States with HIPAA [68], the European Union with its data protection directive [69], and Canada with PIPEDEA [70]. These laws protect the privacy and confidentiality of personal data, which emphasizes how important it is for IDS to comply to strict privacy standards in

order to uphold users' rights and confidence.

3.3 Related Work

Once the theoretical foundations and concepts have been established, it is essential to investigate the suggested approaches and technologies for attack detection. Several studies have presented novel techniques designed especially for 5G and NS scenarios. Furthermore, a number of European projects have promoted cooperation and innovation in the field of attack detection.

This section attempts to present some of the most recent attack detection systems that have been proposed, with an emphasis on those that use ML to safeguard 5G networks, specially NS function. Additionally, emphasis will be placed on the DDoS detection systems.

Thantharate et al. [71] proposed a framework, *Secure5G*, for securing NS function in 5G. The proposed framework is a NS model based on DL CNN and is intended to proactively identify and remove risks based on incoming connections before they infiltrate the 5G core network. Detecting and mitigating DDoS attacks, analyzing traffic patterns, predicting future traffic, allocating resources to the most suitable slice, and identifying unauthorized operations via User Equipment (UE) are the goals of the framework. The model can be used to forecast capacity and changes over time. Additionally, it maintains all of the original and previous connection requests made by any device thanks to an integrated database of devices and user habits from learning. They also introduce a new concept "Quarantine Slice", as a form of attack mitigation, consisting in a slice with bare minimum Quality of Service (QoS) and strict requirements. *Secure5G* is an extension of the *DeepSlice* [72] research work. The total performance was evaluated using volume-based flooding and spoofing attack scenarios, and the detection accuracy was more than 98%. Some future work will include the model training in real-time. The overview of this model is described in Figure 10.

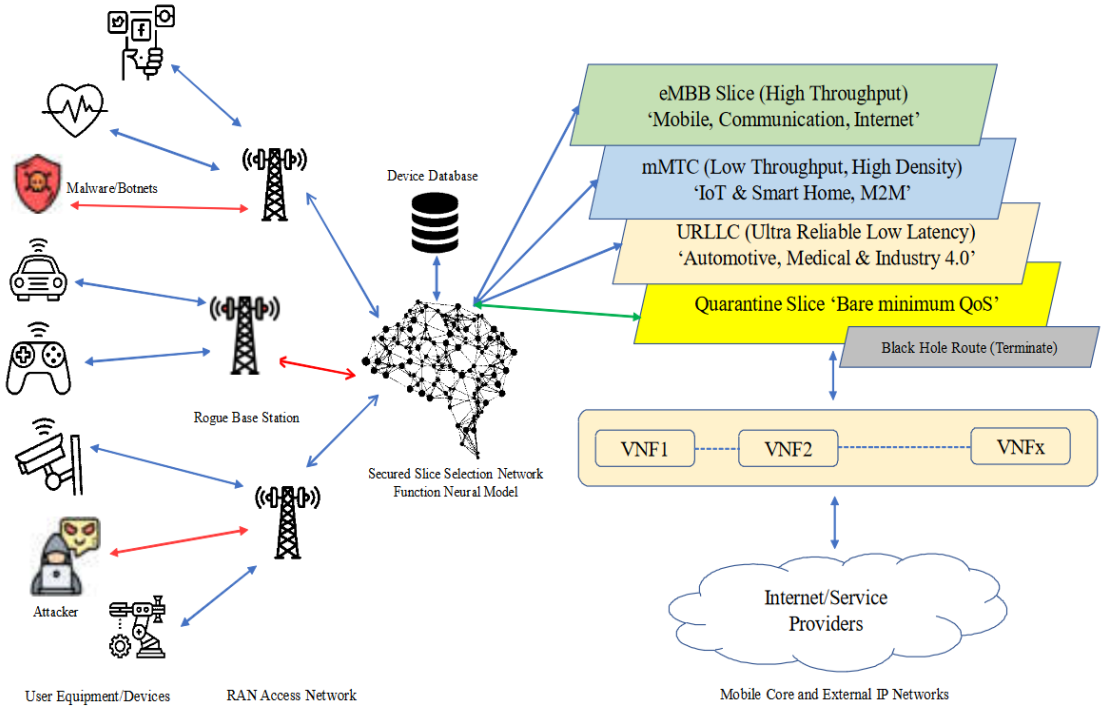


Figure 10: *Secure5G* model overview [71].

The *DeepSecure* framework, proposed by Kuadey et al. [73] is represented in Figure 11. It is based on Long Short Term Memory (LSTM) DL technique and includes models for slice prediction and attack detection in 5G network slices. Based on LSTM, the attack detection model forecasts DDoS attacks from UE network traffic, while the slice prediction model predicts suitable slices for authorised UEs. The *Secure5G* [71], which was previously discussed, is quite similar to this framework. The CICDDoS 2019 [74] dataset was utilized to evaluate the framework . Training parameters for the attack detection and slice prediction models included learning rate, activation function, optimizer, and epochs. It's also crucial to highlight that Python 3.8 and TensorFlow 2.4 were used in the framework's implementation. That said, 99.970% detection accuracy was achieved, surpassing the *Secure5G* framework.

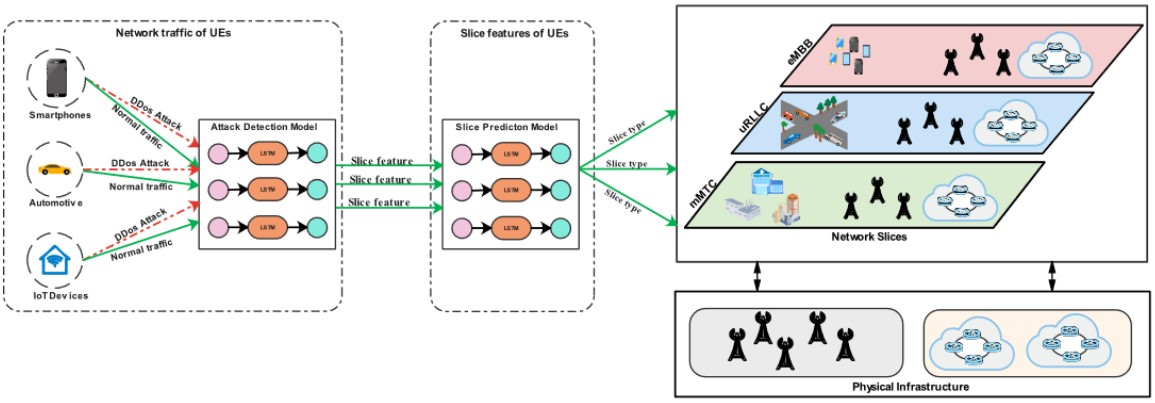


Figure 11: *DeepSecure* framework overview [73].

A two-fold method was presented by Hussain et al. [75] to identify botnet attacks in IoT environments. A botnet attack consists of two stages: scanning activity in the beginning and DDoS at the end. Two models are employed in this study: one to identify scanning activity and the other to identify DDoS attacks. Figure 12 gives an overview of the proposed approach. But, the focus here will be to explore the proposed model for DDoS attack detection. Four distinct datasets were used in the detection process, which employed the RestNet-18 model: DDoSLab, a self-generated dataset, CICIDS-19 [74], CICIDS-17 [76], and Bot-IoT [77]. Since the RestNet-18 model was created for picture classification, the authors suggested converting the three datasets into $15 \times 15 \times 1$ greyscale images. Additionally, the authors altered a few of the model's hyperparameters, including the learning rate, batch size, and epochs. The four models that were produced by using the four datasets to train the RestNet-18 were then compared. The model with highest result was the one trained and tested with the Bot-IoT [77] dataset, achieving an accuracy of 99.70% and a F1-score of 99.59%.

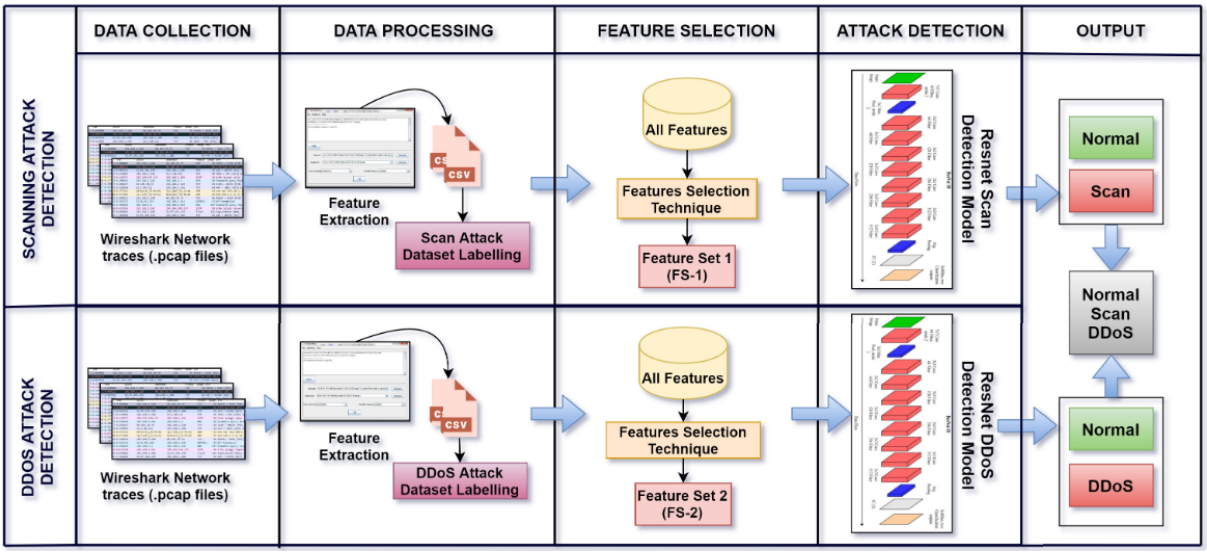


Figure 12: Approach to detect botnet attacks [75].

With an emphasis on DDoS attacks, Bousalem et al. [78] presented a 5G prototype for DL-based attack detection and mitigation in sliced networks. This prototype was developed in the context of the European project 5G-INSIGHT[1], that seeks to develop cutting-edge security features in 5G and beyond Vehicle To Everything (V2X) slicing, from attack detection to attack mitigation. This prototype makes use of a CNN-based DL model, which is implemented with Lightweight, Usable CNN in DDoS Detection (LUCID) [79], a "practical, lightweight DL DDoS detection solution" that classifies traffic flows as benign or malicious based on CNN features. Isolating malicious users inside a sinkhole-type slice with limited physical resources is how attacks are mitigated. Furthermore, according to the authors, the prototype can achieve an accuracy of nearly 97%.

Within the European project ASTRID[2], the aim was creating a cyber-security framework specifically for virtualized services. A DDoS detector component was created inside the complexity of this framework with the purpose of detecting DDoS attacks, as the name suggests, using conventional ML techniques. Sanchez et al. [80] conducted a study, in the scope of this project, to determine which ML algorithm was more effective in detecting DDoS attacks. They also used an exhaustive hyperparameter search to maximize their detection capacity. The datasets used were: ISCXIDS(2012) [81], CICIDS-17 [76] CSE-CIC-IDS (2018) [76] and CICIDDoS2019 [74]. The outcomes demonstrated that, using the most recent dataset, the Random Forest algorithm was able to achieve up to 99% detection accuracy. Additionally, the model's accuracy levels were compared to those of several DL techniques in this study.

Table 2 summarizes all the studies addressed in this section, highlighting the algorithm used, if the

[1] https://5g-insight.eu/

[2] https://www.astrid-project.eu/

proposed solutions were developed or tested for the NS environment, the performance metrics, a small description and the used dataset.

Table 2: Related work summary.

Framework	Algorithm	Network Slicing	Performance Metric	Description	Dataset
Secure5G [71]	CNN	✓	Detection Accuracy 98%	Identifies and neutralizes volume-based flooding and spoofing attacks.	Custom Dataset
DeepSecure [73]	LSTM	✓	Detection Accuracy 99.97% & F1-score 99.96%	Predicts slices and detects DDoS attacks within 5G networks.	CICDDoS2019 [74]
Two-fold method [75]	RestNet-18	✗	Detection Accuracy 99.70% & F1-score 99.59%	It identifies the two stages of a botnet attack (scaning and DDoS), in IoT environments.	Custom Dataset (DDoSLab)
5G Prototype [78]	LUCID [79]	✓	Detection Accuracy 97% & FPR <4%	Designed for detecting and mitigating DDoS attacks within V2X slices.	Custom Dataset
DDoS detector [80]	Random Forest	✗	Detection Accuracy 99.98% & F1-score 99.99%	Developed to detect DDoS attacks in virtualized services.	CICDDoS2019 [74]

Chapter 4

Architecture

This chapter provides a comprehensive overview of the architectural design of the developed system. It is structured into three main sections.

The first section introduces the SDA component, the primary focus of this dissertation. This section outlines the SDA's primary responsibilities. It also discusses the key design properties.

Following this, a contextualization of the project where the SDA component is situated, is given.

Finally, the third section contextualizes the reader with the ZSM architecture. This foundation is crucial, as it outlines the principles and framework within which the developed system operates.

Ultimately, this chapter aims to equip readers with a thorough understanding of the architectural design of the developed component and its critical contributions to enhancing cybersecurity measures.

4.1 Component Architecture

The SDA component serves as the analytical engine within the system where it's inserted, tasked with converting raw network data—such as packets crossing the network—into actionable insights. Its primary functions include detecting patterns, anomalies, and potential threats, as well as generating alerts when suspicious activities are identified. By leveraging ML as a sophisticated analytical technique, the SDA enhances ability of the system where it is integrated, to proactively respond to cybersecurity threats.

Its core features and responsibilities include:

- **Proactive Analysis:** The SDA can spot possible dangers before they result in damage by examining current data. This can be achieved by detecting patterns and trends in the data which can be indicators of consistent threats or system behaviors that need to be kept in check (such as a system vulnerability). AI/ML algorithms are used to detect complex patterns.
- **Anomaly Detection:** It is responsible for detecting anomalies – unusual behaviors or data points

that deviate from the norm and can indicate potential security threats. This may include any actions such as unauthorized access, suspicious network traffic, malicious user activity, and more. ML methods will be employed to find anomalies.

- **ML Models Ensemble:** The SDA also offers an ensemble approach for the detection and prediction of anomalies. This combines the use of multiple ML models to enhance the robustness and accuracy of the system.

- **Data Visualization:** To aid operators and other components of the system understand the security landscape, the SDA offers visual representations of data, highlighting key insights, threats and patterns in real-time. This shall speed up any particular action that is required to be performed manually in the system.

- **Security Report Generation:** It is important to create security reports regularly, to document, communicate and analyse the security state of the system, so the SDA generates some security reports. These reports can become important documentation for legal purposes. Also, they can detail information about security incidents, giving information about causes and impact.

This component can be divided into modules or sub-components, that are described in Table 3. Figure 13 shows a simplified view of the architecture of the SDA component.

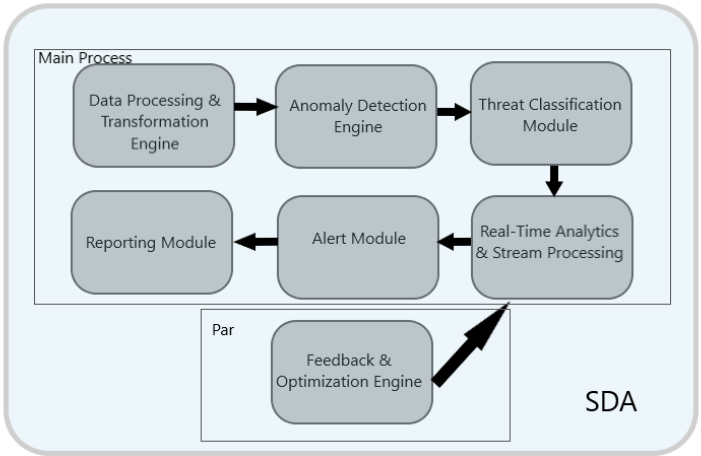


Figure 13: SDA's architecture.

Table 3: SDA modules description.

Modules	Purpose	Key Features	Interfaces
Data Processing & Transformation Engine	Prepares or reconstructs the gathered data.	- Broker consumer	**Input:** Collected Data from Security Data Collection component. **Output:** Analytics-ready data.
Anomaly Detection Engine	Detects anomalies in data.	- Statistical Analysis - DL Models - Alerts	**Input:** Analytics-ready data. **Output:** Identified anomalies, insights or alerts.
Threat Classification Module	Classifies the kind of the detected threat.	- ML Model - Classification	**Input:** Detected Threats. **Output:** Classified Threats.
Real-time Analytics & Stream Processing	It enables insights to be visualized on dashboards and facilitates data processing capabilities.	- Stream Process Engine - Real-time Dashboards	**Input:** Real-time insights and analyzed streams **Output:** Dashboards and processed data.
Alert Module	Sends alerts to the Security Decision component.	- Real-time alert - Alert Generation	**Input:** Processed data. **Output:** Alerts.
Reporting Module	Prepares analysed data to generate reports.	- Automated Report Generation	**Input:** Analytical results and insights. **Output:** Reports and processed data.
Feedback & Optimization Engine	It validates the performance of the ML models.	- Model performance checking	**Input:** Test data. **Output:** ML models performance.

