871780 — MonB5G — ICT-20-2019-2020

Deliverable D5.2
Final report on AI-driven security techniques

Document Summary Information

Grant Agreement No	871780	**Acronym**	MonB5G
Full Title	Distributed Management of Network Slices in beyond 5G		
Start Date	01/11/2019	**Duration**	42 months
Project URL	https://www.monb5g.eu/		
Deliverable	D5.2 Final report on AI-driven security techniques		
Work Package	WP5		
Contractual due date	M34	**Actual submission date**	31/08/2022
Nature	Report	**Dissemination Level**	Public
Lead Beneficiary	EUR		
Responsible Author	Adlen Ksentini (EUR), Sabra Ben Saad (EUR)		
Contributions from	Sabra Ben Saad (EUR), Mohamed Mekki (EUR), Sofiane Messaoudi (EUR), Adlen Ksentini (EUR), Eric Gatel (BCOM), Cao-Than PHAN (BCOM), Cédric MORIN (BCOM), Dimitris Manolopoulos (eBOS), George Tsolis (CTX)		

Disclaimer

Copyright message

Executive Summary

Securing the Beyond 5G networks is of vital importance to gain the trust of the vertical industry, which constitutes the main targeted consumers of this new generation of networks. In this context, network slicing was explicitly designed to unleash the potential of B5G to support vertical industry scenarios, whatever the needed requirements. However, securing B5G networks and network slicing has not yet been completed, and several parallel works from academia and industry are still ongoing. In this context, MonB5G aims to provide a secure and trusted environment for network slice deployment and management. The distributed nature of MonB5G elements and their Artificial Intelligence (AI)-based features are instrumental in achieving these objectives. By applying advanced data analysis and Machine Learning (ML) techniques, early and accurate identification of attacks will be possible via the automatic application of localized, self-healing mechanisms to mitigate them, taking advantage of the distribution of security management tasks and security enforcement points across the architecture.

As previously introduced in D5.1, the security functionality in MonB5G is provided by a security orchestrator (SO) entity, which examines security requirements and provides adequate closed loops (Security as a Service - SECaaS), leveraging the slice template and the Security Service Level Agreement (SSLA). SO instantiates and deploys SECaaS while relaying on the SSLA manager to monitor and check that the security requirements of network slices are met. SECaaS is defined as a selected combination of Monitoring System (MS), Analytical Engine (AE), and Decision Engine (DE) deployed on-demand to provide zero-touch management (ZSM) of security. SECaaS features AI algorithms at the AE that automatically protect the running network slice by detecting anomalies and attacks and mitigating these threats with appropriate action through DE. Besides, the MonB5G system focuses on protecting AI-driven solutions run at AE against the presence of misbehaving decentralized elements, knowing that the produced model can provide a calamitous output when trained based on corrupted data. Hence, the design of new approaches to secure AI-based training is envisioned.

This deliverable provides a detailed technical description of the SECaaS components, including ML algorithms, designed to address challenging security tasks related to three representative use-cases: aLTEr attacks, in-slice massive Machine Type Communications (mMTC) attacks, and poisoning attacks on training a Federated Learning (FL) model. The three selected use-case scenarios correspond to well-known and frequent attacks in a network slicing environment, creating the need to design a ZSM solution relying on AI/ML as designed in MonB5G.

The first scenario, aLTEr attacks, aims to show the functionalities of SO and a detailed description of the MS, AE, and DE to detect and mitigate attacks. To recall, the aLTEr attack is a MITM attack type and is carried out between the user equipment (UE) and the gNodeB (gNB). It consists of breaking layer two of the user radio bearer, exploiting the user data integrity protection can be missing as a vulnerability to carry out the attack. To detect the attacks, we explored three AI/ML algorithms to detect the aLTEr attack, executed at AE, namely One-Class SVM (OcSVM), Isolation Forest (IForest), and Local Outlier Factor. The mitigation step executed by DE consists of a security policy update on the firewall to deny the private DNS address from the UE.

The second use-case scenario, mMTC attacks, aims to demonstrate the combined functionalities of MS, AE, and DE to detect and mitigate Distributed Denial-of-Service (DDoS) attacks that may originate from a subset of User Equipment (UEs) attached to a specific network slice issuing malicious traffic towards the infrastructure services. A typical example is compromised MTC devices (i.e., Internet of Things (IoT) devices) generating a massive number of network attachment requests. This type of attack targets mainly the shared network slice elements, such as the 5G Core Network (CN). To detect this type of attack we leveraged AE with a ML algorithm based on Gradient Boosting that extracts a list of UEs with a probability of being part of the attack. The mitigation executed by DE consists of de-registering the concerned UE and blocking them at the AMF/UDM 5G core functions to prevent future network registration.

The third use-case scenario, poisoning attacks on training a FL model, aims to detect and mitigate adversarial attacks, more precisely poisoning attacks that target the training phase. We assume poisoning adversarial attacks where malicious agents poison some of the parameter updates of the FL algorithm to be sent to the centralized environment. To mitigate this type of attack, we leveraged the MonB5G closed-control loop to protect the FL training process using ML algorithms. First, relaying on a Deep Reinforcement Learning (DRL) algorithm, we dynamically select a trusted participant (i.e., a FL agent), which then applies a dimensionality reduction scheme and unsupervised machine/deep learning to detect the poisoned model(s) and hence malicious participant(s). The mitigation action run by DE consists in deleting malicious participants.

The three scenarios were implemented, and several results have been extracted and presented in this deliverable. Note that details on the implementation are provided in D5.4.

TABLE OF CONTENTS

List of Figures

List of Tables

Abbreviations

ACT	Actuation component
ADAM	ADAptive Moment estimation
AE	Analytic Engine
AI	Artificial Intelligence
AMF	Access and Mobility Management Function
ANN	Artificial Neural Network
APEX	Adaptive Policy EXecution
API	Application Programming Interfaces
B5G	Fifth-generation and Beyond
CISM	Container Infrastructure Service Management
CN	Core Network
CNI	Container Network Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRD	Calico Custom Resource Definition
D-ASO	Domain Autonomic Security Orchestrator
DCI	Data Centers Interconnect
DDoS	Distributed Denial-of-Service
DE	Decision Engine
DL	Deep Learning
DMO	Manager and Orchestrator
DNS	Directory Name Service
DPI	Deep Packet Inspection
DQN	Deep Q Learning
DRL	Deep Reinforcement Learning
DSM	Domain Slice Manager
E2E	End-to-End
EM	Element Manager
eMBB	enhanced Mobile BroadBand
eNB	eNodeB
FL	Federated Learning
gNB	gNodeB
GTP	GPRS Tunneling Protocol
HTTP	Hypertext Transfer Protocol
IDMO	Inter-Domain Manager and Orchestrator
IDS	Intrusion Detection System
IMSI	International Mobile Subscriber Identity
IoT	Internet of Things

KNN	k-nearest neighbors
KPI	Key Performance Indicator
L-ASO	Local Autonomic Security Orchestrator
LCM	Life-Cycle Management
LDA	Linear Discriminant Analysis
MANO	Management and orchestration
MCC	Mobile Country Code
MDP	Markov Decision Process
MEC	Multi-access Edge Computing
MitM	Man in the Middle
ML	Machine Learning
MME	Mobility Management Element
mMTC	massive Machine Type Communications
MNC	Mobile Network Code
MQTT	Message Queuing Telemetry Transport
MS	Monitoring System
MSE	Mean Squared Error
MTTD	Mean Time to Detection
MTTR	Mean Time to Respond
NAS	NAS message
NFV	Network Function Virtualization
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
NG-RAN	Next Generation-Radio Access Network
NIDS	Network Intrusion Detection Systems
NS	Network Slice
NSMF	Network Slice Management Function
NSSMF	Network Slice Subnet Management Function
NST	Network Slice Template
NTP	Network Time Protocol
OAI	OpenAirInterface
OcSVM	One-Class SVM
OSS	Operations support systems
PaaS	Platform as a Service
PCA	Principal Component Analysis
PCO	Protocol Configuration Options
PDU	Protocol Data Unit
PNF	Physical Network Function
QoS	Quality of Service
RAM	Random Access Memory
RAN	Radio Access Network
RBSs	Rogue Base Stations

RL	Reinforcement Learning
RNN	Recurrent Neural Network
RRC	Radio Resource Control
Sc-MB5G	Security services - MonB5G
Sc-Or	Security services - NFVO
Sc-Vnf	Security platform - NFV
SDN	Software Defined Networking
SECaaS	Security Service Platform
SecPaaS	Security Platform as a Service
SGD	Stochastic Gradient Descent
SLA	Service Level Agreement
SMF	Session Management Function
SML	Service Management Layer
SO	Security Orchestrator
So-Os	SO - OSS
SSLA	Security SLA
SUPI	Subscriber Permanent Identifier
SVM	Support Vector Machines
SYN	SYN protocol
TCP	Transmission Control Protocol
UDM	Unified Data Management
UDP	User Datagram Protocol
UE	User Equipment
UPF	User Plane Function
uRLLC	ultra-Reliable Low Latency Communications
VM	Virtual Machine
VNF	Virtual Network Functions
VSF	Virtual Security Function
ZSM	Zero-touch network and Service Management

1. Introduction

1.1 Scope

MonB5G's objective is to define a novel autonomic management and orchestration framework by leveraging the distribution of operations together with state-of-the-art AI-based mechanisms for building zero-touch security management and orchestration. The MonB5G approach focuses on the design of a hierarchical, fault-tolerant, automated, and data-driven network management system that incorporates security as a key feature in order to orchestrate a high number of parallel network slices and significantly higher types of services in an adaptive and zero-touch way. MonB5G defines the Security Orchestrator (SO) as the key element that defines SECaaS, a distributed approach that uses the MonB5G key components Monitoring System (MS), Analytic Engine (AE), and Decision Engine (DE), targeting a distributed and zero-touch security management.

The deliverable provides the MonB5G's activities towards reaching zero-touch security management relying on AI, which have been conducted during tasks 5.1 and 5.2. The deliverable complements D5.1 by focusing on the technical solutions that combine MS, AE, and DE, to detect and mitigate attacks. First, the deliverable recalls the MonB5G security architecture featuring the SO entity, the SECaaS concept, and the MonB5G key components MS, AE, and DE. This architecture was first introduced in D5.1 and then improved in D2.4. Second, the deliverable details three solutions demonstrating zero-touch security management and orchestration based on MonB5G critical components, i.e., SO, MS, AE, and DE. Two solutions identified and introduced in D5.1 are detailed with more technical content, including the AI/ML algorithms and test results aiming to demonstrate zero-touch security management to detect and mitigate attacks from and on network slices, covering mMTC and aLTEr attacks. The third solution aims to detect and protect ML training against poisoning attacks, focusing on the case of the FL training process by using the MonB5G component. Also, details on AI/ML algorithms as well as test results are given.