4.1.1 Attack Detection Approach

Another noteworthy aspect of this component is the selected ML model for threat detection. A DL approach, known as LUCID [79], has been chosen for this purpose. LUCID, based on CNNS, is specifically designed to detect DDoS attacks. Choosing a model specific for detecting DDoS attacks is suitable because the test case proposed for this project will only involve DDoS attacks.

Developed as part of various European projects and used in many others, as stated in section 3.3, LUCID has proven to be an ideal choice for the project's objectives. Besides its proven high accuracy in detecting DDoS attacks, its architecture is suitable for deployment in resource-constrained environments, making it appropriate for edge computing scenarios, where devices possess limited computing capabilities. This aligns well with the scenario where this project will be implemented.

Also, this model seamlessly integrates traffic analysis functionality, streamlining dataset preparation, model training, and testing processes, making it easy to prepare and integrate within other systems.

The LUCID system operates through a series of interconnected functions and components designed to efficiently detect DDoS attacks in network traffic:

- **Network Traffic Preprocessing:** LUCID employs a preprocessing method for network traffic, which involves several key steps. Firstly, the algorithm extracts 11 attributes from each packet within a predefined time window. The packets belonging to the same bi-directional flow are then grouped together in chronological order, forming an array of shape [n, f], where n is the maximum number of packets collected for each flow within the time window, and f is the number of features. Each attribute value is normalized to a [0, 1] scale, and the samples are zero-padded to ensure a fixed length, which is necessary for input to a CNN. Finally, each example is labeled based on its flow identifier. This last step is applicable only during the training or testing phase, where a labeled dataset is used. In online detection, labels are not applied.

  This preprocessing method generates a spatial data representation that allows the CNN to effectively learn the correlation between packets of the same flow, facilitating the identification of DDoS patterns regardless of their temporal positioning.

- **CNN Model Architecture:** The LUCID architecture includes a CNN, as previously mentioned, with several layers:

  - Input Layer: Preprocesses network traffic into a 2-D matrix of packet features, facilitating the CNN's understanding of packet correlations.

  - Convolutional Layer: Applies convolutions to extract local features crucial for identifying DDoS and benign flows.
  - Max Pooling Layer: Down-samples input along the temporal dimension, reducing network complexity.
  - Classification Layer: Uses a sigmoid activation function to output the probability of a flow being a malicious DDoS attack.

- **Learning Procedure:** During training, LUCID minimizes its cost function by iteratively updating all the weights and biases contained within the model. This involves feeding the input data forward through the network, calculating the error, and back-propagating this error until convergence is reached. The model uses a binary cross-entropy cost function to calculate the error over a batch of samples, ensuring that it learns the correct feature representations from the patterns in the traffic flows.

- **Hyper-Parameter Tuning:** LUCID employs a grid search strategy to explore the set of hyper-parameters using the F1 score as the performance metric. This process optimizes the model's accuracy by influencing the model's complexity and learning process.

- **Kernel Activation Analysis:** LUCID utilizes a kernel activation analysis technique to interpret and explain the features to which the model attaches importance when making a DDoS classification. This analysis provides insights into the significance of different features in the detection process.

Building upon the architecture and functions described earlier, the model's operational performance is demonstrated through the results showcased below, highlighting its efficacy, specially in a in resource-constrained environment.

The authors of this model performed some tests to evaluate the model's overall performance. The obtained results demonstrate its high performance and robustness across various test datasets. The model's accuracy in classifying unseen traffic flows as benign or malicious showcases consistently high performance across different test sets, as indicated by the high values of Accuracy, Precision, Recall, and F1-score. For example, the model achieved an F1-score of 0.9889 on the ISCX2012 dataset, 0.9966 on the CIC2017 dataset, and 0.9987 on the CSECIC2018 dataset, demonstrating its effectiveness in detecting DDoS attacks.

In comparison with other models, LUCID's performance was comparable to or outperformed them. Additionally, LUCID exhibited significantly faster detection time compared to those approaches, it was

capable to process 55000 samples per second. Another important aspect to mention is the training time, even without using a Graphics Processing Unit (GPU), the time was around 2 hours, which surpass other existent approaches.

In summary, the results affirm the LUCID model's effectiveness, robustness, and efficiency in detecting DDoS attacks across various datasets, making it a promising solution for environments with limited resources, such as edges. Combining streamlined network traffic preprocessing, a CNN-based architecture, iterative learning, hyper-parameter optimization, and kernel activation analysis, the LUCID system emerges as a potent defense against DDoS threats and a great choice to integrate this system.

4.1.2 Threat Classification Approach

To enhance the system's robustness, a ML approach was selected to classify detected threats. Specifically, this ML model was designed to identify various types of DDoS attacks. This insight is crucial for the mitigation function, which relies on accurate and timely alerts to respond effectively.

Unlike the model used for initial attack detection, which was adapted from third-party sources, this model was designed from the ground up. Its primary objective is to leverage the packet feature extraction performed by the initial model, significantly reducing the system's processing time, as feature extraction is typically a slow process.

Key architectural details include:

- **Model Architecture:** The Random Forest model developed for this project is designed for multi-class classification. This architecture uses an ensemble of decision trees, where each tree is constructed independently using different subsets of the data and features. It can classify 12 types of DDoS attacks: DNS attack, SynFlood, UDPLag, WebDDoS, TFTP attack, MSSQL attack, LDAP attack, NetBios attack, NTP attack, SSDP, SNMP attack and UDP Flood.

- **Learning Procedure:** A Random Forest Classifier is initialized with a fixed random state to ensure reproducibility. The training procedure involves fitting multiple decision trees on various subsets of the data, enabling the ensemble to capture a wide range of patterns and relationships within the data.

- **Hyperparameter Tuning:** The model employs Grid Search with cross-validation to determine the optimal hyperparameters. The best model is selected based on the highest accuracy score, ensuring the most effective performance.

This approach not only enhances the system's robustness but also ensures efficient processing, allowing for quicker and more accurate threat classification. This is critical in maintaining the security and reliability of the system in the face of DDoS attack strategies.

4.2 Overall System Architecture

The SDA component is inserted in the 6G-OPENSEC-SECURITY project. Which architecture can be found in Figure 14. This Figure also highlights the direct alignment of this architecture with the overarching ZSM framework, which will be discussed in the following section.

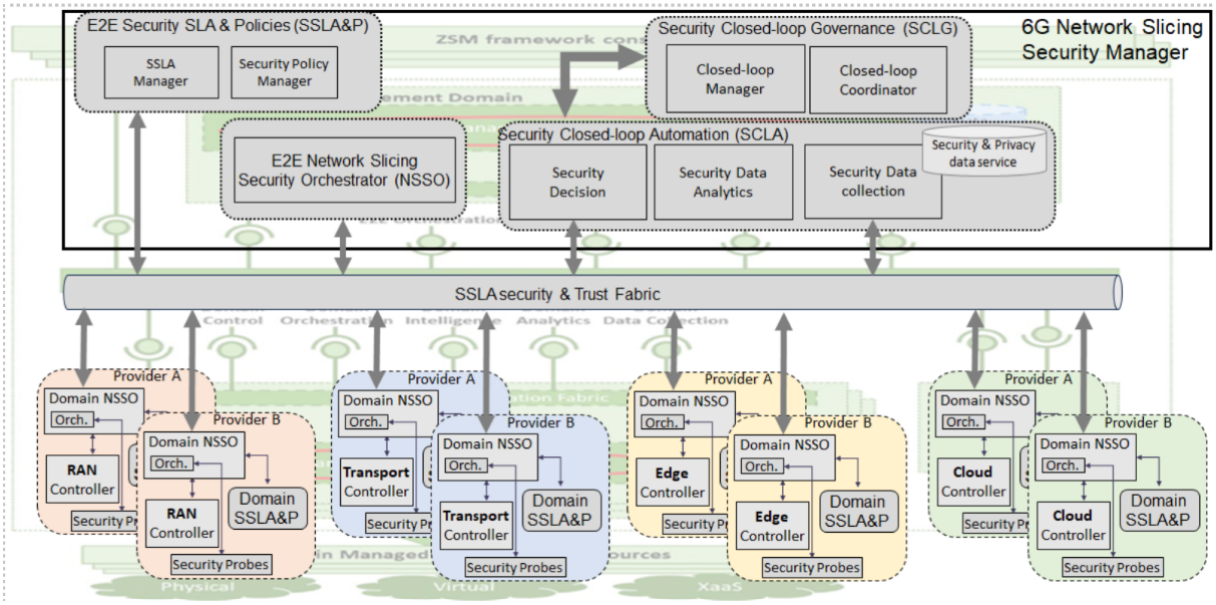


Figure 14: Framework for the 6G-OPENSEC-SECURITY project.

Within this framework, as depicted in Figure 13, one crucial element is the SCLA, comprising four distinct components:

- **Security Decision** - responsible for determining mitigation actions for detected threats, based on policies and Service Level Agreements.
- **SDA** - responsible for detecting threats, leveraging ML mechanisms.
- **Security Data Collection** - collects network traffic and data.
- **Security & Privacy Data Service** - acts as the database of the SCLA.

Notably, among these components is SDA, the primary focus of this dissertation.

4.3 ZSM Architecture

As previously mentioned, the architecture of this project is founded on the ZSM[1] framework specified by ETSI. Specifically, it is rooted in the ETSI GS ZSM 002 [82] specification.

The ZSM architecture addresses the increasing complexity of networks and services, particularly in 5G and beyond environments. This architecture automates network functions and services, focusing on security, privacy, and integrity. ZSM's framework allows for autonomous solutions across network operations, including planning, delivery, deployment, provisioning, monitoring, and optimization—all completed automatically without human intervention [83]. Figure 15 shows the architecture diagram of ZSM.

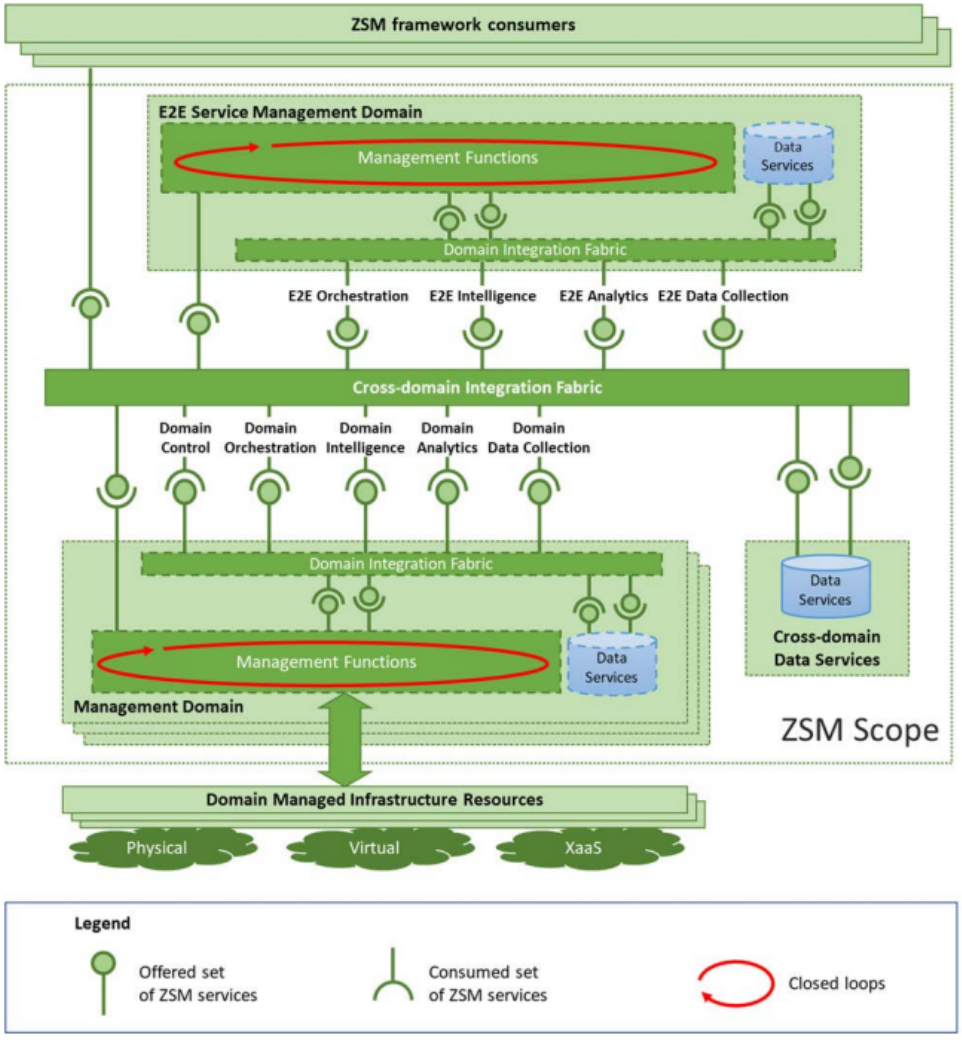


Figure 15: ZSM framework reference architecture [82].

One of the fundamental aspects of this framework is the Closed-Loop Automation (CLA), specified in [84, 85], playing an essential role in ensuring stability, efficiency, and security of operations. CLA stands

[1] https://www.etsi.org/technologies/zero-touch-network-service-management

out as a powerful tool in real-time threat detection and mitigation. CLA's continuous monitoring and evaluation process automatically detects both known threats, such as DDoS and MITM attacks, and addresses emerging and unknown threats. This capability is augmented by the use of ML and AI techniques, enabling an adaptive and proactive response to threats.

In addition to threat detection and mitigation, CLA plays a crucial role in ensuring network stability and efficient coordination of autonomous functions. By introducing preventive measures and coordinating operations simultaneously, CLA contributes to more efficient and reliable automation

For readers interested in delving deeper into this topic, they can consult the specifications provided by the ETSI, accessible on the committee's page[2].

However, it's essential to highlight that Closed-Loop Automation has an important data analytics function, crucial for attack/threat detection, where relies the work documented in this dissertation.

In summary, the ZSM architecture provides a comprehensive framework that supports the autonomous functioning of network management services and is the base architecture to the 6G-OPENSEC-SECURITY project.

[2] https://www.etsi.org/committee/1431-zsm

Chapter 5

Implementation

This chapter details all of the implementation process for the SDA component, the focus of this dissertation.

The first section of this chapter provides a detailed explanation of the selected technologies, libraries, and tools that best meet the system's objectives.

After that, the focus shifts to training the ML models. Key activities included algorithm selection, defining model architectures, and optimizing parameters. Additionally, preparing the dataset was a crucial step in the process.

Then, the implementation of the model is described. Each module was designed to perform distinct tasks aligned with the overall objectives of the component.

Lastly, the testing procedures that were crucial to validate the functionality and performance of the SDA component ae described. This included unit testing of individual modules and integration testing to verify seamless interaction between external components.

5.1 Technologies and Tools

In this project, a variety of technologies and tools were employed to ensure efficient development, system robustness, and simplified maintenance. The technological choices were based on criteria of efficiency, compatibility, and ease of use, ensuring that the project met the proposed requirements and objectives.

In this section, it is presented the core technologies and tools that were essential for the development and deployment of the project. These include the programming languages, development environments, version control systems, database management systems, messaging platforms, containerization technologies, and tools for network traffic analysis and ML. The following subsections provide detailed descriptions of these technologies and their specific contributions to the project.

5.1.1 Core Technologies

The core technologies that form the backbone of the project are:

- **Python 3.10:** The primary programming language used for the project's development. Python was chosen for its simplicity, versatility, and wide range of libraries and frameworks that support diverse applications, including ML.

Figure 16: Python programming language logo. (Python)

- **PostgreSQL:** Used to manage and communicate with the Security & Privacy Data Service database. PostgreSQL was chosen for its robustness, scalability, and support for advanced data types, ensuring efficient and secure data management.

Figure 17: PostgreSQL database managing system logo. (PostgreSQL)

- **Apache Kafka:** A distributed event streaming platform used as a message broker. Kafka facilitated efficient and independent communication between microservices, supporting real-time data processing in a scalable and fault-tolerant manner. It was used to facilitate external communication between the SDA and other components, as well as to enable communication within the modules of the SDA.

Figure 18: Apache Kafka message broker logo. (Apache Kafka)

- **Docker:** Used to containerize the SDA, aiding in deployment and maintenance, ensuring consistency across development, testing, and production environments. Docker also allowed the use of environment files containing specific configurations, simplifying the management of environment variables and secret configurations without altering the source code, enhancing security and ease of application management.

Figure 19: Docker logo. (Docker)

- **TensorFlow:** An open-source ML library developed by Google. TensorFlow provided tools for building and training DL models, fundamental to the project's ML tasks. TensorFlow's flexibility and scalability make it suitable for ML and DL tasks in this context. The version used was the 2.8.0.

Figure 20: TensorFlow library logo. (TensorFLow)

- **GitHub:** It is a web-based platform designed primarily for version control using Git. It allows hosting, managing, and code reviewing. GitHub facilitates version control. In addition, GitHub

supports Continuous Integration/Continuous Delivery (CI/CD) workflows. this way, GitHub was utilized to store the code, facilitating versioning throughout the development process and also as a CI/CD tool.

Figure 21: GitHub plataform logo. (GitHub)

- **Tshark:** Tshark is a network protocol analyzer and a command-line version of Wireshark. It captures and interprets network traffic in real-time, providing insights into network performance and issues. Tshark supports a wide range of protocols and offers various filtering options to focus on specific types of traffic. It can save captured data for later analysis and is commonly used for troubleshooting network problems, monitoring network activity, and conducting security audits. This way, it was used to analyze the incoming traffic. (T-shark)

5.1.2 Libraries and Frameworks

In addition to the aforementioned technologies, several Python libraries and frameworks were utilized to implement various functionalities within the project. The most relevant libraries and frameworks include:

- Flask-SQLAlchemy: An extension that integrates SQL databases seamlessly with Flask applications, providing an Object-Relational Mapping layer for interacting with the database using Python objects.
- psycopg2-binary: A PostgreSQL adapter that allows direct interaction with PostgreSQL databases, supporting SQL commands, connection management, and transaction handling.
- kafka-python: A library for interacting with Apache Kafka, enabling message production and consumption from Kafka topics for real-time data processing.
- pyshark: A wrapper for Tshark that facilitates programmatic capture and analysis of network traffic.

- scikeras: A library that integrates Keras with scikit-learn, allowing Keras models to be used as estimators in scikit-learn pipelines.
- scikit-learn: A ML library offering algorithms for classification, regression, clustering, and dimensionality reduction, along with tools for data preprocessing and model evaluation.
- numpy: It is a fundamental library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
- pandas: It is an open-source data analysis and manipulation library for Python. It provides data structures like DataFrames and Series that are designed to handle structured data intuitively and efficiently.
- Dash: A framework for building interactive dashboards and analytical web applications with rich data visualization components.
- Unittes: It is a Python built-in framework for creating, organizing, and running tests. It was used to perform unit tests.

5.2 Models Training

In this section, it will be covered all procedures related to training the ML models. First, a description about the dataset used for training the models, the CIC-DDoS2019 [74], will be provided. Next, it will be explained how the dataset was processed and how the training phase was executed for each model.

As mentioned in section 4.1, two ML models were utilized in this project. For attack detection, it was selected a DL approach, LUCID [79], specifically designed to detect DDoS attacks. For attack classification, it was developed a Random Forest model.

5.2.1 Dataset description

As previously mentioned, the dataset used to train the models was the CIC-DDoS2019 [74]. Developed by the Canadian Institute of Cybersecurity, this dataset addresses the limitations of existing DDoS datasets, offering a comprehensive and reliable resource for testing and validating DDoS attack detection systems. It was created to provide a realistic and up-to-date dataset that includes complete traffic, attack diversity, and realism, which were often lacking in previous datasets.

Some of the key features of the CIC-DDoS2019 dataset are:

- Realism: The dataset contains real network traffic data, including both benign and malicious traffic, making it suitable for testing the performance of IDS in real-world scenarios.
- Comprehensive Attack Coverage: It includes a diverse set of DDoS attack techniques from the transport and application layers.
- Labeled Data: Each flow in the dataset is fully labeled as either benign or associated with a specific type of DDoS attack.
- Feature-Rich: The dataset includes 80 network traffic features extracted using the CICFlowMeter[1] software, a flow-based feature extractor.
- Public Availability: The dataset is publicly available, in the Canadian Institute of Cybersecurity website [2].

Figure 22 illustrates the attacks covered in this dataset.

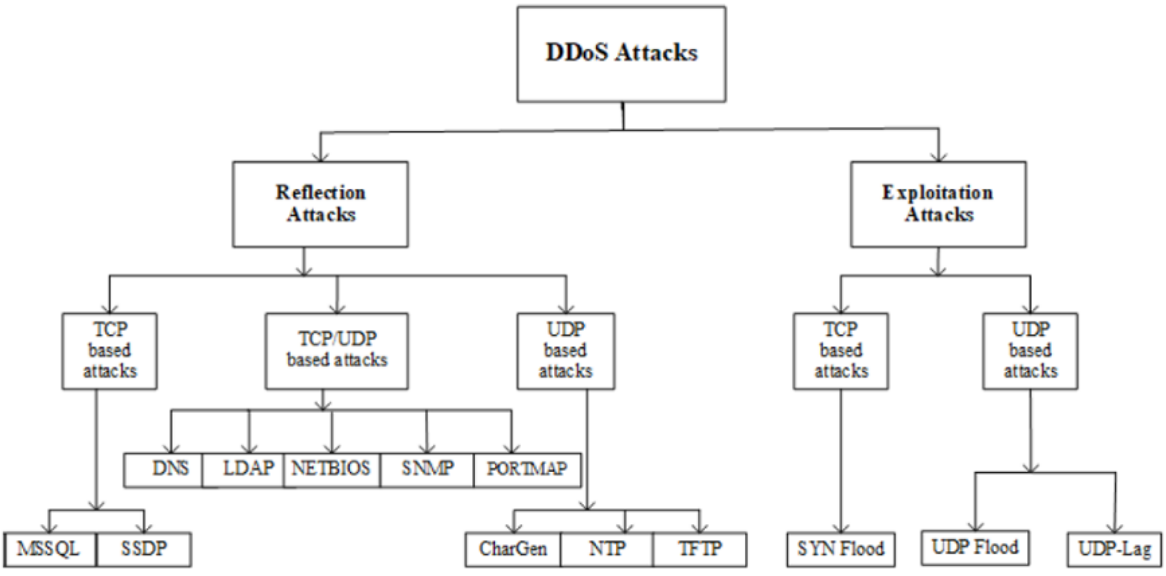


Figure 22: DDoS Attacks proposed taxonomy [74].

The traffic generated for this dataset was produced in two days, the training day and the testing day. This way there was produced two sets: the training set containing 12 types of DDoS attacks (NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS, SYN flood, and TFTP) and the testing set

[1] https://www.unb.ca/cic/research/applications.html

[2] https://www.unb.ca/cic/datasets/ddos-2019.html

that includes 7 types of DDoS attacks (PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, and SYN flood). The attacks were executed at specific times during these days, and the network traffic was captured and processed to create the dataset. Mixed with the attacks there is benign traffic, that it's produced using a benign profile approach to simulate realistic benign user behaviors.

The generation of the dataset used a testbed consisting of two separate networks: the Attack-Network and the Victim-Network. The Victim-Network simulated a typical network environment with a range of common operating systems and network equipment, while the Attack-Network executed various DDoS attacks against it.

The authors also conducted experiments using ML algorithms such as Iterative Dichotomiser 3 (ID3), Random Forest, Naïve Bayes, and Logistic Regression to evaluate the dataset's effectiveness in detecting DDoS attacks. The results indicated that the ID3 algorithm performed the best in terms of execution time and accuracy.

An important note is that the dataset is available in two formats: $'.csv'$ files with extracted features and labels, and raw Packet Capture (PCAP) files containing the original network traffic. For this project, the raw network traffic from the dataset was used.

Therefore, this dataset appeared to be an ideal fit for this project due to its comprehensive nature, extensive volume of data, and up-to-date records of DDoS attacks. Moreover, its proven performance in previous studies further validated its suitability. However, this dataset does not specifically contain 5G NS traffic. Ideally, a dataset tailored to 5G NS traffic would be preferable for this study. Unfortunately, the current literature lacks comprehensive and high-quality public datasets that meet these criteria. Consequently, it was utilized this dataset with general DDoS traffic as the best available alternative. The training set of this dataset was used to train the models, and the testing set was used when testing the system.

5.2.2 Attack Detection Model

To train the LUCID model developed by Doriguzzi-Corin et al. [79], a series of steps were undertaken. Here is a detailed explanation of the procedures:

- **Dataset Parsing and Feature Extraction:** The initial step involves parsing the dataset to extract features from the network traffic. This step transforms raw network data into a structured format suitable for its usage in training the CNN. For this procedure was necessary to use the pyshark library.

- **Data Preprocessing:** Several preprocessing actions are performed on the dataset:

  - Balancing: The dataset was balanced to ensure an equal distribution of benign and malicious traffic.
  - Splitting: The dataset was divided into three subsets: training, validation, and testing.
  - Normalization: Feature values were normalized to ensure they are within a consistent range.
  - Padding: Data padding was applied where necessary to ensure uniformity in feature lengths.

- **Model Training:** The processed dataset is then used to train the LUCID model. For this procedure, was necessary the use of TensorFlow 2.8.0 and the scikeras library, section 5.1. already addressed these technologies.

Also, the numpy library (see section 5.1.) was essential to handle and operate with the arrays of samples.

These steps are executed by following the instructions provided in the GitHub repository[3] with the source code of the model. While the initial steps were straightforward and could be accomplished by running the provided commands, minor modifications were made to the source code for clarity. The training step required additional considerations and infrastructure setup, once it was executed on an Amazon Web Services (AWS) t3.2xlarge instance using Kubernetes. Here's how the training procedure was structured:

1. **Create a Docker Image:** A Docker image was created containing the necessary scripts and dataset required for model training.

2. **Publish the Docker Image**: This image was published to Docker Hub, making it accessible for the AWS instance to pull.

3. **Create a Persistent Volume:** A persistent volume was configured to save the training results. This ensures that the data persists beyond the lifecycle of individual containers.

4. **Deploy a Kubernetes Pod:** A Kubernetes pod was created to host the training container. This pod included the Docker image from step 2 and the persistent volume from step 3.

The deployment and volume configuration were managed using the Kubernetes command line tool, kubectl [4]. After training, to retrieve the results stored in the persistent volume, an additional pod was deployed containing an idle container, also attached to the persistent volume. Using kubectl[5], the results from the persistent volume were copied to a local folder.

[3] `https://github.com/doriguzzi/lucid-ddos`

[4] `https://kubernetes.io/docs/reference/kubectl/`

[5] `https://kubernetes.io/docs/reference/kubectl/`

This process ensured that the model training was efficiently managed and executed independently within a cloud environment.

5.2.3 Threat Classification Model

Similar to the attack detection model, the initial step for the DDoS attack classification model involved preparing and processing the dataset. This process was somewhat more complex compared to the LUCID model.

- **Step 1: Filtering Benign Traffic**

  First, all benign traffic needed to be filtered out from the dataset since this model focuses on classifying types of DDoS attacks, requiring only malign traffic. This was accomplished using Tshark tools (see section 5.1.) and a shell script that iteratively filtered the benign traffic of the PCAP files in a directory, saving the processed files separately.

- **Step 2: Separating Traffic by DDoS Attack Type**

  Once the malignant traffic was isolated, it was necessary to categorize it according to the type of DDoS attack. This process was done manually. Using the start and end timestamps of each attack, provided by the creators of the dataset, the PCAP files were grouped into different folders, each named after the corresponding attack type.

- **Step 3: Parsing, Feature Extraction, and Labeling**

  With the dataset separated and grouped, the next steps were to parse it, extract features, and label it. The features extracted were identical to those used in the LUCID model. No additional feature engineering was performed to determine the optimal features for the model. This decision was driven by the need for low latency in real-time applications; feature extraction from network traffic can be time-consuming. Since this model is intended to complement the LUCID model and aiding in selecting appropriate mitigation actions, low latency was prioritized over potentially higher accuracy. However, the LUCID model samples are in Three-Dimensional (3D) arrays, which is the appropriate format for a CNN. Since now we are using a Random Forest model, the data needs to be in Two-Dimensional (2D) format. To achieve this transformation, it is calculated the mean along axis 1, effectively collapsing the rows within each layer into a single value for each column. For labeling, each flow's label was determined by the name of the folder containing the corresponding PCAP file, which indicated the type of DDoS attack.

- **Step 4: Dataset Processing**

  The processed dataset then underwent several steps:

  - Label Conversion: Labels were converted into numerical values, with each number corresponding to a specific DDoS attack type.
  - Balancing: The dataset was balanced to ensure approximately equal numbers of flows for each attack type.
  - Splitting: The dataset was divided into training, testing, and validation subsets.
  - Normalization and Padding: The flows were normalized and padded as necessary.

- **Step 5: Model Training**

  Following the dataset processing, the training procedure was straightforward. Due to dataset balancing and the relatively small number of samples for certain DDoS attack types (e.g., Web DDoS), the dataset was not large, leading to a less extensive training process. Consequently, the training was conducted using the Google Colab environment to train the threat classification model, based on a Random Forest. This was accomplished using the scikit-learn library

Also, the numpy library was essential to handle and operate with the arrays of samples.

This structured approach ensured that the dataset was prepared and processed, facilitating effective training of the DDoS attack classification model. It's important to note that steps 3 and 4 were carried out using two Python scripts, which were developed by adapting the dataset processing functions from the LUCID model.

5.3 Implementation Details

As mentioned in section 4.1, the SDA component introduced in this dissertation comprises several modules, each with its own functionalities and responsibilities. These modules work together to fulfill the overall purpose of the SDA. Figure 23 provides a visual representation of the interactions among these, as well as the interactions with the external components present in the SCLA.

This section aims to provide a detailed account of the implementation of each module and how they cooperate together.

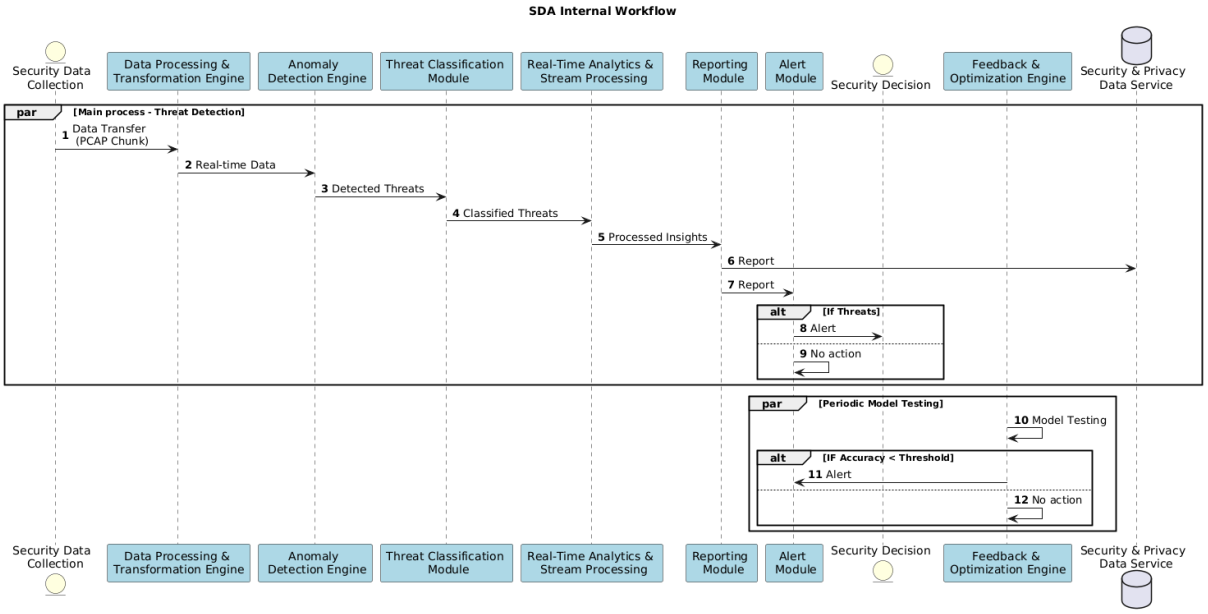


Figure 23: Internal workflow.

5.3.1 Message Broker

The communication between the SDA and other external components is facilitated via the Apache Kafka message broker, as previously mentioned. This means that the SDA sometimes acts as a message producer and other times as a message consumer.

In the communication with the Security Data Collection component, the SDA acts as a consumer, subscribing to the topic $'pcap\_chunk'$. In the communication with the Security Decision component, it acts as a producer, sending messages to the $'threats\_alert'$ and $'acc\_alert'$ topics.

Additionally, the broker is used internally for communication between the Data Processing & Transformation Engine and the Anomaly Detection Engine. In this internal setup, the Data Processing & Transformation Engine acts as a producer, producing the messages in the topic $'anomaly\_detection'$, while the Anomaly Detection Engine acts as a consumer. This design choice allows the system to receive and process network traffic from the Security Data Collection more quickly than the time-consuming tasks of detecting DDoS attacks, classifying them, and storing the results. This setup is important because it naturally creates a queue, allowing network traffic to arrive at a higher rate than what is currently being analyzed.

The broker implementations were done using the kafka-python library (see section 5.1.).

More details about these modules will be provided in the following subsections, ensuring a clearer understanding of this process.

5.3.2 Data Processing & Transformation Engine

This module facilitates communication between the Security Data Collection component and the SDA component. As detailed in previous sections, the Security Data Collection component is responsible for forwarding network traffic, in PCAP file format, to the SDA for further analysis. This communication occurs via a Kafka message broker. Due to broker message size limitations, the PCAP file is divided into chunks before being sent. Consequently, this module is also responsible for reconstructing the PCAP file from these chunks.

Here's a detailed explanation of the process:

1. **Message Reception and Validation:**

    - When a message is received on the broker topic $'pcap\_chunk'$, the first step is to validate the message to ensure it contains all the necessary information for reconstruction. An example of the expected chunk message is in the Appendix B.1.
    - A chunk object is then created from the message.

2. **Storing Chunks:**

    - The chunk object is stored in a dictionary, where the key is the PCAP identifier associated with that chunk.
    - Subsequent chunks are appended to the list of chunks under the same PCAP identifier.

3. **Reconstruction and Cleanup:**

    - Once all chunks are received (identified by sequence numbers and the total number of chunks), the file is reconstructed.
    - The reconstructed file is saved in a specified folder within the container.
    - The chunk object is then removed from memory, and the corresponding dictionary entry is deleted.
    - Finally, the path to the reconstructed PCAP file is sent via the message broker to the Anomaly Detection Engine, in the topic $'anomaly\_detection'$. An example of the message sent is in the Appendix B.2. The value key is essential to identify the message, being the identifier of the PCAP file, and it is part of the broker implementation.