1.2 Structure

The main structure of this deliverable is summarized in the following table, linking each section with the corresponding task.

Section	Description	Task(s)	Starting Month
2	Recalls the security architecture and management devised in MonB5G. It summarizes the role and functions of the Security Orchestrator as well as the SECaaS components.	T5.1 & T5.2	M13

3	Presents the MonB5G adopted solution to detect and mitigate mMTC in-slice attacks. The proposed approach features the MS, AE, and DE components and demonstrates how they interact to protect 5G Core Network components against mMTC DDoS attacks. The section presents test results and compares the proposed solution performances against a statistical-based approach.	T5.1, T5.2 & T5.4	M13
4	Presents the MonB5G solution to detect and mitigate aLTEr attacks. The proposed approach features the SO functions as well as the SECaaS to detect and mitigate attacks. The section presents test results.	T5.1, T5.2 & T5.4	M13
5	Introduces alternate AE solutions for aLTEr and mMTC attacks aiming to provide a generalized solution. It includes other ML approaches for AE different from those detailed in sections 3 and 4 while providing a deeper analysis of the available data sets.	T5.2 & T5.4	M13
6	Presents a zero-touch approach to detect and mitigate poisoning attacks on Federated Learning training. The proposed approach is validated when training a FL model to predict AMF performances involving a number of parallel network slices.	T5.1, T5.2 & T5.4	M13

2. Security Management in MonB5G

2.1 MonB5G Reference architecture instantiated for security management

MonB5G vision is to manage a massive number of slices using fine-grained management services tailored to 5G network slice particularities. Slices in 5G are logically isolated networks providing different services with distinctive network requirements achieved by varying nodes, resources, localisation, etc. This heterogeneity requires specific management per slice. Based on these facts, in this section, we will instantiate the MonB5G reference architecture described in D2.4 for security management.

2.1.1 NETWORK SLICING THREATS AND SECURITY REQUIREMENTS MANAGEMENT

In 5G network slicing, every slice is a unique composition of nodes and requirements. Hence, it has its own threat set that can be identified based on a threat model to select the slice-specific security enablers. The threats depend on the service type. For instance, an Internet of Things (IoT) slice will have different threats when compared with a streaming slice. For this reason, we consider selecting the security requirements for the slice dynamically before the deployment based on its template and, accordingly, its specific security management.

The main threat to network slicing is the weakness of isolation in terms of both resources and security. While the lack of resource isolation might lead to resource starvation, the weak security isolation facilitates attacks between the coexisting slices, for instance, a side-channel attack [1].

However, the slice spans over multiple technological domains composing the 5G network (e.g. Core Network (CN), Multi-access Edge Computing (MEC), Transport Network and Radio Access Network (RAN)) that are exposed to distinct threats:

- The core functions of the Core Network are IP-based services, making them vulnerable to all IP-based attacks, such as Distributed Denial-of-Service (DDoS) attacks.
- Furthermore, the core leverages a set of technologies such as Network Function Virtualization (NFV), Software Defined Networking (SDN), etc., which might be immature and bring a new set of threats.
- The use of edge computing creates new cybersecurity concerns that require dedicated security. Since the network functions hosted on the edge will not be exposed to the same physical and virtual threats as those hosted on the core. To clarify, apart from the physical threats in the user-close management, the edge might act autonomously to avoid the backhaul delay leveraging local services (i.e authentication) which rise new challenges.
- RAN threats are often related to user authentication and authorization systems, such as Rogue Base Stations (RBSs) that can steal user data, track or redirect the user to a malicious endpoint.

The intersection of both end-to-end slice threats and domain threats results in distinct security requirements to be delegated to the sub-slice security manager. In contrast to the end-to-end slice management that does not consider the requirements of the technological domain, or the domain centralized management that lacks strong isolation between sub-slices, the sub-slice dedicated security manager will be efficient and optimized to target the specific sub slice threat related to the hosting technological domain needs and also

maintains the isolation between sub-slices. Yet the sub-slice security manager cannot see the other domain sub-slices (cross sub-slices management) requirements that cannot be omitted but will be centralised per domain.

2.1.2 *SECURITY MANAGEMENT ARCHITECTURE AND COMPONENTS*

In the D2.4, the MonB5G reference architecture was described in detail. In this section, we will describe the main security management components. MonB5G security framework (Figure 1) concerns multiple levels of the management and orchestration hierarchy. The global security is managed using the E2E Autonomic Security Orchestrator (ASO) located in Inter-Domain Manager and Orchestrator (IDMO), besides the security-related management loops. However, at the domain level, the security management is hosted in Domain Manager and Orchestrator (DMO), where we have a Domain Autonomic Security Orchestrator managing the security closed loops and the security enablers (Virtual Security Functions). When managing security goals or intents for multi-domain network slices, Network Slice Management Function (NSMF) (located in IDMO), and when managing security goals or intents for single-domain network slices, Network Slice Subnet Management Function (NSSMF) (located in DMO), delegate some control to the Local Autonomous Security Orchestrator at the slice management level, i.e., to SML.

Security control functions and the core functions of the cybersecurity framework are provided by a Security Service Platform (SECaaS). As a set of Virtual Network Functions (VNFs), SECaaS can be dedicated to the safeguarding of a network slice, or it can be shared. The application of the security framework at this layer requires the presence of a security platform in the Local Autonomic Security Orchestrator (L-ASO), Domain Autonomic Security Orchestrator (D-ASO) and E2E-ASO. Indeed, SECaaS of IDMO manages the security lifecycle globally from the framework core function and identifies the necessary framework core function respond/recover operations. SECaaS of D-ASO can monitor multiple slice instances and amplify the signal to detect a threat that is transparent to every individual network slice. SECaaS of L-ASO provides security management functionalities within a given slice. Another valuable point is the sharing of information between network slices to reinforce protection. In addition, a Security Platform as a Service (SecPaaS, and implementation of PaaS) can be dedicated to a group of network slices according to some criteria: per tenant, vertical, slice type, and security level. The isolation between SECaaS and the security orchestrator allows the possibility of employing multiple strategies and policies to manage the security lifecycle.

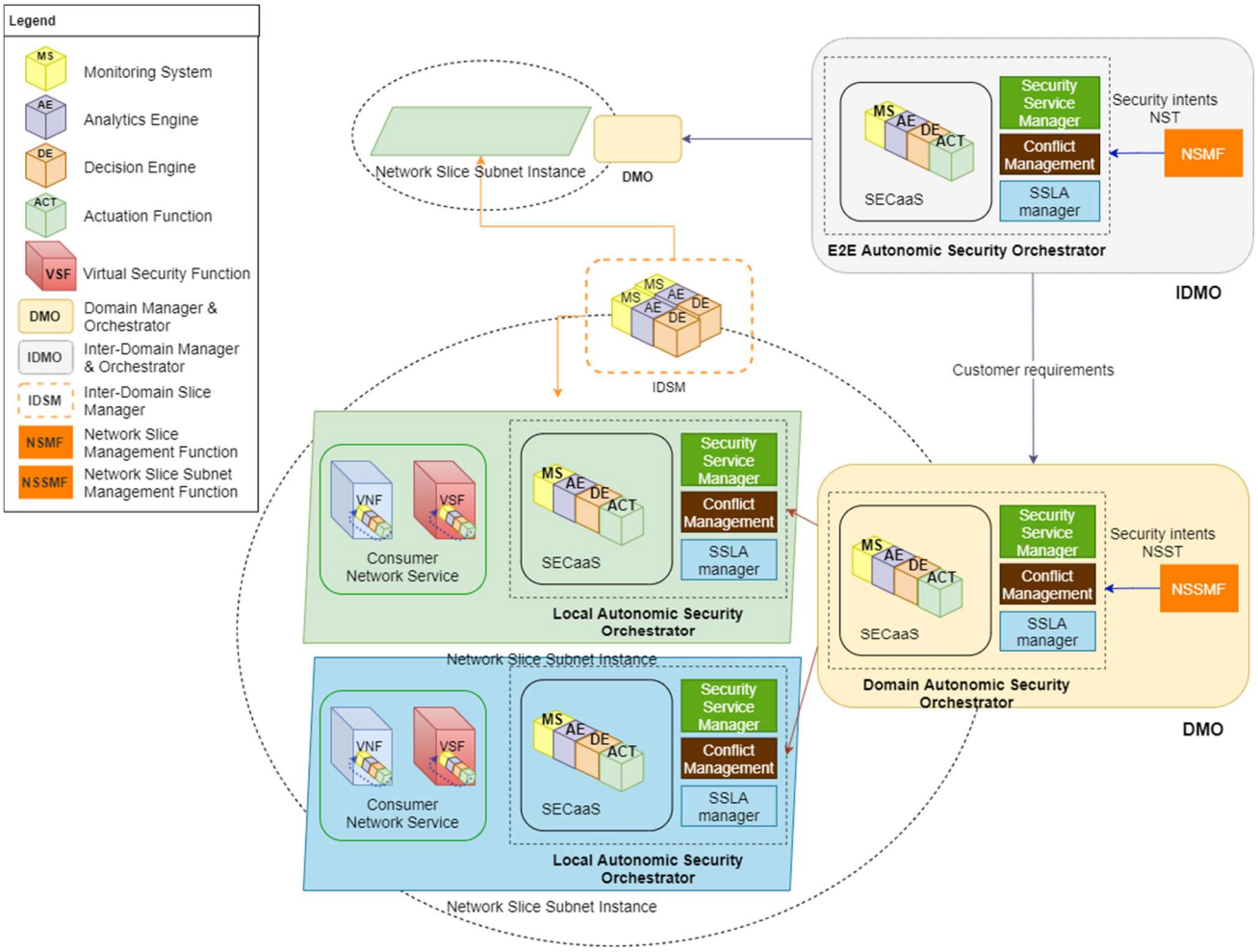

Figure 1. Security components of the architecture.

At the inter-domain level, that of network slice, which aggregates several slice subnets to offer E2E services, a SECaaS is also in place to manage the security lifecycle of network slice instances. From the customer's security intent, SECaaS of NSMF can extract and distribute goals to each NSSMF. In terms of detection, its view over the network segments, allows to gather security reports and detect threats. As a response, for instance, a change of security requirement is sent to NSMF of a network slice subnet instance.