For a better understanding of the PCAP file reconstruction process, refer to Figure 24, which presents a flowchart of the procedure.

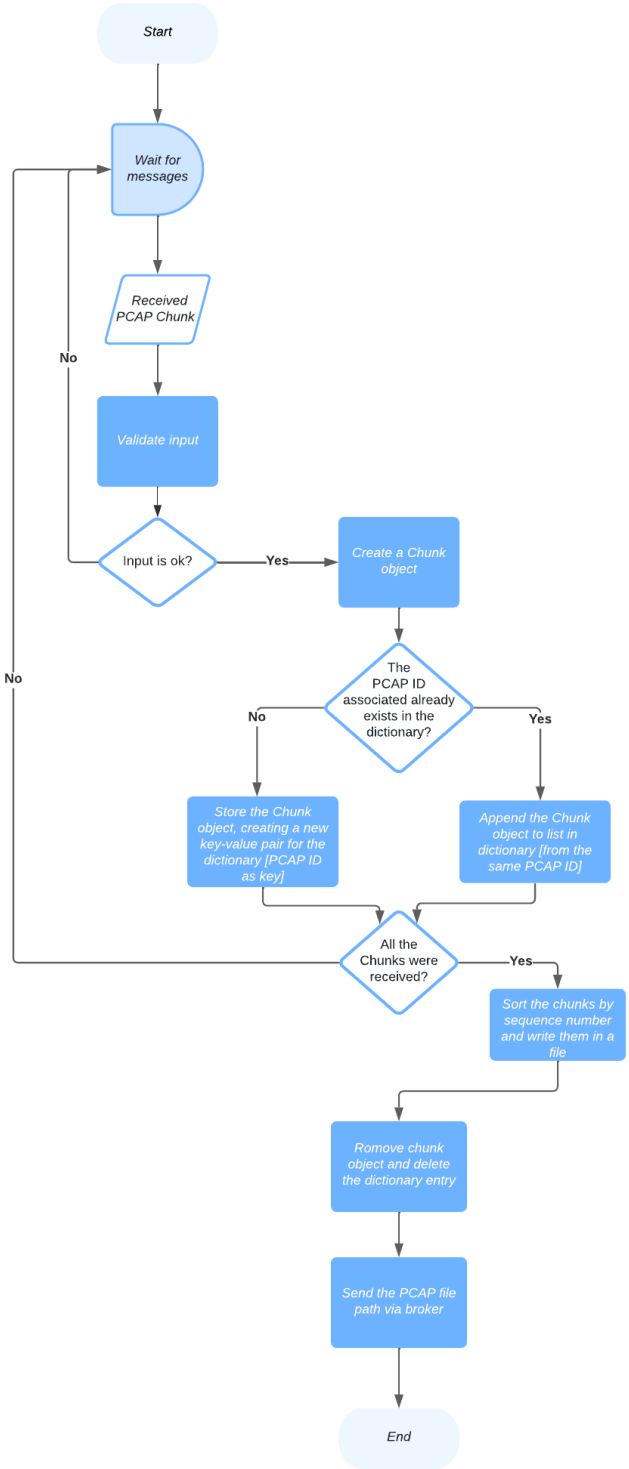


Figure 24: Flowchart of the Data Processing & Transformation Engine process.

5.3.3 Anomaly Detection Engine

This module plays a critical role in the system. It receives the path to the PCAP file via a message broker, by the Data Processing & Transformation Engine, on the specific topic $pcap_chunk$.

The core function of this module, $processing\_pcap\_file()$, manages the entire process of analyzing the PCAP file to detect DDoS attacks and alerting the Security Decision component. This function coordinates with other modules to execute these operations.

Upon receiving the message, the first task is to detect potential DDoS attacks using the pre-trained LUCID model. The LUCID model's source code was extensively modified to retain only the necessary parts, making it more modular and user-friendly. This allows seamless function calls and a better integration with the system. After detection, the function returns all flow identifiers, their predictions, the DDoS rate for the PCAP file, and the prediction timestamp.

If a DDoS attack is detected, the Threat Classification Module is invoked to determine the type of DDoS attack. This module also returns flow identifiers, prediction results, DDoS rate per type, and the prediction timestamp.

The information from both detection and classification stages is then passed to the Real-Time Analytics & Stream Processing component to generate a detailed threat report. This report is stored in the database by the Reporting Module, which also updates relevant database entries. If threat reports are generated, the Alert Module is triggered.

In case of any errors during this process, the system attempts processing up to three times. If processing fails after three attempts, the PCAP file's status is marked as error in the database, indicating the need for further intervention outside this component's scope.

This process is illustrated in the activity diagram in Figure 25.

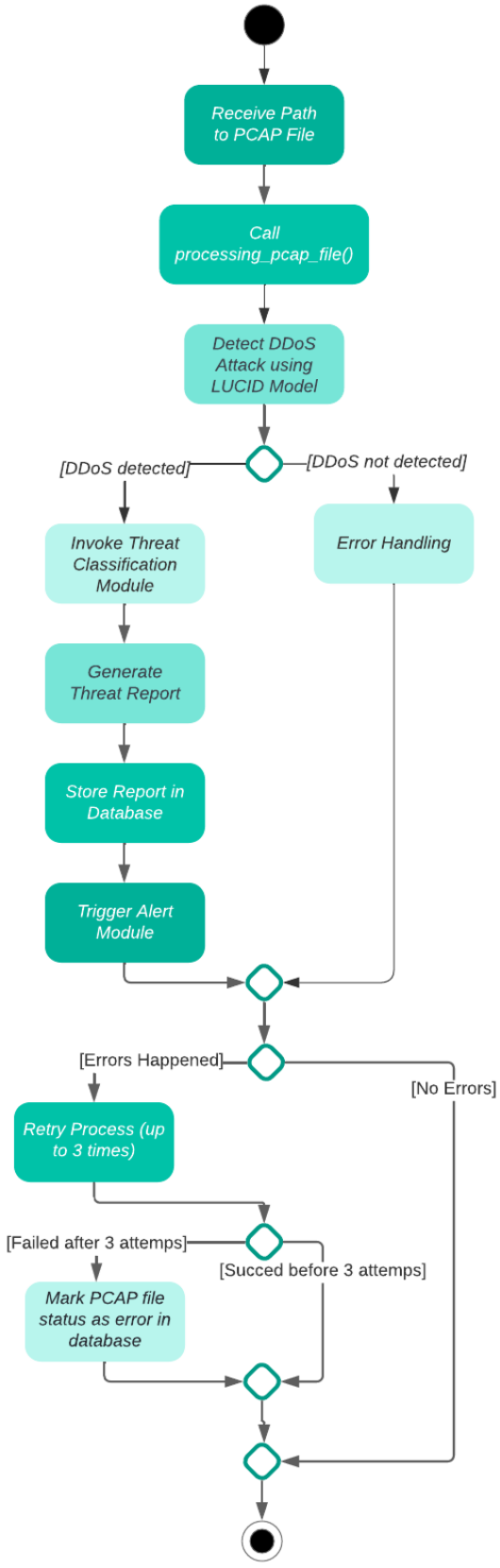


Figure 25: Activity Diagram of the Anomaly Detection Engine process.

5.3.4 Threat Classification Module

This module is responsible for classifying the type of DDoS attack detected. As mentioned in Section 4.1.2, it can classify up to 12 different types of DDoS attacks and plays a crucial role in the system.

To accomplish this, the module first transforms the 3D data from the LUCID model into a 2D format suitable for the Random Forest model. This transformation involves calculating the mean along axis 1, collapsing the rows within each layer into single values for each column. Once transformed, the 2D data is fed into the Random Forest model to make its predictions.

The module then returns several pieces of information:

- Flow identifiers.
- Predictions for each flow.
- The DDoS rate for each type of attack.
- The prediction timestamp.

5.3.5 Real-Time Analytics & Stream Processing

This module has two main functionalities: processing data collected from model predictions and the Feedback & Optimization Engine, and transforming it into suitable data objects for reporting, alert processes, and display on a dashboard.

The module converts data into DataFrame format, facilitating better visualization and making it suitable for display on the dashboard and manipulation by other modules. Specifically, the module creates five DataFrames:

- Results of DDoS detection and classification.
- DDoS rate over time.
- DDoS rate per type.
- Testing results of the LUCID model.
- Testing results of the Random forest model.

Details of these Data Frames are provided in Appendix B.3.

When new data is added to the DataFrames, entries with timestamps older than 30 minutes are filtered out, except for the DataFrame that stores testing results, which has a threshold of 1 hour. This prevents the system's memory from being overloaded with outdated data.

The key libraries for this processing functions were numpy and pandas. For the dashboard, the dash library was essential. (see section 5.1.)

The dashboard is updated with these DataFrames and has features four tabs: DDoS Count, DDoS Rate Over Time, DDoS Rate per Type, and Model Test Results.

1. **DDoS Count:** This tab includes a graphic and a dropdown menu with three options: Internet Protocol (IP) Source, IP Destination, and Protocol. The graphic dynamically updates based on the selected option:

    - IP Source: Displays the count of DDoS flows by IP Source, with the X-axis representing the IP sources and the Y-axis showing the number of DDoS flows detected.
    - IP Destination: Similar to IP Source, but the X-axis represents the IP destinations.
    - Protocol: Shows the count of DDoS flows by protocol encountered, with the X-axis representing different protocols.

2. **DDoS Rate Over Time:** This tab presents a graphic showing the DDoS rate over time for an analyzed PCAP file. The X-axis represents the timestamp, while the Y-axis displays the DDoS rate on a scale from 0% to 100%.

3. **DDoS Rate per Type:** This tab features a graphic depicting the rate of different DDoS types in the most recently analyzed PCAP file, with classes on the X-axis and their corresponding rates on the Y-axis. Below the graphic is a table displaying details, including the class, rate, number of classified flows, total number of flows, and detection timestamp.

4. **Model Test Results:** This tab contains two graphics:

    - The first graphic shows metrics for the LUCID model, with timestamps on the X-axis and metric values ranging from 0 to 1 on the Y-axis.
    - The second graphic is similar but displays the results for the Random Forest model.

Figures 26 and 27 shows examples of two tabs of the dashboard, DDoS Count tab and DDoS Rate tab.

The rest of the dashboard can be found in Appendix C.1.

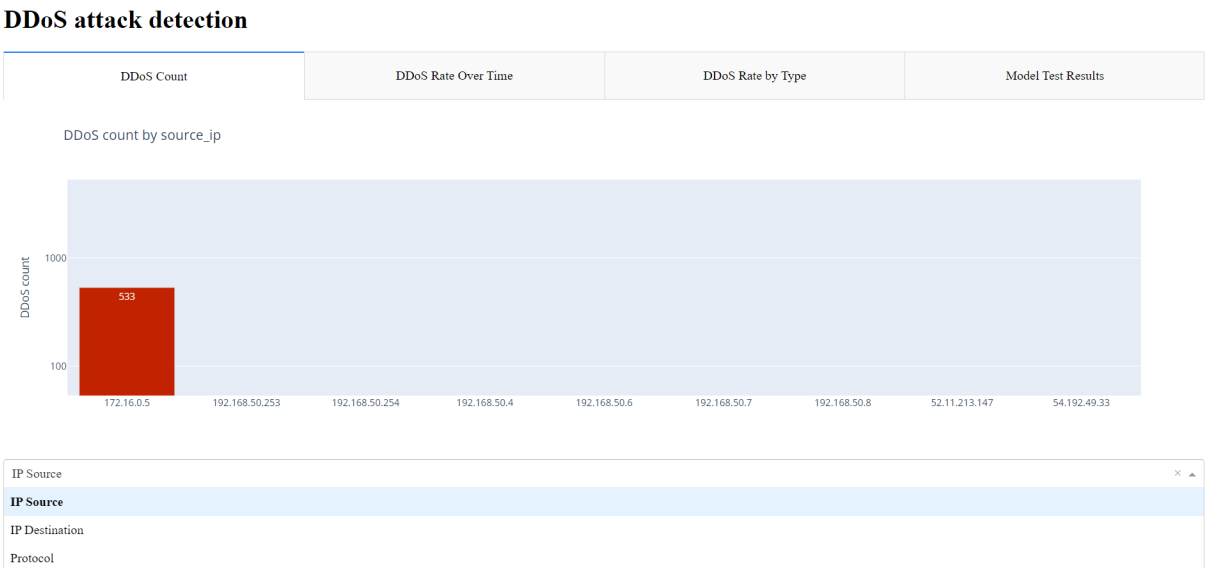


Figure 26: Dash Tab for DDoS Count.

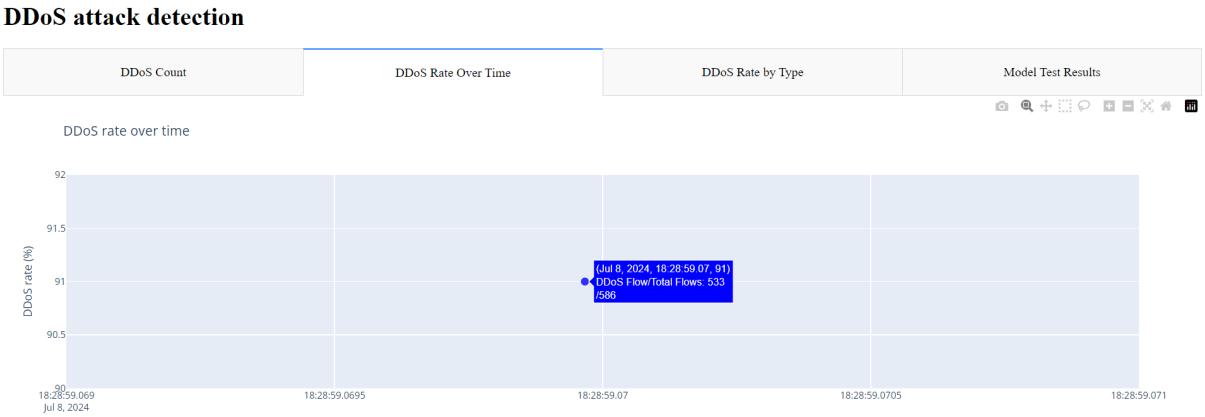


Figure 27: Dash Tab for DDoS Rate.

5.3.6 Reporting Module

This module is responsible for updating the database tables. It stores threat reports and updates entries related to the analyzed PCAP file in the database.

When storing threat reports, certain considerations are taken into account. Multiple flows identified as DDoS attacks may share the same source IP address, destination IP address, protocol, and DDoS type. In such cases, storing multiple reports with essentially the same information is unnecessary. Therefore, the reports are grouped, and only one representative report is stored in the database.

Database interactions are direct, as the database is integrated into the system as a library. This allows for straightforward function calls to perform necessary updates.

After storing the report, the returned result from the database is converted into JavaScript Object Notation (JSON) format and returned, to be further sent to the Security Decision component, by the Alert Module. Details of this data object can also be found in Appendix B.4

This module is essential for maintaining up-to-date database records.

5.3.7 Feedback & Optimization Engine

This module operates in parallel with the main process (detecting DDoS attacks). It is responsible for continuously evaluating the effectiveness of the models. Every 10 minutes, it invokes a test function for each model, using the test dataset obtained from the dataset processing described in Section 5.2. Various metrics such as accuracy, F1-score, precision, and recall are then calculated and the Real-Time Analytics & Stream Processing is called to display the metrics on the dashboard.

If the accuracy of a model falls below a certain threshold (85% for the LUCID model and 65% for the Random Forest model), the Alert Module is called, and a message is sent to the Security Decision component via a message broker (this will be addressed in section 5.3.8). In fact, one of two messages can be sent, or even both one for each model.

Details of the data object of these messages can be found in Appendix B.5.

5.3.8 Alert Module

his module is responsible for sending alert messages to the Security Decision component via a message broker. There are two types of alerts:

- A threat report if any flow is detected as a DDoS attack.
- An alert about the accuracy of the models.

Each type of message is sent to a different topic: $'threats\_alert'$ for threat reports and $'acc\_alert'$ for accuracy alerts. Accuracy alerts can generate two types of messages, depending on the model being monitored. These messages are identified by a key included in the message, which is an integral part of the Kafka broker's implementation. Data objects sent on both of these topics can be found in Appendixes B.4 and B.5, respectively.

5.4 Development Process

The development of this project began with the selection and training of appropriate DL and ML models (section 5.2 addressed the models training process). Once the models were prepared and ready for use, the next phase involved writing and developing the necessary code. Python 3.10 was chosen as the programming language for its versatility and library support, while Visual Studio Code was utilized as the code editor. Visual Studio Code has a great integration with GitHub, where the project's code is stored.

GitHub plays a crucial role in this project, serving not only as a remote repository and version control system but also as a CI/CD tool. Through GitHub Actions, a powerful CI/CD platform, it is possible to automate the build and testing processes. GitHub Actions enables the creation of workflows that automatically build and test each pull request made to the repository.

Whenever a change is proposed, GitHub Actions triggers the deployment of a Docker container to execute unit tests on the modified code, using the unittest library. This automated testing ensures that any issues are identified and resolved before the changes are merged into the main branch. If the tests pass without any issues, the changes are merged successfully. However, if any tests fail, the merge is blocked, and the code must be revised and corrected before being resubmitted. This rigorous process maintains the integrity and reliability of the project's codebase.

Figure 28 gives a visual representation of this process.

In addition to the unit tests, integration tests were also performed. These tests were carried out manually to ensure that the SDA component could effectively communicate with the external components. The goal of these integration tests was to verify that the SDA could interact and function as intended with all the other parts of the SCLA system.

5.4.1 Unit Tests

In this project, unit tests were designed to verify the functionality of specific functions and sections of code. The main goal was to ensure that these sections produced the correct inputs and outputs and to check the behavior of the code when errors occurred.

To test these code sections effectively, some modules and input data objects were mocked.

Due to the complexity of certain sections, such as the application of ML models, manual testing was used. Additionally, modules interacting with the database of the SCLA were not tested within this scope. However, functions responsible for input validation and data transformation were tested to confirm that they produced the correct outputs.

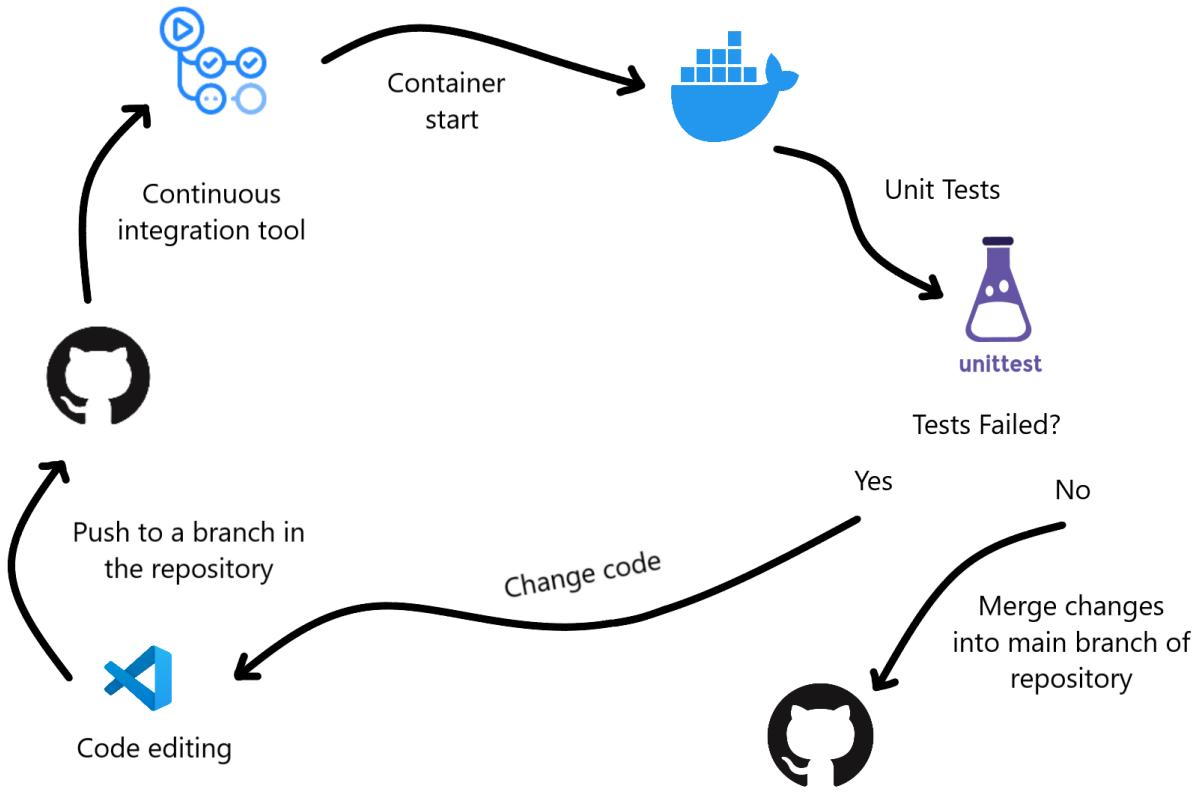


Figure 28: Development process with unitary tests.

The unit tests applied were:

- For the Data Processing & Transformation Engine, it was tested if the network traffic in PCAP format was reconstructed as expected.
- For the Feedback & Optimization Engine, we tested if all output data objects were constructed correctly.
- For the Alert Module, it was tested if all output data objects were constructed correctly and inputted into the message broker.
- For the Real-time Analytics & Stream Processing, it was tested if all data objects were processed and transformed as expected.

5.4.2 Integration Tests

The primary goal of the integration tests was to verify the integration of the SDA with the other components of the SCLA.

To test the communication between the Security Data Collection and the SDA, which interact via a Kafka message broker, the Security Data Collection was mocked using the Kafkacat [6], a Kafka Command Line Interface (CLI) tool. Data that the SDA was expected to receive was input into the appropriate Kafka topic, and any errors in the SDA were observed.

For testing the communication between the SDA and the database (Security & Privacy Data Service), the process was straightforward. Since the database structure is installed in the SDA as a library, it allows easy access to methods for populating tables. If the library is used incorrectly, it will naturally throw an exception, indicating an issue.

Finally, to test the communication between the SDA and the Security Decision component, the messages sent by the SDA were monitored using a Kafka Graphical User Interface (GUI) [7] to ensure they were transmitted as expected. This GUI allows to check all the topics belonging to the broker and the messages sent to these topics. This also helps in checking all the communications that occur via broker. This GUI as more functionalities, such as managing the topics, delete them, create new ones, etc., check which broker nodes are active and much more. Images of this GUI and some of its functionalities can be seen in Appendix D.1.

[6] https://github.com/edenhill/kcat

[7] https://hub.docker.com/r/tchiotludo/akhq

Chapter 6

Results and Discussion

In this chapter, it will be presented the the results from the developed system, focusing on two main areas. First, it will be discussed the results obtained from training and testing the two ML models used in the system. This includes an overview of the evaluation metrics and a detailed analysis of the performance of each model. Following this, it will delve into the results of various experiments conducted with the SDA, highlighting the practical implications and effectiveness of the system. Through this chapter, the aim is to provide a clear understanding of the models' capabilities and the system's overall performance.

6.1 Model's Training and Testing

This section presents the results from training and testing the ML models used in the component. It begins with a brief overview of the metrics employed for models evaluation, followed by the discussion of the results. It is important to note that these tests were done to the models separately from the rest of the component, so they don't represent the performance of the component when all operational.

6.1.1 Models Evaluation Metrics

In order to ensure a full assessment of the ML models' performance, a variety of metrics are typically used. These metrics offer insights into numerous aspects of the model's performance. Some of these metrics are calculated using the measures True Positive (TP), True Negative (TN), FP, and FN. To evaluate the performance of the employed models in this project, the used metrics were:

- **Accuracy:** Measures the proportion of correct predictions relative to the total predictions made by the model.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6.1}$$

- **Precision:** Indicates the proportion of TP relative to the total predicted positive results by the

model.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.2}$$

- **Recall (or True Positive Rate (TPR)):** Measures the proportion of TP relative to the total positives.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.3}$$

- **F1-Score:** A metric that combines precision and recall into a single measure, useful in situations with imbalanced classes.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.4}$$

- **Mean Squarred Error (MSE):** Quantifies the average of the squares of the differences between predicted and actual values.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{6.5}$$

- **True Negative Rate (TNR):** Indicates the proportion of TN relative to the total negatives.

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{6.6}$$

- **FPR:** Indicates the proportion of FP relative to the total negatives.

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{6.7}$$

- **False Negative Rate (FNR):** Indicates the proportion of FN relative to the total positives.

$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} \tag{6.8}$$

- **Area Under ROC Curve (AUC):** A measure of how well the model can distinguish between two classes, representing the probability that the model will rank a random positive instance higher than a random negative instance.

$$\text{AUC} = \int_0^1 TPR(FPR^{-1}(t))\, dt \tag{6.9}$$

6.1.2 DDoS Detection Model

After parsing the dataset, we obtained a total of 2,446,546 instances, comprising 130,824 benign instances and 2,315,722 DDoS instances. However, the dataset needed to be balanced due to the disproportionate number of DDoS instances compared to benign ones. After balancing, the dataset contained a total of 261,648 instances, evenly divided into 130,824 benign and 130,824 DDoS instances.

The dataset was then split into training, testing, and validation sets. 10% of the total instances was allocated for the testing set. From the resulting 90% of the instances, 10% were allocated for the validation set and rest for the training dataset. This resulted in 211,933 instances for training, 26,166 for testing, and 23,549 for validation.

The LUCID model was trained on an AWS t3.2xlarge instance, which has the following specifications: 8 vCPUs (Intel Xeon Scalable Skylake, 3.1 GHz), 32.0 GiB of memory, and up to 5 Gbps of bandwidth. The primary drawback of this instance is the absence of a GPU, which could expedite the training process for DL models. Despite this, the training process was completed in about 2 hours without the use of a GPU. The best hyper-parameters encountered during the training process were:

- Batch Size: 2048
- Dropout: None
- Kernels: 32
- Learning Rate: 0.1
- Regularization: None

After the training process, the validation set was used to confirm the model's performance and the obtained Accuracy was 0.9983 and the F1-score was 0.9984.

To ensure that the model has a great performance, the model was tested with the testing set, data that the model had never seen. The results of all the tested metrics are presented in Table 4.

Table 4: LUCID Testing Model Results.

Metric	Value
Accuracy	0.9985
F1-Score	0.9985
TPR	0.9973
FPR	0.0003
TNR	0.9997
FNR	0.0027
Precision	0.9997
Recall	0.9973
AUC	0.9985
MSE	0.0015

The number of tested samples was 26,166, with a prediction time of 0.109 seconds, which gives approximately 240,055 samples per second. This is a fast prediction time, considering the test was conducted on Google Colab with the default runtime.

As evidenced by the evaluation metrics, the model demonstrated great performance, not only in terms of good predictions, but also in terms of processing time. All metrics achieved good values, indicating the model's high accuracy, precision, recall, and overall effectiveness. This evaluation confirms that the model meets the desired standards for its application in this system.

6.1.3 Threat Classification Model

The first step was to parse the dataset. After parsing, the number of samples for training was relatively small, with only 2,424 instances. Despite having a large number of DDoS samples (2,315,722), it was necessary to balance the samples across different types of DDoS attacks, as some types had many samples while others had very few, namely the WebDDoS type. Consequently, there were 202 samples for each type of DDoS attack.

The dataset was then divided into training, testing, and validation sets, using the same logic as for the LUCID model. This resulted in 1,962 instances for training, 243 for testing, and 219 for validation.

Four different algorithms were trained and tested before selecting Random Forest as the model to be used in the system. The algorithms evaluated included Decision Tree, Random Forest, SVM, and CNN models. Table 5 presents the results from the validation set and the optimal hyper-parameters identified

during training. All models were trained using the Google Colab environment with the default runtime.

Table 5: Models training and validation results.

Model	Accuracy	F1-Score	Hyper-parameters
Decision Tree	0.7534	0.7497	Criterion: entropy, Max Depth: 20, Max Features: None, Min Samples Leaf: 1, Min Samples Split: 5, Splitter: best
Random Forest	0.7580	0.7496	Bootstrap: False, Max Depth: None, Max Features: sqrt, Min Samples Leaf: 2, Min Samples Split: 5, Number Estimators: 100
SVM	0.4384	0.3799	C: 100, Gamma: 1, Kernel: rbf
CNN	0.7077	0.7056	Batch Size: 2048, Dropout: None, Kernels: 32, Learning Rate: 0.01, Regularization: None

As it is possible to see, the model with best performance was the Random Forest, achieving an Accuracy and a F1-score of approximately 0.75, even though the performance of the Decision Tree was very close.

To ensure the performance of the models, they were tested with the testing set and with more metrics. The results are presented in Table 6. Additionally, the Receiver Operating Characteristic Curve (ROC) curve graphs for each model and all classes can be found in Appendix A.1.

Table 6: Testing Models Results.

Metric	Random Forest	Decision Tree	SVM	CNN
Accuracy	0.8436	0.8066	0.4486	0.8148
F1-Score	0.8364	0.8025	0.3766	0.8152
AUC	0.9798	0.9692	0.9187	0.9676
Precision	0.8367	0.8022	0.3649	0.8358
Recall	0.8436	0.8066	0.4486	0.8148
MSE	7.1852	9.0947	13.5885	6.7202

The Random Forest model demonstrates the highest overall performance. During the testing phase, it achieved the highest accuracy (0.8436), F1-score (0.8364), and AUC (0.9798).

The Decision Tree model also performs well, with a testing accuracy of 0.8066 and an F1-score of 0.8025.

The CNN model exhibits better performance in the testing phase than in the training phase, with a testing accuracy of 0.8148 and an F1-score of 0.8152. Additionally, the CNN has the lowest MSE at 6.7202, indicating better prediction accuracy in terms of numerical error.

In contrast, SVM model performs poorly compared to the other models. It has the lowest accuracy (0.4486) and F1-score (0.3766) during testing, and the highest MSE (13.5885), suggesting that it struggles significantly with this classification task. This indicates that the SVM is not be suitable for this particular problem.

In summary, the Random Forest model is the best performer, followed by the Decision Tree and CNN models, which also show strong results. The SVM model, however, does not perform well and may require re-evaluation or further tuning.

However, the metrics for the best model, Random Forest, are not exceptionally high. The achieved Accuracy and F1-score do not exceed 85%. Several factors can explain these results. Firstly, the number of samples is low, which can affect the model's learning capacity. Secondly, some types of attacks are very similar to others, leading to misclassifications when the model doesn't have enough samples to distinguish them accurately.

For a clearer understanding, Table 7 presents the values of Precision, Recall, and F1-Score for each class, obtained during the testing phase for the Random Forest model.

Based on the Table 7, the Random Forest model demonstrates varying performance across different classes of attacks. For classes such as Syn Flood, TFTP, and MSSQL, the model achieves perfect precision, recall, and F1-scores (1.00), indicating effectiveness in correctly identifying these attacks. The LDAP class also shows high performance with metrics close to 1.00.

For Web DDoS, NetBios, and NTP attacks, the model maintains relatively high precision and recall values, resulting in F1-scores around 0.96, which indicates good but not perfect performance. The DNS class has perfect precision but a recall of 0.87, leading to an F1-score of 0.93, suggesting that while the model accurately identifies DNS attacks, it misses 13% of them.

However, the model's performance drops significantly for classes such as UDP Lag, SSDP, and UDP Flood, with precision, recall, and F1-scores ranging from 0.15 to 0.45. This indicates substantial difficulties in correctly classifying these attacks. Particularly, the UDP Lag class has a low precision of 0.25 and a very low recall of 0.11, resulting in a poor F1-score of 0.15. This means that while the model correctly identifies some UDP Lag attacks, it misses 89% of actual instances.

Class	Precision	Recall	F1-Score	Support
DNS - 0	1.00	0.87	0.93	15
Syn Flood - 1	1.00	1.00	1.00	16
UDP Lag - 2	0.25	0.11	0.15	18
Web DDoS - 3	0.92	1.00	0.96	23
TFTP - 4	1.00	1.00	1.00	18
MSSQL - 5	0.95	1.00	0.97	19
LDAP - 6	1.00	0.96	0.98	24
NetBios - 7	0.93	1.00	0.96	26
NTP - 8	1.00	0.92	0.96	26
SSDP - 9	0.39	0.47	0.42	15
SNMP - 10	0.96	1.00	0.98	24
UDP Flood - 11	0.40	0.53	0.45	19

Table 7: Precision, Recall, F1-Score, and Support for Each Class

Key insights reveal that the model shows high precision but low recall for certain classes, indicating a conservative prediction approach that leads to many missed instances. Balanced performance is observed in classes like Web DDoS, LDAP, and NTP. The model struggles with specific attacks, likely due to insufficient training data or the inherent similarity of these attacks.

Another factor contributing to poor classification is that the samples inputed in this model have the same features as the ones inputed in the LUCID model, as already described in section 5.2.3, this will helps prioritize latency over good model's performance.

Nevertheless, the lack of higher Accuracy and F1-score in this model is not a significant concern. The LUCID model already demonstrates great performance by identifying nearly all DDoS attacks. This model serves primarily as an additional tool to assist the decision-making process to mitigate the attack.

6.2 System Evaluation and Testing

This section aims to document various tests conducted to evaluate the overall component, working integrally. Initially, it will cover precision and efficiency tests to assess the performance of the ML models when incorporated in the component. Following that, it will examine the system's overall performance. Finally, it will evaluate the system's resilience. Also, a discussion of the results will be encountered.

The PCAP files encountered in CICDDoS2019 [74] dataset (testing set), were also used in this testing phase, due to the lack of trustworthy data available. However, other sources of traffic were employed. The objective was to test the model using 5G NS traffic, but suitable public datasets are limited. Nonetheless, Khan et al. [86] developed a dataset called SliceSecure, which includes various types of DDoS attack traffic in the NS environment. Although the publicly available portion on the author's GitLab [1] contains only a few examples, it is still useful for testing the system. Additionally, benign examples from the dataset developed by Coldwell et al. [87], which features 5G traffic, were used. Some captures available in [88] were also utilized.

6.2.1 Precision and Efficacy

The primary focus is on assessing the capability on the entire system, specifically the LUCID model, to differentiate between malicious and benign traffic. Subsequently, the efficacy of the Random Forest model in distinguishing between various types of DDoS attacks was evaluated. Due to the difficulty in obtaining diverse traffic samples for all the addressed types of DDoS attacks, only the ones in the testing set of the CICDDoS2019 [74] dataset were tested.