2.1.3 Security orchestration

Security is distributed with slice instances implementing the required security management closed loops following the SECaaS paradigm. The closed loops and the security enablers used to enforce the security policies are managed through the "Security Service Manager" component of the security orchestrators (SOs) deployed at E2E, domain, and slice levels, namely the E2E Autonomic Security Orchestrator (E2E-ASO), D-ASO, and L-ASO. In what follows, we will elaborate on the security orchestration capabilities of the three types of SOs.

2.1.3.1 E2E AUTONOMIC SECURITY ORCHESTRATOR

E2E-ASO (part of IDMO) has a global view and is responsible for the security of all slices from the E2E perspective. At the slice creation phase, E2E-ASO checks (via communicating with domain SOs) whether the slice can be deployed at the agreed Security Service Level Agreement (SSLA) requirements depending on the security capabilities provided by the respective technological domains where the slice will be deployed. If feasible, the E2E-ASO identifies the security policy to enforce and the corresponding enablers to deploy based on the slice blueprint, the received SSLA and the performed risk assessment. During the slice runtime, E2E-ASO is responsible for enforcing cross-domain security decision policies coming from IDMO closed loops, such as migrating a sub-slice to a new domain to avoid a security threat in the original domain [2].

D-ASO manages those IDMO closed loops and might coordinate and assist the local security orchestrator in selecting SECaaS to be deployed with each domain slice (sub-slice). E2E-ASO stores the policies enforced from his level into the conflict manager that permits to check any conflicts.

2.1.3.2 DOMAIN AUTONOMIC SECURITY ORCHESTRATOR

Local to every domain, D-ASO (part of DMO) ensures the local security management, by instantiating and managing the appropriate closed loops using a SECaaS approach when deploying a sub-slice instance. The adoption of the SECaaS model allows reusability of the deployed closed loops or some of their components (i.e., MS, AE, DE and ACT) between sub-slices. D-ASO ensures that the reusability is performed according to the isolation level of slices. The security policies issued by DEs and their enforcement status are stored and managed by the “Conflict Management” component, providing D-ASO an overall view of all security policies enforced within the domain’s sub-slices and allowing to avoid any conflict between policies. Once issued, the security policy is saved with “Enforcing” status. Its status is changed to “Success” if the ACT component can correctly execute the policy’s actions, otherwise it is set to “Failure”. D-ASO ensures the successful enforcement by adjusting the security policy and executing the necessary corrective operations. If the mitigation of the security issue is not possible at the domain level, D-ASO escalates the problem to E2E-ASO, which assesses the problem and generates the mitigation actions in E2E scope [2].

2.1.3.3 LOCAL AUTONOMIC SECURITY ORCHESTRATOR

L-ASO is located at the slice level, which is responsible for managing the internal security of a sub-slice. L-ASO is in charge of communicating with the sub-slice VNFs to monitor, analyse local data and take mitigation actions directly within the sub-slice scope. In case the mitigation is not possible, L-ASO informs D-ASO in order to take adequate actions to address the security issue.

Similarly, to E2E-SO, D-ASO has a SSLA manager that interprets the SSLA requirements into security actions and Key Performance Indicator (KPIs) while deploying a slice to monitor and ensure that the tenant security requirements are met.

2.1.3.4 SECURITY AS A SERVICE COMPONENT

Our architecture is strongly based on the triplet of components consisting of the MS, AE, and DE. At certain levels, a component called actuator (ACT) is added. These components provide security as a service (SECaaS) and can be shared between multiple slices. As we expect to have different security and management components, we use the layering concept for reusability and simple communication. To clarify, all MSes are connected in a layer (Monitoring System Sublayer), and similarly, we have an Analytic Engines Sublayer and Decision Engines Sublayer. A vertical channel connects the three sublayers; for example, an AE can request

additional data from other MSes. Also, this channel opens an interface between the other hierarchical management layers, so that any management element can have access to any service.

- The Security AE subscribes to MS the real-time security-relevant data generated by MS. The data sources are identified depending on the hierarchical level and the final objective of DE. MS pre-processes the collected data.
- The Security AE processes the security-related monitoring data in order to provide high-level security information and events. Analysing the collected KPIs, network flows and resources status will help in diagnosing the node and the network to detect or predict attacks and security issues.
- The Security DE is the mastermind that has the ability to tell the system what to do as a reaction or prevention to protect the network against security threats. The Security DEs' role is crucial in detecting attacks and deciding on dynamic security policy per slice and per attack episode. The decision can configure an existing security enabler in the slice or deploy a new one; however, these decisions are described in an abstract model rather than vendor-specific. Among DEs distributed horizontally and vertically, each can be related to an application/VNF, domain slice, or an E2E slice scope. DEs are atomic elements that implement a specific autonomic security function. For instance, at the VNF level, the embedded DE is responsible only for a unique security threat that may target its hosting VNF. In this case, the security vulnerability or the potential attack should be known earlier (before deploying DE) based on the application running or the protocols which may differ from one VNF to another. At the higher level, the slice-attached DEs depend on the slice threats and characteristics. Similarly, for slice DEs deployed in IDSM. This distribution will greatly simplify autonomic threat detection and fast, local response.
- The Actuation component (ACT) is in charge of executing security policies. First, it performs a translation from the high level into vendor-specific configuration according to the targeted enablers. ACT can trigger deploying a specific Virtual Security Function (VSF) through the NFV MANO or update the configuration on an existing VNF/VSF.

2.1.4 *SECURITY ORCHESTRATOR INTERFACES*

The following reference points are defined for SO and the security platform SECaaS at the inter-domain, domain and functional layers (cf. Figure 2):

- **So-Os**: The reference point between SO and OSS is used by OSS to delegate the management of the security goals of the network slice to SO. This is the IDMO/DMO internal interface.
- **Sc-MB5G**: The reference point between the security services and the MonB5G sublayer components is used to interact with MonB5G components to create closed loops for the security process.
- **Sc-Sc**: The reference point is used for the control communications between two security services. The E2E security service has a global view over the activities of slice subnet security platforms and manages the E2E security requirements. In turn, the slice subnet security service controls the functional security platforms to ensure that each slice subnet instance is well protected.
- **Sc-Or**: The reference point between the security platform and NFV Orchestrator (NFVO). It is used to monitor the health of NFV objects and perform management operations on them. It is related to the reference point Sc-Or as defined in ETSI GS NFV-IFA 033 [3].

- **Sc-Vnf**: The reference point between the security platform and the consumer NFV object. The functional security platform offers VSFs such as firewall, Intrusion Detection System (IDS), access management as protection measures to improve the security of the slice subnet instance. This is a slice internal interface.

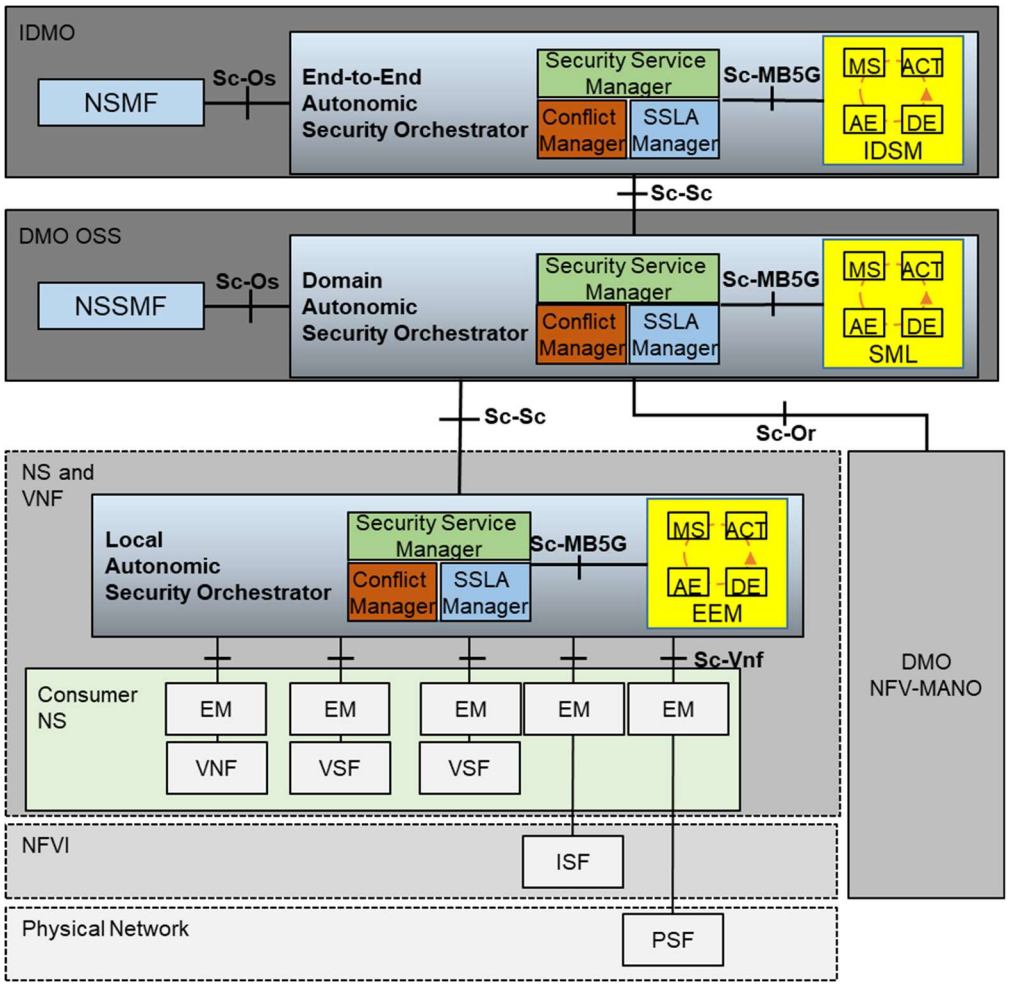

Figure 2. Reference points of the security orchestrator.

2.2 AI-driven security through MS/AE/DE

MonB5G advocates for LCM (Life-Cycle Management) procedures mainly based on ML and AI and relies on three distinct entities that collaborate, namely, MS, AE, and DE. The combination of the three elements allows enforcing the concept of ZSM using ML. MS and AE are used to monitor the system's functioning and detect security threats, while DE will consider actions to mitigate threats and avoid that the system fails. In the context of network slicing, ZSM is highly needed to mitigate the high complexity of managing network slices, as several components of different technological domains (VNF, Physical Network Function (PNF), MANO, NFV Infrastructure (NFVI), RAN, CN, etc.) are involved; this increases the number of threats that may come

from several sources. MonB5G mitigates this complexity with the usage of MS/AE/DE. MS is deployed to monitor key KPIs, logs and events from different domains, AE correlates and analyses the monitored data to detect threats, and DE considers and enforces multi-domain decisions.