It is important to note that all the tests done here were done to the all system, using PCAP files. The traffic was carefully selected and inputed in the system with prior knowledge of the type of traffic it represents.

First, the efficacy of the system to identify benign flows was tested using traffic samples from the Coldwell et al. [87] dataset, which is from a UE in a 5G network, and the SliceSecure [86] dataset from two network slices.

Table 8 shows the obtained results for the benign traffic classification, including the source of the traffic, the TP, TN, FP, and FN flows, the total number of flows, the classified type of DDoS if applicable, and the DDoS rate. As shown, the system performed well in identifying benign traffic. However, it is evident that some flows were misclassified as malicious, specifically with the Syn Flood DDoS type. This misclassification can be attributed to certain communications that resemble a Syn Flood attack due to the high quantity of SYN-ACK packets.

Then, the efficacy of the system to identify DDoS flows was tested, focusing on how these flows were classified in terms of the type of DDoS. The testing set from the CICDDoS2019 [74] dataset and some samples of DDoS attacks found in [88] were used. Table 9 presents the obtained results, including the source of the traffic, the DDoS type to be tested, the TP, TN, FP, and FN flows, the total number of flows,

[1] https://gitlab.com/sajidkhan382067/ddos-data-sets-2022

Source	TP	TN	FP	FN	Total flows	Classified Type DDoS	DDoS Rate (%)
[87]	0	18	0	0	18	N/A	0
[86]	0	17895	647	0	18542	Syn Flood	3.5
[86]	0	35165	420	0	35585	Syn Flood	1.2

Table 8: Results for benign flows classification.

the classified type of DDoS, and the DDoS rate. Due to limitations of the dataset and finding trustworthy examples of specific types of DDoS attacks, the tests were only performed on NetBios, LDAP, MSSQL, UDP Flood, Syn Flood, and NTP DDoS attacks.

The classification results indicate high accuracy for most DDoS types, with detection rates nearing 100% for NetBios, LDAP, MSSQL, UDP Flood, and Syn Flood attacks. However, the NTP DDoS shows a significantly lower detection rate of 83.19%, suggesting challenges in accurately classifying NTP flows compared to other DDoS types. This discrepancy may occur because the signature of the attack sample used for this test might differ slightly from the one the model was trained with. For the other tested types, the signature is likely similar to those in the training set, as these samples are from the same author. The breakdown of classified types within each category highlights the presence of misclassifications and overlaps. This is especially notable in the case of NTP, where a substantial portion of the flows were classified as other types. Also, in the case of the UDP Flood, where more than half of the flows were classified as SSDP. This may happen because both UDP Flood and SSDP attacks utilize the UDP protocol, which does not have built-in mechanisms for establishing a connection or ensuring delivery. This similarity can make it difficult for some classification algorithms to distinguish between different types of UDP-based attacks.

After conducting all these tests, it is evident that the system can correctly classify benign and malicious traffic flows. However, the training set should be more extensive to include a broader range of signatures for different DDoS types, thereby improving the model's accuracy. While the results for classifying the types of DDoS attacks are as expected, they could be further enhanced by utilizing a larger and more diverse training dataset.

6.2.2 Performance

To evaluate the system's performance, two metrics were assessed: scalability and latency. Scalability was measured by inputting a large volume of data, and observing the system's response. Latency was

Source	Type DDoS	TP	TN	FP	FN	Total flows	Classified Type DDoS	DDoS Rate (%)
[74]	NetBios	9984	0	0	9	9993	NetBios - 9013 DNS - 357 Others - 623	99.99
[74]	LDAP	2207	0	0	1	2208	LDAP - 2171 SNMP - 19 DNS - 10 Others - 10	99.99
[74]	MSSQL	9697	0	0	11	9708	MSSQL - 9216 SNMP - 286 Others - 206	99.88
[74]	UDP FLood	9193	0	0	9	9202	SSDP - 5490 UDP Flood - 3519 UDP Lag - 193	99.99
[74]	Syn FLood	7111	0	0	104	7215	Syn Flood - 7111	98.55
[88]	NTP	485	0	0	98	583	NTP - 189 Others - 394	83.19

Table 9: Results for malicious flows classification.

measured to determine the time required for the system to process a PCAP file and classify the flows.

Regarding the scalability of the system, it was observed that PCAP files with large amounts of flows and data—specifically, files over 20 MB—failed to be processed. The container exited with error code 137, indicating insufficient RAM, as the flows are stored in memory until classified. It is important to consider the hardware used for these tests: a Dell Latitude 5510 laptop with 16GB of RAM, and the maximum RAM that the container can use is 8GB. However, the containerized environment utilizes only a fraction of the available RAM, which explain this problem. It is anticipated that better resources will be available where the system is ultimately implemented, mitigating this issue. Nevertheless, as expected, processing smaller amounts of data at a time would be more efficient.

The latency was tested using the PCAP files already used to test the precision and efficacy of the system. So, it was measured the time since the PCAP is received until all the reports are generated and

sent. Table 10 presents the results obtained including the size of the PCAP file, the total number of flows, the number of identified DDoS flows and benign flows and the time of processing.

Table 10 presents the relationship between traffic type, processing time for PCAP files, and the number of flows. The data analysis reveals a trend of increasing processing time as the number of flows increases.

For instance, processing times exceeding 300 seconds are associated with benign traffic containing 35,585 flows, whereas shorter times, such as 38.99 seconds, correspond to smaller flow quantities, like 2,208 in the case of LDAP traffic. It's also notable the variation in processing time for different traffic types with similar flow quantities. For example, the Syn Flood has a bigger processing time than the NetBios, MSSQL and UDP Flood that have more flows, this may happens because the Syn Flood is based on TCP packets which require more processing given the nature of the protocol. The other types of traffic are based on UDP packets.

The system shows efficiency in handling smaller flow quantities, as evidenced by the shorter processing times for LDAP. However, there is room for improvement, especially in enhancing consistency and reducing processing times for larger data volumes.

Traffic Type	**Time (s)**	**Number of flows**
Benign	157.62	18542
Benign	303.26	35585
NetBios	56.15	9993
LDAP	38.99	2208
MSSQL	61.82	9708
UDP Flood	63.95	9202
Syn Flood	64.09	7215

Table 10: Processing time and quantity of flows.

In summary, the system demonstrates the capability to process various types of network traffic, but the required time increases as the number of flows grows. Improvements in the processing algorithm or underlying infrastructure could reduce processing times for large flow quantities, thereby increasing system efficiency and also the scalability of the system. It is important to note that the hardware used in these tests is limited and the system can achieve better results with more resources

6.2.3 Resilience

To test resilience, different signatures of DDoS attacks not included in the training dataset were input into the models to evaluate their ability to correctly classify these new attack signatures. This helps determine if the system can adapt to and detect new attacks. However, it is expected that the classification of the type of DDoS attack may fail since the model was developed to classify 12 specific types of DDoS attacks. Therefore, if the attack is not among these 12, the classification will naturally be incorrect.

For this evaluation, traffic samples from various DDoS attacks were used:

- **BACnet Protocol Attack:** Typically used in IoT environments, this attack was chosen to see if the system could handle protocols outside the standard set.
- **SYN-ACK Reflection Attack:** Unlike a SYN flood attack, this involves the attacker sending SYN packets to a server with the victim's IP, causing the server to respond with SYN-ACK packets to the victim, resulting in traffic overflow.
- **Flood of Fragmented UDP Packets:** This tests the system's ability to manage fragmented packets, which can be challenging to classify.
- **Flood of IPv4 Packets with Random Protocols:** This scenario evaluates the system's capability to handle a variety of unknown protocols.

These tests are crucial for assessing the system's flexibility and robustness in detecting DDoS attack types beyond the ones it was specifically trained to recognize.

Table 11 presents the results of the conducted tests. The system had difficulty identifying most types of attacks. While it was able to detect a flood of fragmented UDP packets, it incorrectly classified these flows as Syn Flood, which is a very different type of attack.

With further refinement and training, the system could enhance its accuracy and adaptability, leading to more reliable detection of a wider range of DDoS attacks. These findings provide valuable insights for future development, suggesting that with targeted improvements, the system can become more robust and effective in identifying novel attack vectors.

Source	Type DDoS	TP	TN	FP	FN	Total flows	Classified Type DDoS	DDoS Rate (%)
[88]	BACnet	2345	0	0	5926	8271	LDAP	28.35
[88]	Syn ACK reflection	0	0	0	7673	7673	N/A	0.00
[88]	Fragmented UDP packets	84	0	0	9	93	Syn Flood	90.32
[88]	IPv4	3	0	0	17201	17204	LDAP - 2 DNS - 1	0.017

Table 11: Results for unknown malicious flows classification.

Chapter 7

Conclusions and future work

This chapter presents the conclusions drawn from the developed work and outlines perspectives for future research and development. It summarizes key findings and suggests potential directions for further investigation.

7.1 Conclusions

This project aimed to develop a system capable of autonomously detecting and classifying DDoS attacks using ML and DL mechanisms. The objective was to construct a system following the ZSM architecture principles [84, 85]. Additionally, this system was designed to integrate into a larger framework: the SCLA, which itself operates within a Security Manager to ensure the security of network slices in a 6G network environment.

Initially, a study was conducted to survey other systems or projects, either already developed or ongoing, in this context. It was found that the scientific community is actively researching the enhancement of security using ML and DL techniques. Several studies employed ML and DL to detect DDoS and other attacks in the 5G and NS environments. However, none of these systems were designed with ZSM principles in mind, nor were they part of an SCLA system. Furthermore, they only distinguish between DDoS and benign traffic and they don't classify the type of DDoS attack.

Thus, the focus was to develop a system capable of detecting various kinds of DDoS attacks using a CNN and classifying their types with good accuracy. This system was designed to introduce low latency and generate security reports to aid the mitigation function in addressing the attacks. Additionally, the system provides a dashboard summarizing the network's security status and continuously evaluates various model metrics to ensure optimal operation.

During the development of this project, several challenges were encountered, such as selecting an appropriate dataset. The literature lacks a robust public dataset containing DDoS samples in the 5G NS

environment. Therefore, the CICDDoS2019 dataset [74] was chosen despite not containing 5G NS traffic, due to its other good characteristics. Another challenge was training the models. For the CNN-based LUCID model, using a GPU would expedite this process; however, the training process was relatively quick even using a CPU. For the Random Forest model, the limited size of the training dataset compromised the model's learning process. During the testing phase, it was anticipated that the model would be implemented and tested in the network, but this was not possible at the time of writing this dissertation.

The need to use mechanisms that ensure the proper development of this system—such as documentation, integration and unit testing, as well as the adoption of new technologies and tools, and the nature of the system itself—increased the complexity of this project. Even so, all the decisions made took into account the available resources and the project's needs.

Finally, it can be considered that all objectives were achieved. The system performs well in detecting DDoS attacks, and the classification of DDoS types provides valuable insights for the mitigation function to adapt accordingly. The system is fully operational and integrated with external systems, ready for testing in a real 5G NS environment. It is expected that this system will serve as a solid foundation for enhancing security in 6G open networks.

7.2 Prospect for future work

In order to further enhance the capabilities and robustness of the system, several directions for future work are proposed. Firstly, it will be essential to deploy and test the system in real 5G NS environments to validate its efficacy and performance. This real-world testing will provide insights into the system's behavior under various network conditions and allows to fine-tune it accordingly. This will help in extending the system to meet the needs of emerging 6G networks, ensuring that the system remains relevant and effective as network technologies evolve.

Currently, the dataset used for training and evaluating our models consists of standard DDoS traffic rather than traffic specific to 5G NS. While a dataset that includes 5G NS traffic would be ideal for achieving a readier system to be inserted in this specific domain, there is a notable gap in the literature, as no high-quality dataset with these characteristics is readily available. Addressing this limitation is crucial for enhancing the accuracy and relevance of the system in the context of 5G networks. As such, future work will involve the creation or acquisition of a more suitable dataset and the subsequent re-training of the models with this data.

Given the dynamic nature of network traffic and attack patterns, developing methods for automatic

retraining will be crucial. This will ensure that the system adapts continuously to new threats and remains up-to-date without requiring manual intervention.

Moreover, the current system classifies only 12 types of DDoS attacks, which is a limitation. To address this, there is a need to generalize the DDoS type classification and train the model with a more extensive and diverse dataset. This expansion will enhance the system's ability to detect and classify a broader range of attack types. Through these steps, it is possible to create a more adaptable, accurate, and comprehensive system capable of addressing the evolving challenges in 5G and 6G networks security.

References

[1] L. Bonati et al. “Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead”. In: *Computer Networks* (2020). DOI: 10.1016/j.comnet.2020.107516.

[2] V. Ziegler et al. “Security and Trust in the 6G Era”. In: *IEEE Access* (2021). DOI: 10.1109/ACCESS.2021.3120143.

[3] C. J. Bernardos and M. A. Uusitalo. *European Vision for the 6G Network Ecosystem*. June 7, 2021. DOI: 10.5281/ZENODO.5007671.

[4] S. H. A. Kazmi et al. “Security Concepts in Emerging 6G Communication: Threats, Countermeasures, Authentication Techniques and Research Directions”. In: *Symmetry* (2023).

[5] J. Cunha et al. “Enhancing Network Slicing Security: Machine Learning, Software-Defined Networking, and Network Functions Virtualization-Driven Strategies”. In: *Future Internet* 16.7 (2024). DOI: 10.3390/fi16070226.

[6] P. Alemany et al. “Security and Trust in Open and Disaggregated 6G networks”. In: *2024 24th International Conference on Transparent Optical Networks (ICTON)*. 2024. DOI: 10.1109/ICTON62926.2024.10647935.

[7] M. M. d. Silva and J. Guerreiro. “On the 5G and Beyond”. In: *Applied Sciences* (Jan. 2020). DOI: 10.3390/app10207091.

[8] L. Chettri and R. Bera. “A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems”. In: *IEEE Internet of Things Journal* (Jan. 2020). DOI: 10.1109/JIOT.2019.2948888.

[9] H. Fourati, R. Maaloul, and L. Fourati. “A survey of 5G network systems: challenges and machine learning approaches”. In: *International Journal of Machine Learning and Cybernetics* 12 (). DOI: 10.1007/s13042-020-01178-4.

[10] P. Hedman. *Description of Network Slicing Concept*. Tech. rep. NGMN Alliance, Jan. 2016.

[11] ETSI. *5G*. URL: https://www.etsi.org/technologies/5g (visited on 12/29/2023).

[12] J. Ordonez-Lucena et al. “Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges”. In: *IEEE Communications Magazine* (May 2017). DOI: 10.1109/MCOM.2017.1600935.

[13] X. You et al. “Towards 6G wireless communication networks: vision, enabling technologies, and new paradigm shifts”. In: *Science China Information Sciences* (Nov. 24, 2020). DOI: 10.1007/s11432-020-2955-6.

[14] I. Afolabi et al. “Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions”. In: *IEEE Communications Surveys & Tutorials* (2018). DOI: 10.1109/COMST.2018.2815638.

[15] A. A. Barakabitze et al. “5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges”. In: *Computer Networks* (Feb. 2020), p. 106984. DOI: 10.1016/j.comnet.2019.106984.

[16] M. Giordani et al. “Toward 6G Networks: Use Cases and Technologies”. In: *IEEE Communications Magazine* (Mar. 2020). DOI: 10.1109/MCOM.001.1900411.

[17] M. Z. Chowdhury et al. “6G Wireless Communication Systems: Applications, Requirements, Technologies, Challenges, and Research Directions”. In: *IEEE Open Journal of the Communications Society* (2020). DOI: 10.1109/OJCOMS.2020.3010270.

[18] E. C. Strinati et al. “Reconfigurable, Intelligent, and Sustainable Wireless Environments for 6G Smart Connectivity”. In: *IEEE Communications Magazine* (Oct. 2021). DOI: 10.1109/MCOM.001.2100070.

[19] W. Jiang et al. “The Road Towards 6G: A Comprehensive Survey”. In: *IEEE Open Journal of the Communications Society* (2021). DOI: 10.1109/OJCOMS.2021.3057679.

[20] L. U. Khan et al. “6G Wireless Systems: A Vision, Architectural Elements, and Future Directions”. In: *IEEE Access* (2020). DOI: 10.1109/ACCESS.2020.3015289.

[21] P. Porambage et al. “The Roadmap to 6G Security and Privacy”. In: *IEEE Open Journal of the Communications Society* (May 2, 2021). DOI: 10.1109/OJCOMS.2021.3078081.

[22] M. Ylianttila et al. “6G white paper : research challenges for trust, security and privacy”. In: (June 30, 2020).

[23] S. J. Nawaz et al. “Quantum Machine Learning for 6G Communication Networks: State-of-the-Art and Vision for the Future”. In: *IEEE Access* (2019). DOI: 10.1109/ACCESS.2019.2909490.

[24] Y. Zhou et al. “Service-aware 6G: An intelligent and open network based on the convergence of communication, computing and caching”. In: *Digital Communications and Networks* (Aug. 1, 2020). DOI: 10.1016/j.dcan.2020.05.003.

[25] A. Imanbayev et al. “Research of Machine Learning Algorithms for the Development of Intrusion Detection Systems in 5G Mobile Networks and Beyond”. In: *Sensors* (Jan. 2022). DOI: 10.3390/s22249957.

[26] Y. Siriwardhana et al. “AI and 6G Security: Opportunities and Challenges”. In: *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, June 8, 2021. DOI: 10.1109/EuCNC/6GSummit51104.2021.9482503.

[27] U. GOV. *Understanding vulnerabilities*. URL: https://www.ncsc.gov.uk/information/understanding-vulnerabilities (visited on 11/27/2023).

[28] Rhebo. *Glossary | Industrial anomaly detection explained*. en. Jan. 2019. URL: https://rhebo.com/en/service/glossar/anomaly-detection-en/ (visited on 11/27/2023).

[29] Cisco. *What Is a Cyberattack? - Most Common Types*. en. URL: https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html (visited on 11/27/2023).

[30] MITRE. *Denial of Service, Technique T0814 - ICS | MITRE ATT&CK®*. URL: https://attack.mitre.org/techniques/T0814/ (visited on 11/28/2023).

[31] “A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks”. In: *IEEE Communications Surveys & Tutorials* (2013). DOI: 10.1109/SURV.2013.031413.00127.

[32] M. Conti, N. Dragoni, and V. Lesyk. “A Survey of Man In The Middle Attacks”. In: *IEEE Communications Surveys & Tutorials* (2016). DOI: 10.1109/COMST.2016.2548426.

[33] Cisco. *DNS Tunneling*. Cisco Umbrella. URL: https://learn-cloudsecurity.cisco.com/umbrella-resources/umbrella/dns-tunneling (visited on 12/07/2023).

[34] Y. Ye et al. “A Survey on Malware Detection Using Data Mining Techniques”. In: *ACM Computing Surveys* (May 31, 2018). DOI: 10.1145/3073559.

[35] Cisco. *What Is Phishing? Examples and Phishing Quiz*. URL: https://www.cisco.com/c/en/us/products/security/email-security/what-is-phishing.html (visited on 11/29/2023).

[36] Cisco. *Understanding SQL Injection*. URL: https://sec.cloudapps.cisco.com/security/center/resources/sql_injection.html#1 (visited on 12/07/2023).

[37] IBM. *What is a Zero-Day Exploit? | IBM*. URL: https://www.ibm.com/topics/zero-day (visited on 12/07/2023).

[38] H.-J. Liao et al. “Intrusion detection system: A comprehensive review”. In: *Journal of Network and Computer Applications* (Jan. 1, 2013). DOI: 10.1016/j.jnca.2012.09.004.

[39] A. L. Buczak and E. Guven. “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection”. In: *IEEE Communications Surveys & Tutorials* 18 (2016). DOI: 10.1109/COMST.2015.2494502.

[40] I. Butun, S. D. Morgera, and R. Sankar. “A Survey of Intrusion Detection Systems in Wireless Sensor Networks”. In: *IEEE Communications Surveys & Tutorials* (2014). DOI: 10.1109/SURV.2013.050113.00191.

[41] A. Khraisat et al. “Survey of intrusion detection systems: techniques, datasets and challenges”. In: *Cybersecurity* (July 17, 2019). DOI: 10.1186/s42400-019-0038-7.

[42] R. Singh et al. “Internet attacks and intrusion detection system: A review of the literature”. In: *Online Information Review* (Jan. 1, 2017). DOI: 10.1108/OIR-12-2015-0394.

[43] S. Singh and S. Silakari. “A Survey of Cyber Attack Detection Systems”. In: *International Journal of Security and Its Applications* (2014). DOI: 10.14257/ijsia.2014.8.1.23.

[44] D. Zhang et al. “A survey on attack detection, estimation and control of industrial cyber–physical systems”. In: *ISA Transactions* (Oct. 1, 2021). DOI: 10.1016/j.isatra.2021.01.036.

[45] K. Shaukat et al. “A Survey on Machine Learning Techniques for Cyber Security in the Last Decade”. In: *IEEE Access* (2020). DOI: 10.1109/ACCESS.2020.3041951.

[46] Z. Ahmad et al. “Network intrusion detection system: A systematic study of machine learning and deep learning approaches”. In: *Transactions on Emerging Telecommunications Technologies* (2021). DOI: 10.1002/ett.4150.

[47] G. Kocher and G. Kumar. “Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges”. In: *Soft Computing* (Aug. 1, 2021). DOI: 10.1007/s00500-021-05893-0.

[48] B. Mahesh. “Machine Learning Algorithms -A Review”. In: *International Journal of Science and Research (IJSR)* (Jan. 1, 2019). DOI: 10.21275/ART20203995.

[49] P. Mishra et al. “A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection”. In: *IEEE Communications Surveys & Tutorials* (2019). DOI: 10.1109/COMST.2018.2847722.

[50] G. Bonaccorso. *Machine Learning Algorithms*. Packt Publishing Ltd, July 24, 2017. ISBN: 978-1-78588-451-1.

[51] M. Mohammed, M. B. Khan, and E. B. M. Bashier. *Machine Learning: Algorithms and Applications*. CRC Press, Aug. 19, 2016. ISBN: 978-1-4987-0539-4.

[52] W.-H. Chen, S.-H. Hsu, and H.-P. Shen. “Application of SVM and ANN for intrusion detection”. In: *Computers & Operations Research* (Oct. 2005). DOI: 10.1016/j.cor.2004.03.019.

[53] L. Breiman. “Random Forests”. In: *Machine Learning* (Oct. 1, 2001). DOI: 10.1023/A:1010933404324. (Visited on 07/09/2024).

[54] Z. Ghahramani. “Unsupervised Learning”. In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 72–112. ISBN: 978-3-540-28650-9. DOI: 10.1007/978-3-540-28650-9_5.

[55] Y. Wu, D. Wei, and J. Feng. “Network Attacks Detection Methods Based on Deep Learning Techniques: A Survey”. In: *Security and Communication Networks* 2020 (Aug. 28, 2020). Publisher: Hindawi, e8872923. ISSN: 1939-0114. DOI: 10.1155/2020/8872923.

[56] A. Aldweesh, A. Derhab, and A. Z. Emam. “Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues”. In: *Knowledge-Based Systems* 189 (Feb. 15, 2020), p. 105124. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2019.105124.

[57] G. Zaccone and M. R. Karim. *Deep Learning with TensorFlow: Explore neural networks and build intelligent systems with Python, 2nd Edition*. Packt Publishing Ltd, Mar. 30, 2018. 483 pp. ISBN: 978-1-78883-183-3.

[58] V. Sharma, S. Rai, and A. Dev. “A Comprehensive Study of Artificial Neural Networks”. In: *International Journal of Advanced Research in Computer Science and Software Engineering* 2 (Sept. 2012). ISSN: 22776451, 2277128X.

[59] K. O’Shea and R. Nash. *An Introduction to Convolutional Neural Networks*. Dec. 2, 2015. DOI: 10.48550/arXiv.1511.08458.

[60] M. Mandal. *Introduction to Convolutional Neural Networks (CNN)*. Analytics Vidhya. May 1, 2021. URL: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ (visited on 07/09/2024).

[61] S. Kumar and K. Dutta. “Intrusion detection in mobile ad hoc networks: techniques, systems, and future challenges”. In: *Security and Communication Networks* (2016). DOI: 10.1002/sec.1484.

[62] O. Adeleke. “Intrusion Detection: Issues, Problems and Solutions”. In: *2020 3rd International Conference on Information and Computer Technologies (ICICT)*. IEEE, Mar. 2020. DOI: 10.1109/ICICT50521.2020.00070.

[63] “Fair Resource Allocation in an Intrusion-Detection System for Edge Computing: Ensuring the Security of Internet of Things Devices”. In: *IEEE Consumer Electronics Magazine* (Nov. 2018). DOI: 10.1109/MCE.2018.2851723.

[64] T. Thuvakudimalai and D. A. Kumar. “INTRUSION DETECTION SYSTEMS: A REVIEW”. In: *International Journal of Advanced Research in Computer Science* (Aug. 30, 2017). DOI: 10.26483/ijarcs.v8i8.4703.

[65] J. Arshad et al. “A Review of Performance, Energy and Privacy of Intrusion Detection Systems for IoT”. In: *Electronics* (Apr. 2020). DOI: 10.3390/electronics9040629.

[66] E. Lundin and E. Jonsson. “Anomaly-based intrusion detection: privacy concerns and other problems”. In: *Computer Networks*. Recent Advances in Intrusion Detection Systems (Oct. 1, 2000). DOI: 10.1016/S1389-1286(00)00134-1.

[67] S. Niksefat, P. Kaghazgaran, and B. Sadeghiyan. “Privacy issues in intrusion detection systems: A taxonomy, survey and future directions”. In: *Computer Science Review* (Aug. 1, 2017). DOI: 10.1016/j.cosrev.2017.07.001.

[68] O. f. C. Rights (OCR). *Health Information Privacy*. June 9, 2021. URL: https://www.hhs.gov/hipaa/index.html (visited on 01/27/2024).

[69] E. Comission. *Data protection in the EU - European Commission*. July 4, 2023. URL: https://commission.europa.eu/law/law-topic/data-protection/data-protection-eu_en (visited on 01/27/2024).

[70] L. S. Branch. *Consolidated federal laws of canada, Personal Information Protection and Electronic Documents Act*. June 21, 2019. URL: https://laws-lois.justice.gc.ca/eng/acts/p-8.6/ (visited on 01/27/2024).

[71] A. Thantharate et al. *Secure5G: A Deep Learning Framework Towards a Secure Network Slicing in 5G and Beyond*. Jan. 1, 2020. DOI: 10.1109/CCWC47524.2020.9031158.

[72] A. Thantharate et al. *DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks*. Oct. 11, 2019. DOI: 10.1109/UEMCON47517.2019.8993066.

[73] N. A. E. Kuadey et al. “DeepSecure: Detection of Distributed Denial of Service Attacks on 5G Network Slicing—Deep Learning Approach”. In: *IEEE Wireless Communications Letters* (Mar. 2022). DOI: 10.1109/LWC.2021.3133479.

[74] I. Sharafaldin et al. *Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy*. Oct. 1, 2019. DOI: 10.1109/CCST.2019.8888419.

[75] F. Hussain et al. “A Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks”. In: *IEEE Access* (2021). DOI: 10.1109/ACCESS.2021.3131014.

[76] I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani. *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*. Jan. 1, 2018. 108 pp. DOI: 10.5220/0006639801080116.

[77] N. Koroniotis et al. *Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset*. Vol. 100. Nov. 1, 2018, pp. 779–796. DOI: 10.1016/j.future.2019.05.041.

[78] B. Bousalem et al. “Deep Learning-based Approach for DDoS Attacks Detection and Mitigation in 5G and Beyond Mobile Networks”. In: *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. Milan, Italy: IEEE, June 27, 2022. ISBN: 978-1-66540-694-9. DOI: 10.1109/NetSoft54395.2022.9844053.

[79] R. Doriguzzi-Corin et al. “Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection”. In: *IEEE Transactions on Network and Service Management* 17.2 (June 2020), pp. 876–889. ISSN: 1932-4537, 2373-7379. DOI: 10.1109/TNSM.2020.2971776.

[80] O. R. Sanchez et al. “Evaluating ML-based DDoS Detection with Grid Search Hyperparameter Optimization”. In: *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, June 28, 2021. DOI: 10.1109/NetSoft51509.2021.9492633.

[81] A. Shiravi et al. “Toward developing a systematic approach to generate benchmark datasets for intrusion detection”. In: *Computers & Security* 31 (May 1, 2012), pp. 357–374. DOI: 10.1016/j.cose.2011.12.012.

[82] E. G. Z. 0. v1.1.1. *Zero-touch network and Service Management (ZSM); Reference Architecture*. URL: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_ZSM002v010101p.pdf.

[83] M. Liyanage et al. “A survey on Zero touch network and Service Management (ZSM) for 5G and beyond networks”. In: *Journal of Network and Computer Applications* 203 (July 1, 2022), p. 103362. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2022.103362.

[84] E. G. Z. 0.-1. v1.1.1. *Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers*. URL: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/00901/01.01.01_60/gs_ZSM00901v010101p.pdf.

[85] E. G. Z. 0.-2. v1.1.1. *Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 2: Solutions for automation of E2E service and network management use cases*. URL: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/00902/01.01.01_60/gs_ZSM00902v010101p.pdf.

[86] M. S. Khan et al. “SliceSecure: Impact and Detection of DoS/DDoS Attacks on 5G Network Slices”. In: *2022 IEEE Future Networks World Forum (FNWF)*. 2022. DOI: 10.1109/FNWF55208.2022.00117.

[87] C. Coldwell et al. “Machine Learning 5G Attack Detection in Programmable Logic”. In: *2022 IEEE GLOBECOM Workshops, GC Wkshps 2022 - Proceedings*. 2022 IEEE GLOBECOM Workshops, GC Wkshps 2022 - Proceedings (2023). DOI: 10.1109/GCWkshps56602.2022.10008647.

[88] L. Haaijer. *DDoS Packet Capture Collection*. 2022. URL: https://github.com/StopDDoS/packet-captures/tree/main (visited on 07/22/2024).

Appendix A
Details Of Results

A.1 ROC Curve Graphics

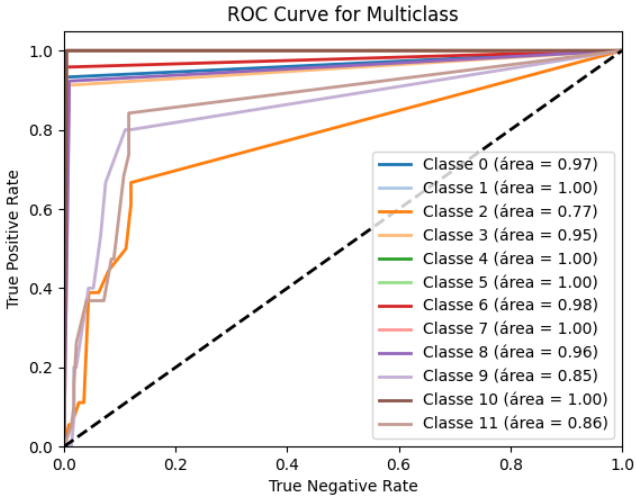


Figure 29: Decision Tree model ROC curve.

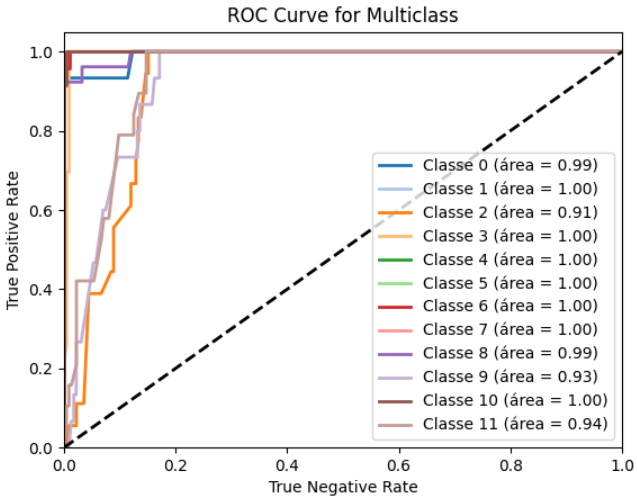


Figure 30: Random Forest model ROC curve.

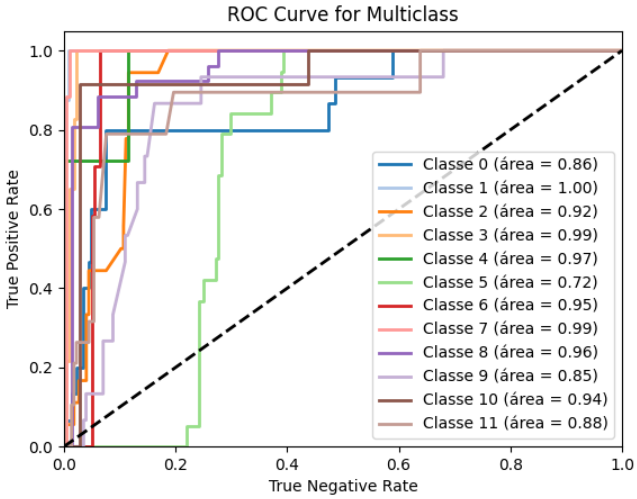


Figure 31: SVM model ROC curve.

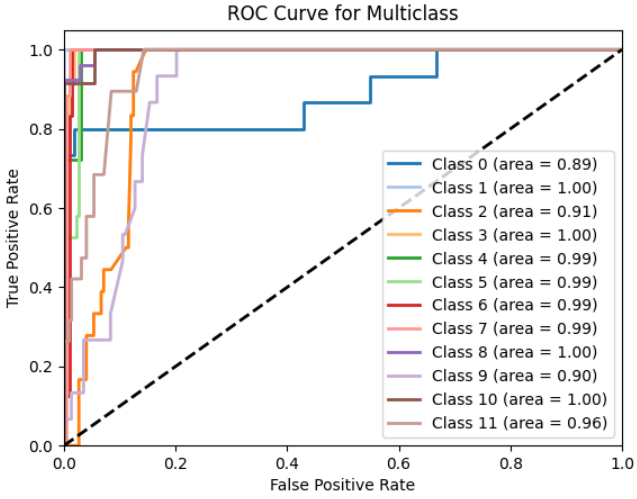


Figure 32: CNN model ROC curve.