In this section, following D5.1 and D2.1, we recall each entity's functioning and its role in the AI-driven security solution provided by MonB5G. More details on the three MonB5G key entities are available in D2.1.

2.2.1 *MONITORING SYSTEM (MS)*

The role of the MS in MonB5G is to collect critical information on the functioning of a system and provide this information, after, for example, aggregation or normalization, to AE, which in turn uses this information to detect and react to Network Slice (NS) LCM events, such as performance degradation, performance optimization, and security threats.

MS interacts with different entities that orchestrate and manage the end-to-end NS, i.e., different DMOs. Besides, MS interacts with slice-specific VNF and applications, as well as shared VNF and PNF among network slices.

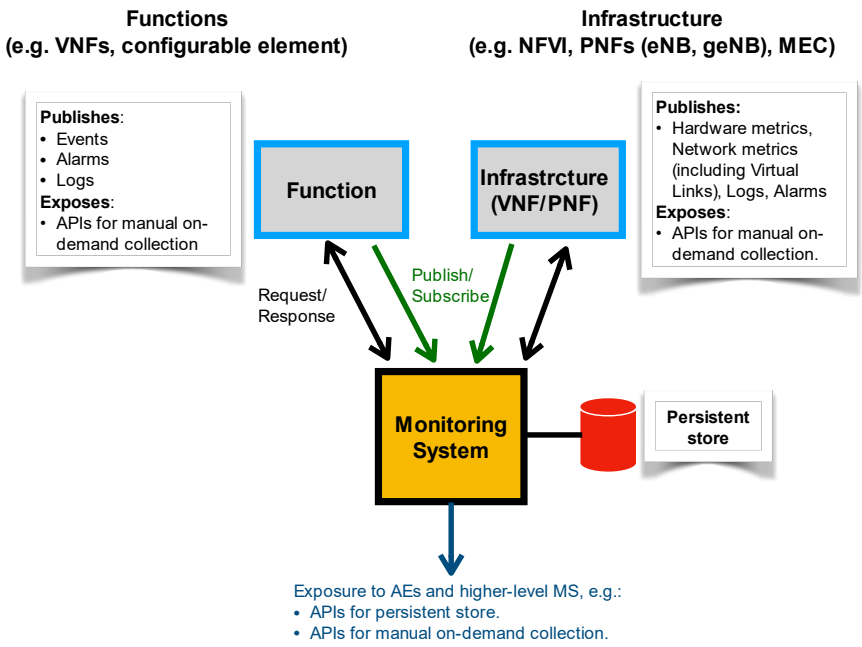

Figure 3. MS interaction.

As depicted in Figure 3, we distinguish between information that monitors the state of the infrastructure shared by the running slices and the information that monitors the VNF of tenants and applications state.
For the infrastructure monitoring, MS has to interact with DMO(s) to collect information on:

- NFVI: such as computing platforms and hardware;
- PNF running network functions on dedicated hardware: such as eNB/geNB, router, and UPF;
- VNFs running common virtualized network functions: such as Core Network (CN) functions or Directory Name Service (DNS).

Regarding Function monitoring, MS has to interact with the VNFs or applications of the tenants. Since there is no standard way to enforce such interaction, MonB5G may dictate design guidelines that would allow MS to request and collect metrics from VNFs or applications. In this case, MS may collect all information considered as service-level metrics, such as Events, Alarms, and Logs.

The principal consumer of MS information is AE, which is in charge of triggering the monitoring of needed information from MS. The latter starts the monitoring process by connecting to the appropriate source, i.e., infrastructure and Function. Accordingly, MS exposes two types of Application Programming Interfaces (API): control API and the data collection API. Control API may be used by AE to request metrics to monitor, the periodicity, the duration, etc. While the data collection API is the interface from which data are provided to AE as requested through the control API. The control API also indicates how data are provided, i.e., publish/subscribe, request/response, the data format, etc.

Particularly, in the context of security, MS is envisioned to monitor metrics on the computing and network resource consumption of VNFs and PNFs to detect overconsumption of resources, which may indicate abnormal functioning of the VNF or PNF. Besides, MS should be able to monitor service-level KPI of VNF or PNF, such as the attach and detach requests of UEs at the Mobility Management Element (MME), which can be used to detect DDoS attacks.

2.2.2 *ANALYTICAL ENGINE (AE)*

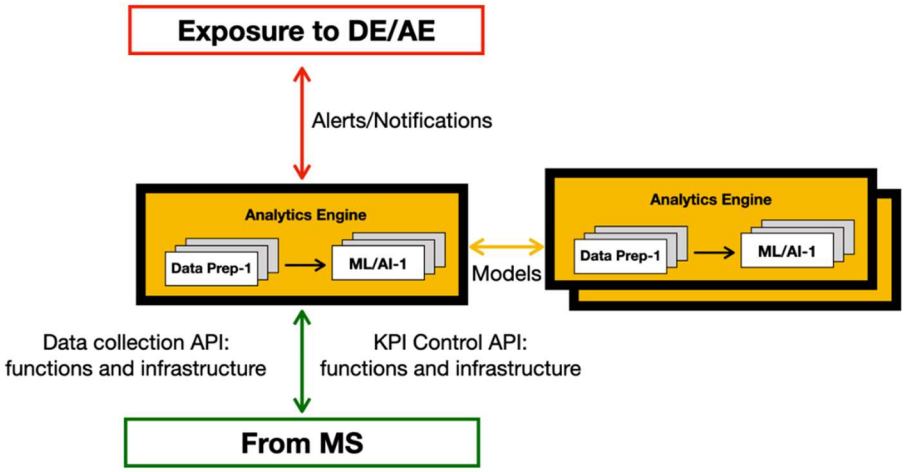

Figure 4. AE interactions.

As opposed to MS, AE does not store but processes data gathered from the same or lower-level MS or AE and exposes the result to any requester (i.e., DE or other AE) in an on-demand or periodic fashion. AE to AE communication is possible, mainly by building a learning model using federated learning techniques. Figure 4 illustrates the interaction between AE and MS and between AE and DE.

Generally, the main functions of AE are: (i) identify performance degradation of a network slice; (ii) optimize the performance of a network slice or the DMO resources; (iii) react to security threats. To this aim, AE subscribes for data types to which it is interested in using the control API exposed by the MS. The data type will be determined according to the logic of the LCM application execution. Then, AE starts receiving the

stream of data or uses a request/response mechanism, depending on the purpose of the analysis. AE may adapt the monitoring data rate or stop the request and request for other related monitoring information.

AE is able to complete an inference task locally, extract features, and to analyse these features and send alerts and notifications to DE. AE may collaborate with other AE to build distributed learning (based on federated learning) model to realize the analysis and notify the DE accordingly.

In the case of AI-oriented security, AE has a crucial role. AE collects data from several MS, covering a wide range of information coming from different domains. The collected data is correlated and interpreted by AE, using ML algorithms. An example of AE relying on ML is anomaly detection using neuronal networks or a classifier.

2.2.3 *DECISION ENGINE (DE)*

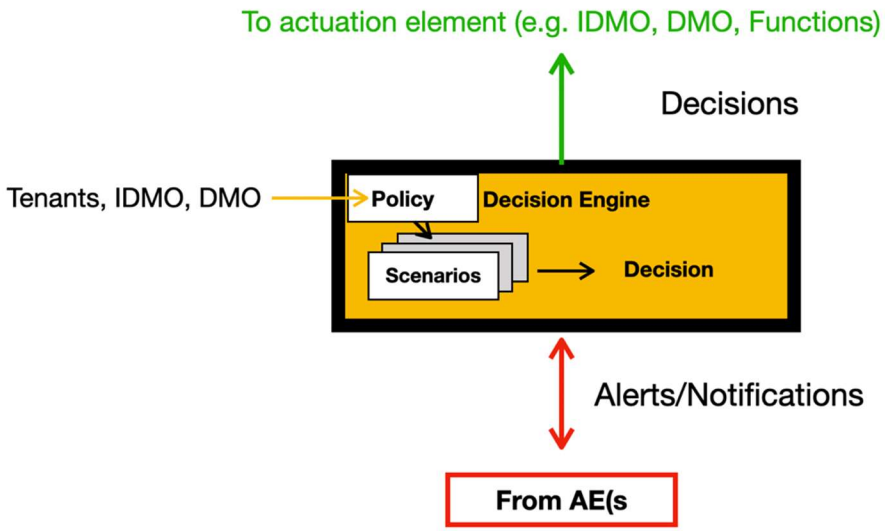

Figure 5. DE interactions.

As depicted in Figure 5. DE interactions, DE is the decision-making element of the MonB5G architecture. It analyses alerts and notifications from AE(s) and considers a decision to take. The decisions are either derived using a local ML algorithm, based mainly on Reinforcement Learning (RL), or a predefined policy enforced by the Tenant or DMO through Intent, or a combination of both.

DE may collect notification from several AEs, which may interact with MS monitoring different TDs to consider a global decision on the end-to-end NS. Global decisions are mainly considered at the IDMO level.

DE interacts with actuation elements (function, DMO, or IDMO) to enforce the considered decisions. For local decisions, DE interacts with the DMO and function; while for global decisions, DE has to interact with IDMO.

3. MonB5G AI-driven security techniques: Attack identification and mitigation

3.1 In-Slice attack mitigation: case of MME/AMF

In recent years, computing has evolved to support the growing market of IoT, which involves pieces of hardware, such as sensors, smart door locks, smartwatches, etc., having the ability to connect to the network, transmit and receive data. A recent forecast estimate that the number of IoT devices that should be connected to the network will exceed 25.4 billion by 2030 [4].

In terms of connectivity to the network, several technologies are candidates to connect these devices. We can mention LORA, SigFox, IEEE 802.14.5, and 5G. The latter is seen as an excellent alternative as it allows continuous connectivity of the IoT devices, supporting even mobile devices (i.e., embedded in cars or drones). 5G is the latest generation of mobile networks, is promising the support of several new emerging services, including those relying on IoT. 5G uses the concept of network slicing to efficiently manage the common infrastructure and provision of network resources tailored to the network service needs. A network slice is a virtual network built using VNFs that runs as Virtual Machine (VM) or container on top of cloud infrastructure (central or edge) and PNFs such as 5G base stations. A network slice is composed of different sub-slices that span different technological domains, radio, core network, cloud, and transport. Sub-slices are stitched together to build the end-to-end network slice tailored to the application. Sub-slices can be specific to the owners (ex., VNF running applications or network services) or shared among slices (ex., 5G CN, gNB).