Appendix B
Data Objects

B.1 Data Object Sent by Security Data Collection


```
{
    "key": 48_1,
    "message":{
        "total_chunks": 5,
        "chunk_sequece": 1,
        "chunk_data":"r8UCXFqoAgA8AAAAPAAAAJCxHKLA03DzWkJz6AgARQAAKDAwAADyBvngrBAABcCoMgFopWilAAAAAAAAAAABQAhb
        QLHcAAAAAAAAAAK/FAlyKqAIAPAAAADwAAACQsRyiwNNw81pCc+gIAEUAACgwMAAA8gb54KwQAAXAqDIBAwIB0wAAAAAAAAAAUAIW0Js
        1AAAAAAAAAACvxQJci6gCADwAAAA8AAAAkLEcosDTcPNaQnPoCABFAAAoMDAAAPIG+eCsEAAFwKgyAQMCAdMAAAAAAAAAAFACFtCbNQA
        AAAAAAAAr8UCXMioAgA8AAAAPAAAAJCxHKLA03DzWkJz6AgARQAAKDAwAADyBvngrBAABcCoMgGhf6BeAAAAAAAAAABQAhbQQVUAAAA
        AAAAAK/FAlzIqAIAPAAAADwAAACQsRyiwNNw81pCc+gIAEUAACgwMAAA8gb54KwQAAXAqD...",
        "checksum": "097a4dd603af47b754024e3bcd83f9f0"
    }

}
```


Figure 33: Example of data object containing a PCAP chunk.

B.2 Data Object Sent to Anomaly Detection Engine


```
{
    "key":48,
    "message": "security_data_analytics_app/src/pcap/48.pcap"
}
```


Figure 34: Example of data object sent to Anomaly Detection Engine containing the path to the PCAP file.

B.3 Data Frames

	source_ip	dest_ip	protocol	ddos_status	ddos_type	timestamp
0	172.16.0.5	192.168.50.4	17	True	11.0	2024-07-04 12:33:49.611393
1	192.168.50.4	172.16.0.5	1	False	NaN	2024-07-04 12:33:49.611393
..	...	...	...	...	...	
584	172.16.0.5	192.168.50.1	6	True	1.0	2024-07-04 12:33:49.611393
585	172.16.0.5	192.168.50.1	6	False	NaN	2024-07-04 12:33:49.611393

Figure 35: Example of Data Frame containing the results of flows detection and classification.

	ddos_rate	ddos_flows	total_flows	timestamp
0	0.91	533	586	2024-07-04 12:33:49.611393

Figure 36: Example of Data Frame containing the DDoS rate overtime.

	class	rate	ddos_flows_by_class	total_flows	timestamp
0	MSSQL	0.039400	21	533	2024-07-04 12:33:49.793890
1	NTP	0.016886	9	533	2024-07-04 12:33:49.793890
	...	...	...	...	...
5	UDP	0.045028	24	533	2024-07-04 12:33:49.793890
6	UDPLag	0.022514	12	533	2024-07-04 12:33:49.793890

Figure 37: Example of Data Frame containing the DDoS rate per type.

	prediction_time	accuracy	f1	true_positive_rate	false_positive_rate	true_negative_rate	false_negative_rate	precision \
0	0.132283	0.998548	0.998511	0.997338	0.000299	0.999701	0.002662	0.999686

	recall	mse	auc	data_source	timestamp
0	0.997338	0.001452	0.99852	10t-10n-DOS2019-dataset-test.hdf5	2024-07-04 12:33:41.422042

Figure 38: Example of Data Frame containing the results of LUCID model testing.

	prediction_time	accuracy	f1	precision	recall	mse	auc	data_source	timestamp
0	0.123578	0.73	0.71	0.7823	0.7325	8.2305	0.9692	10t-10n-DOS2019-dataset-test.hdf5	2024-07-04 11:01:30.35679

Figure 39: Example of Data Frame containing the results of Random Forest model testing.

B.4 Data Object of Threat Report


```
{
    "id": 100,
    "pcap_id": 48,
    "dest_ip": "192.168.50.1",
    "source_ip": "172.16.0.5",
    "status": 1,
    "protocol": 6,
    "ip_count": 450,
    "timestamp": "2024-07-04T12:00:00Z",
    "prediciton_time": 5.3,
    "ddos_type": 1.0,
    "last_updated": "2024-07-04T15:30:45Z"
}
```


Figure 40: Example of data object representing a threat report in JSON format.

B.5 Data Objects of Model Testing Results


```
{
    "key": "lucid",
    "message": {
        "prediction_time": 0.132283,
        "accuracy": 0.998548,
        "f1": 0.998511,
        "true_positive_rate": 0.997338,
        "false_positive_rate": 0.000299,
        "true_negative_rate": 0.999701,
        "false_negative_rate": 0.002662,
        "precision": 0.999686,
        "recall": 0.997338,
        "mse": 0.001452,
        "auc":0.99852,
        "data_source": "10t-10n-DOS2019-dataset-test.hdf5",
        "timestamp": "2024-07-04 12:33:41.422042"
        }
}
```

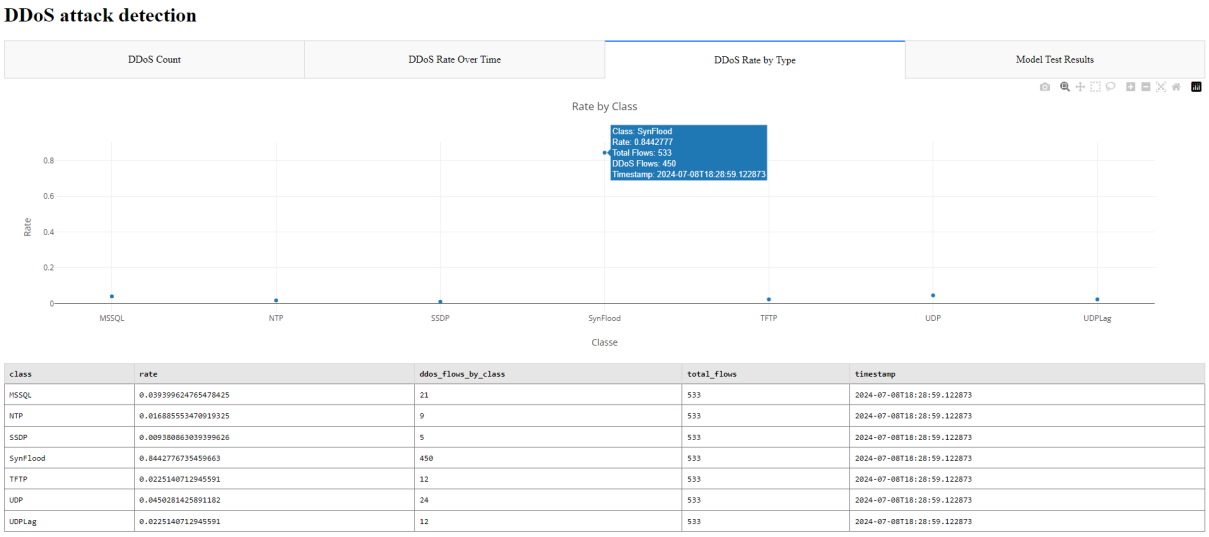

Figure 41: Example of data object representing the LUCID results sent to Security Decision.

```
{
    "key": "rf",
    "message": {
        "prediction_time": 0.016496,
        "accuracy": 0.73251,
        "f1": 0.711959,
        "precision":0.782289,
        "recall": 0.73251,
        "mse":8.230453,
        "data_source": "10t-10n-DOS2019-dataset-multi-test.hdf6",
        "timestamp": "2024-07-04 11:01:30.35679"
        }
}
```


Figure 42: Example of data object representing the Random Forest results sent to Security Decision.

Appendix C
Frontend Templates

C.1 Dashboard Templates



class	rate	ddos_flows_by_class	total_flows	timestamp
MSSQL	0.039399624765478425	21	533	2024-07-08T18:28:59.122873
NTP	0.016885553470919325	9	533	2024-07-08T18:28:59.122873
SSDP	0.009380863039399626	5	533	2024-07-08T18:28:59.122873
SynFlood	0.8442776735459663	450	533	2024-07-08T18:28:59.122873
TFTP	0.0225140712945591	12	533	2024-07-08T18:28:59.122873
UDP	0.0450281425891182	24	533	2024-07-08T18:28:59.122873
UDPLag	0.0225140712945591	12	533	2024-07-08T18:28:59.122873

Figure 43: Dash Tab for DDoS Rate per type.

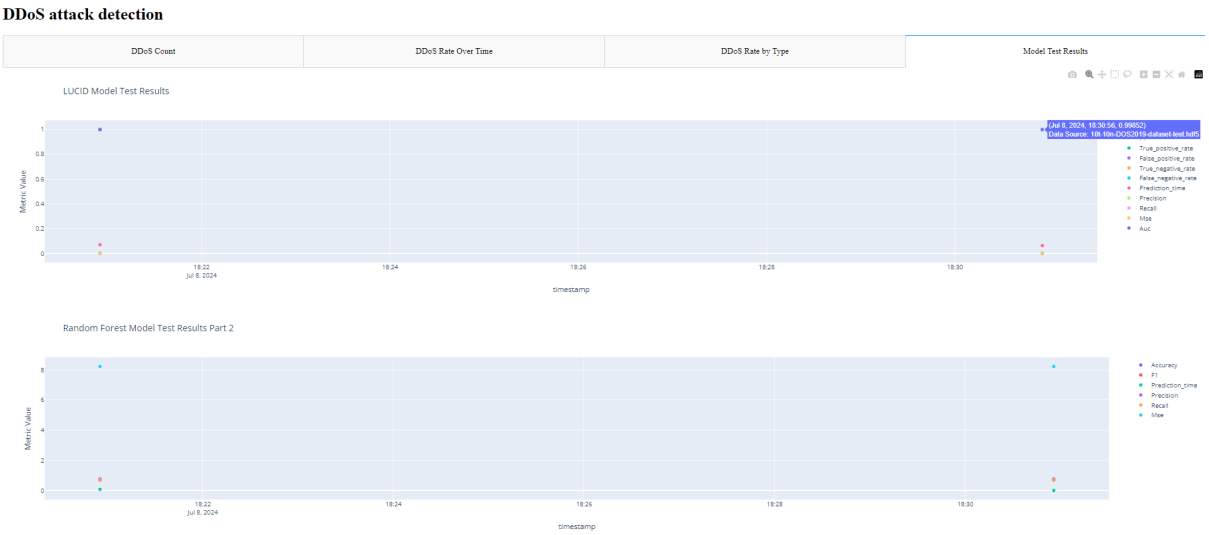


Figure 44: Dash Tab for Model Test Results.

Appendix D
Tooling

D.1 Kafka GUI

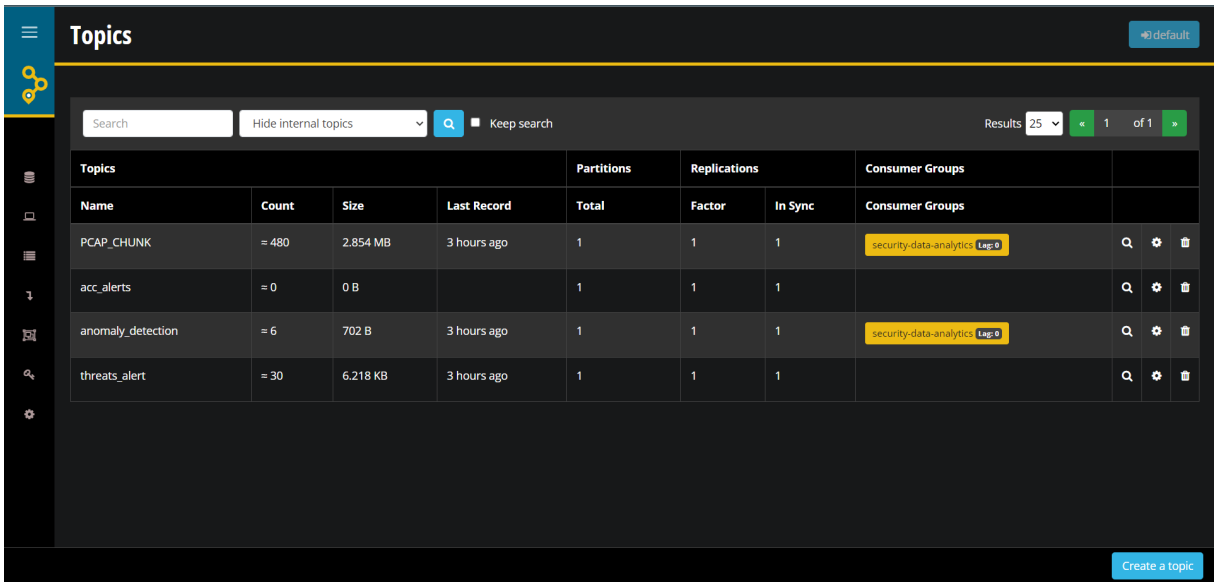


Figure 45: Broker topics in Kafka GUI.

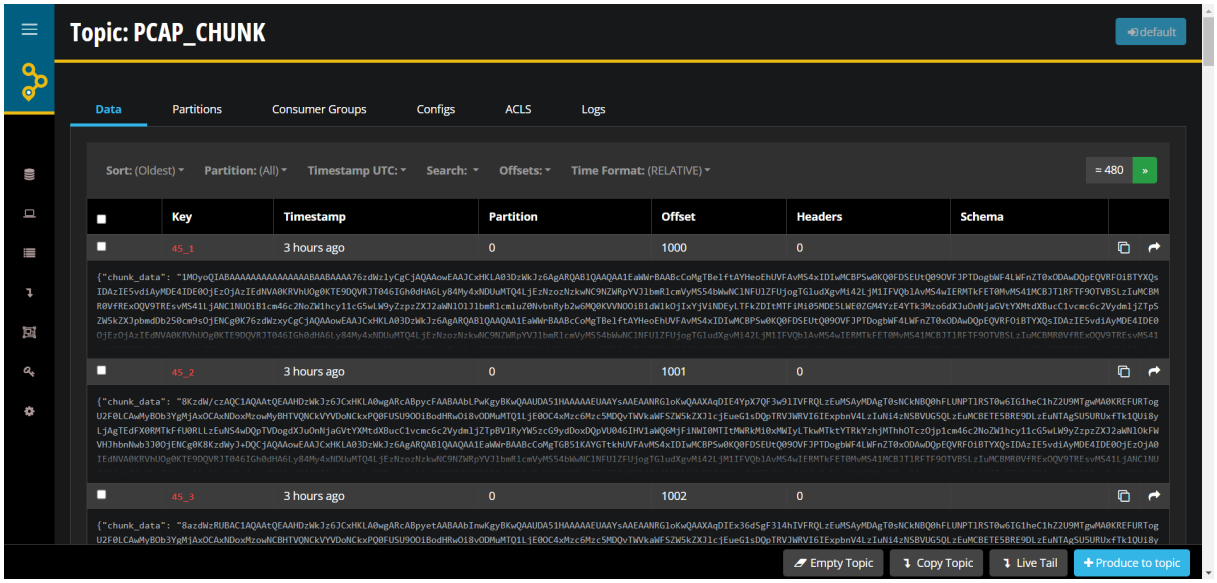


Figure 46: Messages sent in the pcap_chunk topic.

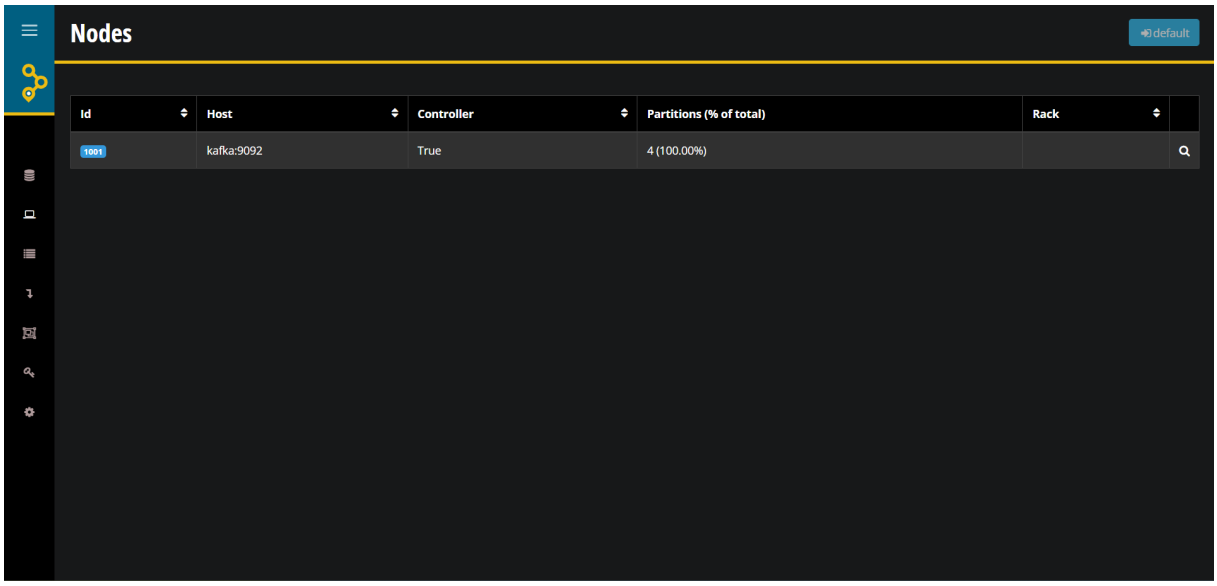


Figure 47: Active broker nodes.