5G envisions the running of network services relying on IoT as a network slice of massive Machine Type Communications (mMTC) type. As its name indicates, this type of network slice is intended and optimized to connect a massive number of IoT devices to a service or an application. It is worth noting that 5G also specifies two other types of network slice, namely ultra-Reliable Low Latency Communications (uRLLC) and enhanced Mobile BroadBand (eMBB), which are envisioned for low-latency-demanding applications (e.g., autonomous driving, Industry 4.0) and high bandwidth-demanding applications (e.g., Virtual and Augmented Reality), respectively. Last but not least, 5G supports multi-tenancy, as network slices run in parallel, and a high degree of isolation is ensured among them; a network slice component (control or data plane) cannot access other network slice components. In this work, we focus on mMTC network slices as they are used to deploy IoT applications and services using MTC devices.

Meanwhile, the growing market of IoT devices has increased cyber security threats and widened attack surfaces; hence securing modern networks is a challenging task. Indeed, IoT devices are often weak when it comes to security, considering: (1) their usual low-cost and the lack of the necessary built-in security controls to defend against threats; (2) their constrained environment and limited computational capacity. Besides, they are a high-value asset to attackers. Indeed, finding a vulnerability of a type of device allows attackers to infect more devices of that type and, hence, conduct attacks from the infected equipment. For example, in 2016, a Malware called Mirai infected between 800,000 and 2.5 million devices through default passwords and used them to conduct DDoS attacks against some public web applications.

5G was designed with built-in security controls to address many of the threats found in 4G/3G/2G networks, such as enforcing mutual authentication to prevent fake base-station attacks, encrypting subscriber identities, and enforcing more secure cipher suites. However, more functionality always comes with new risks

and attack vectors when opening the network to IoT devices, particularly. Some of these risks can affect the performance (or the service availability) of 5G network functions. By embracing network slicing, 5G networks can mitigate inter-slice attacks as isolation is one of the key features. Indeed, the isolation guaranteed by Network Slicing offers performance guarantees to the applications and ensures isolation, such that attacks (e.g., leakage, breach, Distributed DDoS) remain contained and do not propagate to the network. However, an in-slice attack may correspond to a subset of the UEs attached to a specific network slice issuing malicious traffic towards the infrastructure services. A typical example is compromised MTC devices (i.e., IoT devices) generating a massive number of network attachment requests. These attacks need to be quickly identified and mitigated to avoid system failure. It is worth noting that 3GPP stated clearly in [5] that 5G networks should be protected from DDoS attacks, where mechanisms detecting and mitigating this type of attack are needed.

In this section, we propose a ZSM solution that addresses the challenges of in-slice DDoS attack detection and mitigation, considering the case of mMTC network slices. Generally, this type of attack targets the 5G CN elements shared among the network slices. The proposed ZSM solution relies on the MonB5G closed-control triplet (MS, AE, and DE) that interacts with the 5G CN in order to detect attacks and automatically react by mitigating the attacks. The critical challenge addressed in this work is how to detect a DDoS attack initiated by a compromised set of MTC devices inside a network slice. Indeed, there are no available traces or datasets that reproduce abnormal traffic on 5G, unlike other types of networks where many datasets are available. Besides, it is very challenging to detect during an event whether an attack is occurring and which devices are involved. To overcome these limitations, we followed the 3GPP traffic models [6] [7] for MTC to identify normal traffic and train an ML model to distinguish normal traffic from abnormal traffic. Our solution is to wait until an event has ended before analysing the traffic (normal or abnormal) and deducing whether an attack occurred and which devices were involved. Then, all the devices will be detached and banned from future access to the network. We believe our solution is pertinent for mitigating DDoS attacks as the latter loses its intensity if the number of involved devices is reduced by detaching and blocking them.

The contributions of this section are:

- A novel closed-control loop featuring AI/ML to achieve ZSM objective in 5G and beyond network,
- A novel approach that detects MTC attach events over time,
- A novel ML algorithm that leverages Gradient Boosting to learn and predict an upper bound interval for normal MTC traffic (following a β(3,4) as per the recommendation of 3GPP),
- A novel DDoS detection algorithm that defines the detection rate of all MTC devices involved in an attack,
- A novel mitigation algorithm that detaches and blocks all MTC devices that have been suspected to be part of an attack,
- An implementation of the ZSM concept (i.e., closed-control loop) has been done on a 5G facility to prove the concept and evaluate the performance of the detection algorithm.

3.1.1 Background and related work

In this section, we provide the necessary background and related work to understand the concepts. First, we start highlighting the 5G thread landscape. Second, we describe how ML is used to detect different types of threats. The section ends with an examination of 5G DDoS attacks that rely on IoT devices.

3.1.1.1 5G THREAT LANDSCAPE

A 5G network is composed of four major technological domains: the Radio Access Network (RAN), Core Network, transport network, and inter-connect network [8]. From a security perspective, there are threats that affect all of these planes, and others that affect only specific ones. Threats in 5G can be classified according to the parts (or technological domain) of the network they are impacting [9]:

- UE threats: Mobile botnets can launch DDoS attacks on multiple network layers, impacting the 5G infrastructure, web servers, and other UEs. The goal is to bring services offline.
- RAN Threats: Rogue base station threats for conducting Man in the Middle (MitM) attacks, this class of attacks can compromise user information, break privacy and track users, and cause a Denial of Service.
- CN Threats: These attacks relate to elements of the Core Network, including SDN, NFV, and Network Slicing. These attacks can lead to Denial of Service, eavesdropping, interception or hijacking [10].
  - Network Slicing Threats: attacks that can break the isolation between network slices [9].
- SDN Threats: Separation of control and user (data) plane that allows a malicious user to attack the link between the control and the user planes, a DDoS attack could be performed, or control could be gained over network devices, an example threat is Topology Poisoning attacks [11].

3.1.1.2 MACHINE LEARNING FOR THREAT DETECTION

ML has gained a wide interest in variety of applications and fields of study, particularly in cybersecurity. With hardware and computing power becoming more accessible, ML methods can be used to analyse and classify bad actors from a huge set of available data. There are hundreds of ML algorithms and approaches, broadly categorized into supervised and unsupervised learning [12].

Supervised learning approaches are done in the context of Classification where input matches to an output, or regression where input is mapped to a continuous output. Unsupervised learning is mostly accomplished through clustering and it has been applied to exploratory analysis and dimension reduction. Both of these approaches can be applied in cybersecurity for analysing malware in near real-time, thus eliminating the weaknesses of traditional detection methods [12]. ML techniques have been used to effectively detect and mitigate DDoS attacks on networks. Successful approaches have always used a variety of features to achieve great results, and have targeted DDoS attacks that are either very popular (that have public datasets available), or that are easy to reproduce (for which creating a dataset similar to real-world behaviour is possible), most of these target common application protocols like Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

Table 1 summarizes some of the existing research in this area.

Table 1: Comparison of some DDoS attack detection solutions.

Reference	Attacks Mitigated	ML algorithms used
João Henrique Corrêa et al. [13]	TCP SYN flood	k-nearest neighbors (KNN) and Decision Trees

Rohan DDoShi et al [14]	TCP SYN flood, UDP flood, HTTP GET flood	KNN, SVM, Decision Trees, Random Forests and Neural Networks
Faisal Hussain et al. [15]	Various attacks on application protocols	RESNET
Xiaoyong Yuan et al. [16]	Various attacks on application protocols	Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN)
Maryam Ghanbari et al. [17]	Various attacks on application protocols	CNN after SVM pre-processing
Roberto Doriguzzi-Corin et al. [18]	Various attacks on application protocols	CNN

The difference in the datasets used makes it hard to compare different solutions. Moreover, accuracy levels can be misleading, because they consider that a packet is a sample in the dataset (instead of a malicious host), and with the flooding nature of the DDoS attacks considered, even a naive decision of considering that all the traffic is malicious would yield an accuracy higher than 80%. Moreover, accuracy levels can be misleading because they consider a packet to be a sample of the dataset (rather than a malicious host), and when the flooding nature of DDoS attacks is taken into account, even a naive assumption that all traffic is malicious would result in an accuracy greater than 80%. In the context of attack detection on 5G networks, there is very few publicly-available data, datasets are hard to find, and attack reproduction involves simulating a user-equipment and its communications with 5G network components, or simulating messages within the 5G core network.

3.1.1.3 DETECTION OF DDOS ATTACKS IN IOT AND 5G NETWORKS

One of the main threats in 5G network is DDoS. Therefore, many works have designed algorithms and tools to detect and mitigate DDoS attacks. They mainly rely on ML. An ML algorithm has been proposed in [19] to circumvent attacks through an automated solution in 5G networks. It is integrated with the Deep Packet Inspection (DPI) function that is used to detect unfamiliar attacks in the traffic of IoT devices, and the proposed solution does not require high-cost infrastructure. Another solution presents real-time threat detection in order to prevent DDoS attacks [20]. The solution named “Corero” would help organizations to comprehend the 5G attacks as the solution, using an automatic and real-time detection. However, the presented approach considers a very high-cost infrastructure. In [21], the authors present an IoT security framework, including Azure, AWS IoT, and SmartThings. The AWS framework supports mutual authentication of IoT devices, which can be done using certificates, groups and roles, and AWS Cognito identities. The used authorization is policy-based, with the rules mapped to each certificate, allowing the owners of IoT devices to define their own rules. All IoT traffic is encrypted to secure the communications. In [22], the authors propose deploying security monitoring methods to timely detect and prevent massive IoT devices from being attacked or controlled. It is used to prevent these IoT devices from being used maliciously, such as launching DDoS attacks. The latter causes network congestion, leading mMTC services to fail. The detection mechanism of the proposed solution is based on a randomization technique to defend against DDoS signalling attack that emerged in the new 5G RRC three-state model.

The work in [23] proposes a modified Network Intrusion Detection System (NIDS) that defends against DDoS attacks on the mobile edge of multi-tenant 5G networks. The proposed approach solves the problem by combining traditional NIDSs with a classifier, using Support Vector Machines (SVM), to obtain the needed information about the device's traffic. Two filters are implemented: the first filter is placed on the mobile edge computing servers to prevent spoofing attacks close to the source, and the second filter located on cloud servers classifies the traffic of the complete network.

Unfortunately, the few works mentioned in this section propose solutions that need new software or specific protocols, which are not standard and need high-cost implementation. Besides, they do not consider the context of network slicing, where most of the addressed challenges are solved thanks to network slice isolation. Therefore, most of the remaining security threats and challenges arise when the devices constituting the slice are compromised. In this case, shared network components are vulnerable to in-slice attacks since the devices are under the control of the tenant and have the green light to access the network. To the best of the authors' knowledge, our work is the first to address the challenge of in-slice attack detection and mitigation in 5G considering mMTC network slices.

3.1.2 Proposed approach towards Zero-touch security management for in-slice attack: Assumption & System architecture

Our solution aims to provide a solution featuring zero-touch security management of in-slice attack detection and mitigation considering mMTC slices in 5G. The proposed solution relies on machine learning to detect abnormal traffic of MTC devices that could cause DDoS on the control plane of the 5G core network (by flooding the Access and Mobility Management Function (AMF) with signalling messages). Hence, it will be possible to mitigate the attack by making efficient decisions to prevent flooding of the AMF with traffic and causing DDoS or deteriorating performance for legitimate users. This type of attack can be more effective on mMTC than other 5G services, assuming the very high number of MTC devices supposed to support.

We assume that a mMTC slice is composed of a shared sub-slice with other existing network slices that run the 5G Core Networks (including the AMF) and gNB, and a specific sub-slice to run the application that collects data from the MTC devices. It is well accepted that MTC devices generate two main types of traffic [24]. The first type is “Periodic”, where the devices emit data periodically. This corresponds to the case of meteorological data. The second type is “Event-Driven”. In this case, the devices emit data when a specific event occurs; for example, smoke (the signal of a possible fire) is detected. Detecting anomalous traffic in the first kind is simple, as most anomaly detection algorithms can be applied directly to the problem. However, for the second type of traffic, it is very challenging to predict when an event will occur consistently. While there exist models for the traffic during activity periods, it is difficult to solve the problem of finding anomalies just by trying to learn the function the data distribution follows; if we consider time-series data. To be able to detect malicious traffic of MTC devices, we will consider the traffic model proposed by 3GPP, which suggests that the traffic of MTC devices' connection after detecting an event follows the $\beta(3, 4)$ probability distribution [25]. Accordingly, we assume that normal traffic follows the $\beta(3, 4)$ probability distribution, while the detection events do not overlap (i.e., each event is independent of the other). We argue the latter assumption by the fact most devices run only one type of application that monitors a single event.

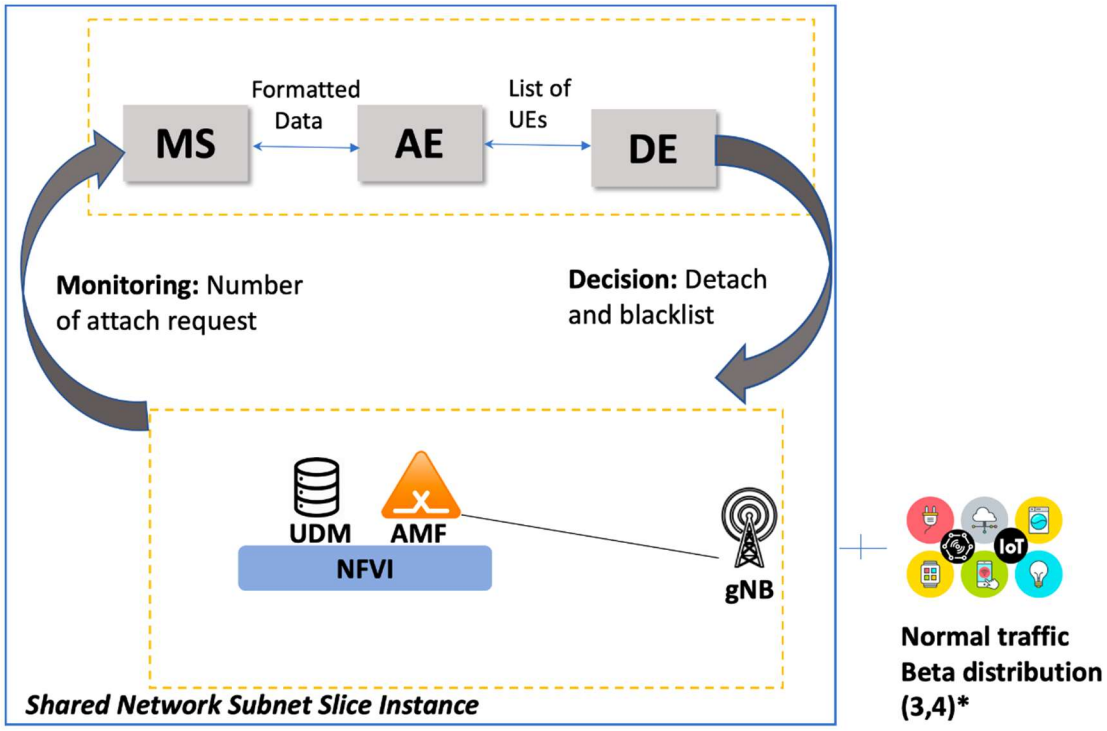

Figure 6: The high-level vision of envisioned system architecture.

Figure 6 illustrates the high-level vision of the system, which is composed of the closed-control components (MS, AE, and DE) that interact and protect the shared sub-slice components (5G CN and gNBs) against DDoS attacks. Here, we focus on protecting the AMF as it is the entry point of the 5G CN and treat all the attach requests coming from the different gNB under its control. The closed-control loop is composed of three entities: MS, which collects information from the AMF, AE, which uses ML to predict attacks, and the DE, which reacts to the alert sent by the AE by acting on the AMF (block and blacklist UE). The control-loop runs as software and can be co-located with the orchestrator managing the life cycle of the shared sub-slice [26]. It should be noted that the AMF, via an Element Manager (EM), exposes API for an orchestrator or a manager to extract and monitor information on the AMF's functioning or to change the configuration of the latter. In the proposed framework, the MS monitors the Attach Request message received by the AMF, and the DE requests to send Registration Reject message to suspected devices.

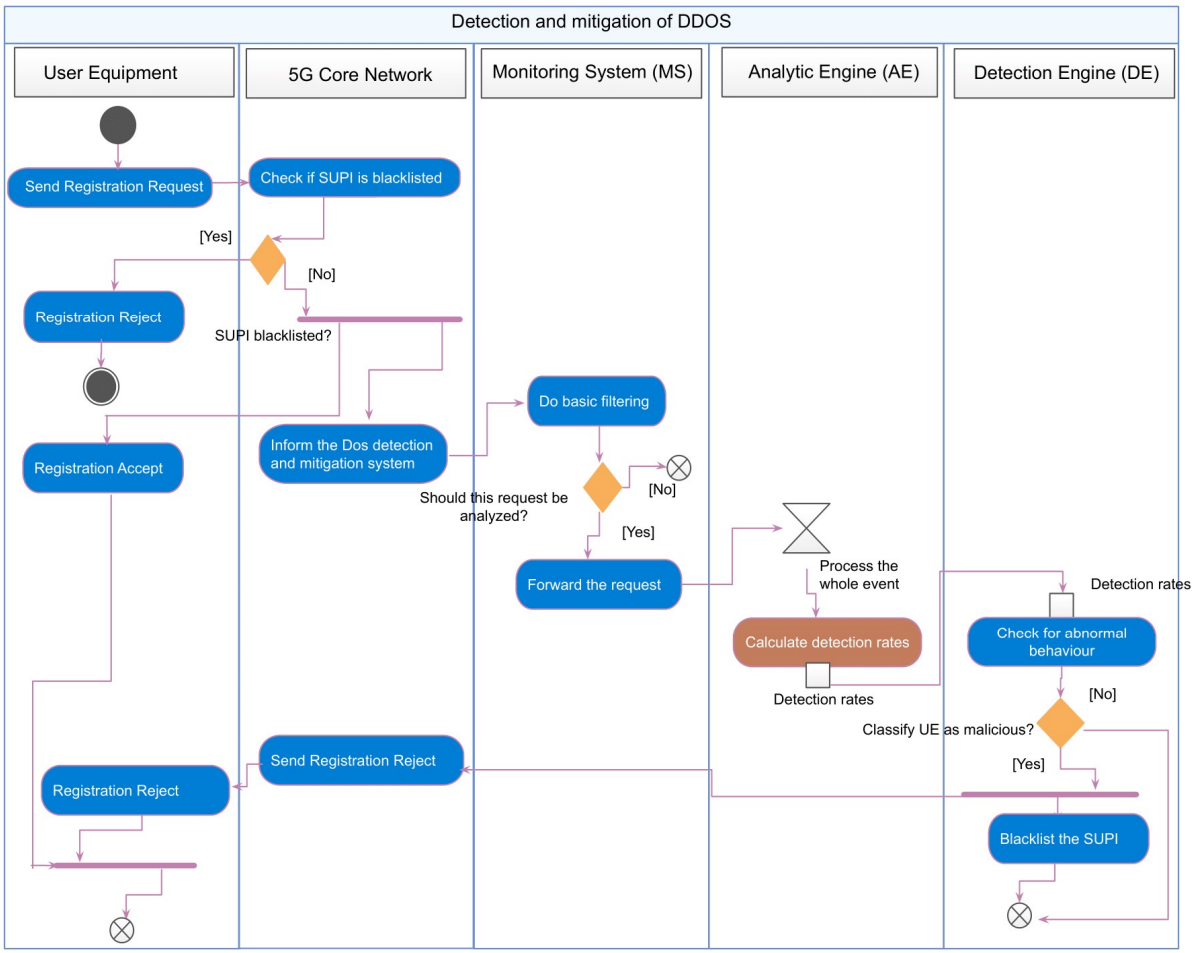

Figure 7: Interaction of the mMTC network slice and the closed-control loop to detect and mitigate attacks.

Figure 7 highlights the interaction among the different actors involved in detecting and mitigating DDoS attacks: the mMTC network slice components (UEs and 5GCN) and the closed-control loop elements (MS, AE, and DE). It is worth noting that the closed-control loop runs in parallel to the mMTC network slice elements and uses only the monitored attach requests to detect and mitigate attacks.

In the considered scenario, the MTC devices (or UEs), when detecting an event or participating in an attack, first send an Attach request to AMF. The latter must first authenticate the device and then give it access to the network resources (register the device), mainly to the data plane, to send its data. During the authentication process, the AMF checks with the Unified Data Management (UDM) if the device is blacklisted or not. To recall, the UDM is the 5G Core network function, which stores subscribers' information (Subscriber Permanent Identifier -SUPI, Quality of Service –QoS, Policy, the key k, Operator key, etc.). The device can be blacklisted if it has participated in an attack.

Meanwhile, MS, via the EM/AMF API, monitors the Attach Requests received by AMF. MS filters the data to extract the needed information, such as timestamp and SUPI. This information is communicated to the AE that processes the whole event (attack) in order to classify if the event corresponds to an attack or not. When the event finishes, the AE communicates the list of UEs involved; for each UE, a probability of being part of the attack is included. The DE then mitigates the attack by requesting the AMF to send a Registration Reject message to UEs having a high probability of being in the attack while adding the concerned UEs to a blacklist maintained by the UDM.

3.1.3 MonB5G Closed-control loop: Attacks detection

3.1.3.1 MONITORING SYSTEM (MS)

MS collects data from AMF on every Attach Request received from the MTC devices. This data is accessible through the API exposed by EM of AMF. For each Attach Request, MS extracts the device identifier SUPI and a precise timestamp. Indeed, in the 5G protocol, each UE is identified with a unique identifier called SUPI. The latter is encrypted to provide better privacy and prevent the International Mobile Subscriber Identity (IMSI) catcher attacks that were popular on previous-generation telecommunication protocols. The SUPI should not be transferred in clear text over the RAN except for routing information, e.g., Mobile Country Code (MCC) and Mobile Network Code (MNC) [27]. For this reason, it is challenging to identify devices from the traffic received on the radio layer; hence our solution has to intervene in the 5G Core Network (i.e., AMF), which can decrypt the SUPI information. The extracted information is then forwarded to AE via a communication bus based on the Publish/Subscribe concept.

3.1.3.2 ANALYTICAL ENGINE (AE)

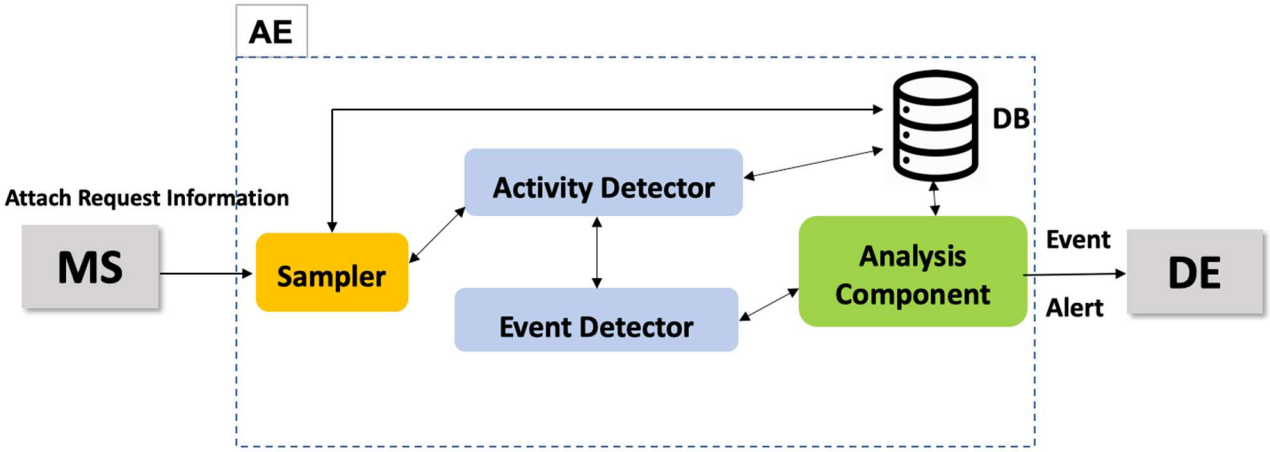

Figure 8: System architecture.

Figure 8 illustrates a detailed vision of the AE components. The latter comprises the Sampler, Activity detector, database, Event Detector, and Analysis components. They interact together to: (1) detect when an event starts and ends. It can correspond to MTC devices report (normal traffic) or attacks; (2) analyse the event to detect if it is normal or abnormal traffic; (3) compute the detection rate for each device (probability that a device has participated in the attack) and send a report to DE. We decided to separate event detection

from event analysis to improve performances and consider all the relevant data when running the overall attack detection algorithm. Indeed, we decided to detect activity periods (i.e., events) in the network traffic and only feed data to the ML algorithm at the end of an event, which provides the advantage that the resource-intensive component (detection analysis) runs once every event. We also consider two corner-cases: after a duration clearly greater than the maximum length of an event, or when peak traffic exceeds a limit indicating that it is definitely an attack, we tag the devices as malicious.

The only downside of separating event detection from the analysis is that UEs participating in an attack will not get banned right away when the attack starts. However, since the damage in DDoS attacks stems from their duration in time, the devices will get disconnected and blacklisted a few seconds or minutes after the event starts by DE. On the other hand, the performance impact is significant. The detection algorithm does not need to run in real-time, and it can look at data of the whole event.

3.1.3.1.1. Event detection

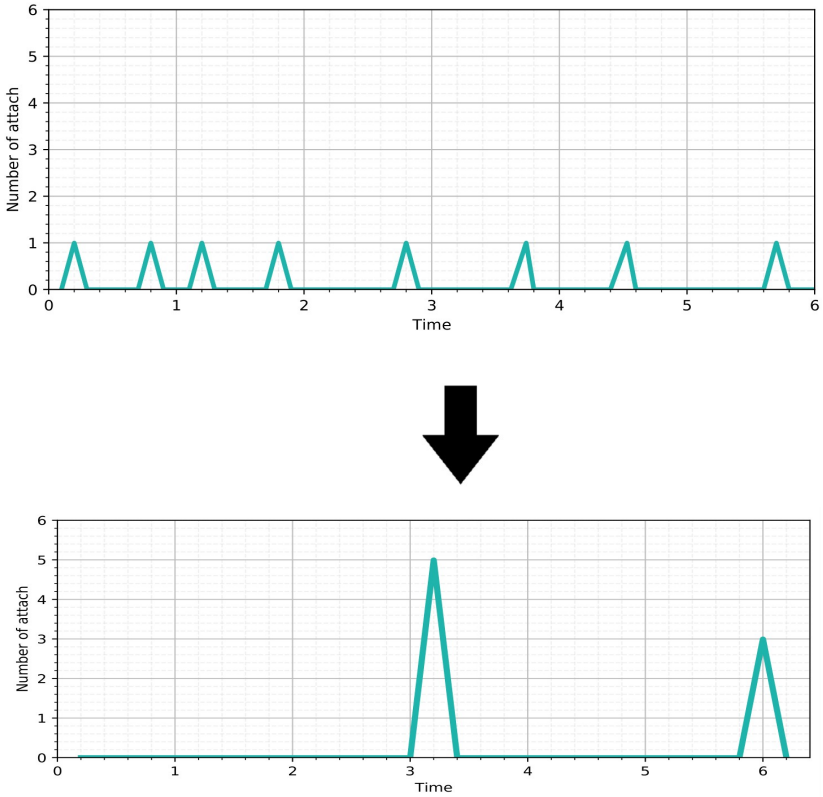

Figure 9: Sampler: attach requests on time intervals of a fixed length.

To detect activity periods, we first calculate the rate of Attach Request. To this end, we devise a new component, called the Sampler, which receives data that reaches AE from MS and emits data periodically by grouping the attach Requests into time intervals of a fixed length. Figure 9 illustrates how the Sampler works. It consists of grouping Attach Requests on time intervals of a fixed length.

The upper part of the figure shows the real Attach Request timestamp reception, while the lower part of the figure shows the output of the sampler that groups the Attach Request every 3 seconds. The Sampler outputs allow detecting the start and end of an event. If there is traffic, and we are not already in an event, we consider that one has started. If we are in at event and detect that the system has not received traffic for a given duration, we assume that the event has ended. We use a database to store all the data relevant to the event. To have a modular system, we separate the Activity Detector (the part that detects whether there is traffic or not) and the Event Detector (the part that delimits the start and end timestamps of an event).

As stated earlier, a database is used to store information about the event, namely the number of Attach Requests received for each event, a list of SUPI values that identify the devices that emitted each attach request, and timestamps. The database is accessible and used by: (1) the Sampler to store pairs **(timestamp, supi),** retrieve the largest timestamp stored, retrieve all the data stored, and delete all the data stored; (2) Activity and Event Detectors to store and edit a boolean value **is_event**.

It is worth noting that a relational database is not an optimal choice in terms of the database model. A better choice is a Time Series Database, as we will be storing time events. After many considerations, we finally opted for a key-value database. Indeed, our processing of timestamps will still be efficient, as we will be using a sorted set, and we will be able to store the boolean value needed by the Activity Detector. This avoids the need for multiple databases inside the AE.

The design of the database is highlighted in Table 2:

Table 2: The design of our key-value database.

Key	Type	Used By
devices	sorted set	Sampler
is_event	string	Activity and Event Detector
last_attach_requests	sorted set	Activity and Event Detector

3.1.3.1.2. Event analysis: ML Algorithm

The core component of AE is the Analysis Component, which receives data about one event, including the SUPI identifiers of all the devices that sent an Attach Request; then, it calculates a percentage of each device being part of an attack. The Analysis Component output (i.e., a percentage) should be zero ideally for normal traffic, as no abnormal behaviour is present. When some devices misbehave and send Attach Requests that clearly do not correspond to normal traffic, the Analysis Component output should be higher than 0. The detection rate should increase as the traffic rate increases above the normal level. We also want high detection rates (preferably 100%) for traffic likely to cause a DDoS attack.

Many algorithms can be applied to solve this problem. However, not all of them can be used as we have two important considerations:

- We already have a model for our data, the β(3,4) probability distribution, but the goal of ML algorithms is usually to estimate a function that we do not know from its inputs and sometimes its outputs (in the case of supervised learning).

- New equipment can be introduced so that the event lengths can vary. Hence, the envisioned algorithm should not detect these changes as anomalies.

Besides, we are not just trying to classify traffic as malicious or benign. While malicious traffic that triggers DDoS is easily discernible from normal traffic, it is hard to evaluate the detection rates for anomalies that are not blatant attacks (when the traffic rate is just a bit above the prediction interval bound, for example).

In what follows, we enumerated a list of algorithms that can be used for this purpose. The algorithms that can be used for this purpose are listed in the section that follows.

- Statistical tests, checking the mean, median, extremums, variance, and more, and comparing them with the properties of the $\beta(3, 4)$ distribution. Note that we will compare our solution against a statistical-based solution.
- k-NN, SVM, Isolation Forests, Decision Trees, Gradient Boosting, etc.
- Deep Learning algorithms, based on Neural Networks (Auto Encoders, Recurrent Neural Networks).

We discarded Deep Learning based algorithms as learning from data is not something we want. This would break our second consideration, and the change in the number of connected devices could trigger wrong predictions. Further, we believe that a neural network trained on normal traffic will not give a gradually higher detection probability of outliers because it does not consider outliers as the elements that are more distant from normal data. It rather works with the combination of linear and non-linear mathematical operations.

We considered the algorithms that can calculate prediction intervals (which are intervals that likely include our data). Indeed, if we can get an interval that includes our data, we can use the distance from its upper bound to calculate a prediction percentage. The most popular ML algorithm for calculating prediction intervals is Gradient Boosting. It is an extension of Boosting, where the additive generation of weak models is based on the gradient descent algorithm over an objective function [28]. The Gradient Boosting decision tree algorithm has demonstrated great performance on many machine learning tasks, including global contests [29]. It produces a prediction model in the form of an ensemble of weak prediction models, often decision or regression trees. It combines weak “learners” into a single strong learner in an iterative fashion, lowering the error estimated with the chosen loss function at each stage.

3.1.3.1.3. Event analysis - Data generation and model training

We generated data for training the ML algorithm, in the same format as the data we run our detection on, timestamps of Attach Requests in a simulated event. More specifically, we used a 5G testbed with emulated UE to emulate an event. The details of the 5G testbed are provided in section 3.1.5.1.

Algorithm 1 Data Generation

```
procedure GENERATE_EVEN_DATA (int duration, int sample duration )
    ▷ The average number of equipments that would connect during an event of length duration
    num_equipments = floor(duration×3 / 7)
    timestamps is an array of num_equipments reals
    for i ← 1 To num_equipments do
        timestamps[i] ← random_β3,4() × duration
    end for
    Sort the timestamps array
    ▷ At this point, timestamps has real values in the range [0,duration] sampled is a list of integers
    k ← 0.0
    i ← 0
    sampled_index ← 0
    while k <= duration do
        j ← 1        ▷ Count the number of samples in the range [k, k + sample_duration]
        count ← 0
        while j < num_equipments and timstamps[j] < k + sample_duration do
            count ← count + 1
            j ← j + 1
        end while
        i ← j
        sampled[sampled_index] ← (k, count)
        sampled_index ← sampled_index + 1
        k ← k + sample_duration
    end while
    Output sampled
end procedure
```

Algorithm 1 is used for generating data. The complexity of this algorithm is $O(n^2)$, where $n * n$ operations are required for inputs, and for each outer loops runs n times.

We consider that the average number of equipment that would take part in an event is: $\frac{duration*3}{7}$. We argue this by the fact that the mean of the β (a, b) distribution is $\frac{a}{(a+b)}$, where *a=3* and *b=4*. We generate random numbers following the β (3, 4) probability distribution. These numbers represent the expected timestamps of Registration requests (when sending an Attach Request).

We send Attach Requests and sleep for a duration equal to the difference between the random numbers generated to simulate the event. At the end of this step, the data is stored in a file, where each entry corresponds to a sample period. Let us note by S the sample vector defined as $\{s_1, s_2, s_3, \ldots, s_n\}$, where s_1 corresponds to the sample period 1, and noted in entry of the file as (t_1- 0) (t_1= Δ); as (s_i is (as (t_{i+1}- 0) ($t_{i+1} = i + 1 * \Delta$). The duration of the sample period is identical and obtained as follows: $\Delta = \frac{\text{Total\_Duration\_Event}}{n}$. During the training phase, we will use the previously generated data to train a Gradient Boosting algorithm to predict for each sample s_i , the upper bound of the number of attach request expected during that sample period; we note it by Predict_i . Here we are interested in the upper bound value as it corresponds to the maximum intensity of normal traffic; hence, exceeding this value means that we are most probably facing a DDoS attack. Once the training is done, we obtain a vector by P_i that contains n predicted values corresponding to the maximum expected Attach requests per sample period. The Predict_i values are also stored in the file with each corresponding entry. The file will be used later as an entry to analyse an event and calculate the detection rate.

3.1.3.1.4. Event analysis - Detection

During the detection step, the event detected by the Sampler is stored in the DB. It is organised in sample periods equal to **n** with a duration Delta (the same value used for the training phase). This will allow us to normalize the number of samples of an event since each event has a different duration period, and the β(3,4) intensity depends on the duration. Thus, we do not need to train the model using different durations, as the normalization step will allow training on a single duration corresponding to β (3,4) distribution. Knowing that the event has been organized by sample period with the total number of Attach received during the sample period as well as the SUPI of the UE, we use the ML model (mainly the file obtained in the precedent step) to extract the by Predict_i values for each sample period. Then, we derive another bound for each sample period as follows: by $\text{Predict}_i * (\text{AE\_DETECTION\_THRESHOLD} - 1)$, a limit above which any traffic gets a 100% detection rate and gets classified as malicious. For each sample period, we calculate the detection rate as follows:

For x_i , the number of Attach requests in the sample period [s_i]:

If $x_i \leq Predict_i$ *, then* $detection_i = 0$.

Else,

$$detection_i = min\ (1.0,\ \frac{x - Predict_i}{Predict_i * (AE\_DETECTION\_THRESHOLD\ -\ 1)}).$$

$$Where\ 0 < detection_i < 1.$$

Let us suppose that during an interval (Δ), the number of Attach Requests is greater than the predicted one, meaning abnormal traffic. In this context, to estimate the other bounds, from which detection values are 100%, we use Predict_i times $(\text{AE\_DETECTION\_THRESHOLD} - 1)$. This means that the predicted value is multiplied by a constant, corresponding to the rate between the distance of x from the Predict_i . This is needed to reduce the ML errors impact in order to reduce the False Positives.

3.1.4 MonB5G Closed-control loop: attacks mitigation via DE

DE is the decision-maker of the closed-control loop system. It receives data from AE and decides on the actions to take for UEs that emit abnormal traffic. DE receives as input a list including the suspected UE (SUPI) and their corresponding detection rates (i.e., detection_i) values belonging to the attacks. We devise two versions of DE. The first solution blacklist devices if their calculated prediction is higher than $\text{DE\_DETECTION\_THRESHOLD}$, a configurable parameter. The lower the threshold value, the higher the probability that devices will be blacklisted. Therefore, the network operator would use low values in order to be more strict, but it may increase the false positive in this case. To avoid having high false-positive results, we introduce a second solution that relies on three thresholds. The second solution considers the whole event and classifies the received list of UEs into three categories: F_1 , $F_2, and\ F_3$. DE calculates how many UEs have obtained a detection rate (detection_i) higher than 0.8 and assigns it to the first category, namely F_1. The second category includes UE having a detection rate between 0.3 and 0.8. This category corresponds to F_2. The remaining UEs are obviously included in F_3. Then, DE checks if, in the event, a significant part of the devices had higher than usual detection rates. As a result, different decisions have to be taken:

- First, if $F_1 \geq F_2\ and\ F_1 \geq F_3$, DE blacklists all the devices by adding their SUPI values to a table of blacklisted values, and disconnect them from the network.
- Second, if $F_2 \geq F_1\ and\ F_2 \geq F_3$ **,** DE adds the SUPI values to a table named ``non-trusted devices". Each UE belonging to this table has a counter, named T_{imsi}. The counter is increased by 1 each time the UE is involved in abnormal traffic that is not blatantly an attack. The counter is incremented until it reaching a value that yields to blacklist the device.
- Third, DE ignores the alert sent by AE, and it will do nothing.

The justification for using three categories (F_1 , F_2, F_3) is based on the AE analysis and the results accuracy of the employed ML algorithm. Indeed, we notice that if the detection values associated with the connected devices sent by the AE present high values, this indicates that the traffic generated by the devices does not follow the Beta distribution. Hence, they should be immediately blacklisted as they present abnormal behaviour, justifying the need for the first category. The latter is considered a red alert, and the associated devices SUPI are blacklisted. However, when the values are neither too high nor too low, meaning that the traffic is almost close to the Beta distribution. In this case, the detected devices could be either malicious behaviour or caused by technical failure. Therefore, we introduced the second category, which means that the devices are not blacklisted, but the DE will memorize the associated SUPI for future events. If the devices are classified in the second category more than 2 times, they will be considered malicious and moved to the first category; they will be blacklisted. The last category corresponds to the detected traffic being very close to the Beta distribution. This case could be either an error in ML calculation or a delay in sending the Attach Request.

3.1.5 Results

3.1.5.1 THE TEST PLATFORM

To validate the proposed zero-touch security management system, we have used a 5G testbed deployed at EURECOM[1]. The testbed has been developed and used in many 5G European projects such as 5GEve[2] and 5G!Drones[3]. We have implemented the closed-loop components (i.e., MS, AE, and DE) and an EM on top of the AMF. The latter exposes the API to (1) MS to monitor the Attach Request message; (2) DE to detach and blacklist UE involved in an attack. Figure 10 illustrates the different technologies used to implement the above-mentioned components. As a quick reminder, the roles of the different components are:

- MS and sampler: MS is the first component to receive traffic from the 5G Core Network. It performs basic filtering on it. While sampling the input data, the sampler gathers information on Attach Requests as they are received (with no guaranteed periodicity) and emits periodic data containing the number of Attach Requests received in time intervals of a given length.
- Activity and Event Detectors: These components receive the sampled data and should detect an event. For each event, this component only emits data at its end, with the number of requests on each time-slice and all UEs that emitted traffic during the event.
- Analysis Component: This component runs the ML algorithm on the given data, calculating a detection rate for each time-slice (for all devices in the time-slice).
- DE: This component receives data from the Analysis Component and decides which devices should be disconnected from the network.
- MQTT Broker: is used to implement the communication bus between the different components of the closed-loop control system, and between the closed-loop control system and the AMF.

Regarding the UE, we used and updated a 5G UE emulator, my5G-RANTester, to be able to send a high number of UE Attach Request messages in parallel to simulate an attack or normal traffic. Indeed, my5G-RANTester is a tool for emulating control and data planes of the UE and gNB. my5G-RANTester[4] follows the 3GPP Release 15 standard for NG-RAN (Next Generation-Radio Access Network). By using my5G-RANTester, it is possible to generate different workloads and test several functionalities of the 5G core, including its compliance with the 3GPP standards. Scalability is also a relevant feature of the my5GRANTester, which is able to mimic the behaviour of a large number of UEs and gNBs accessing simultaneously the 5G core. Currently, the wireless channel is not implemented in the tool. The AMF and 5G CN components are based on OpenAirInterface (OAI).

[1] https://www.youtube.com/watch?v=90SRV9ZpPVo&t

[2] https://www.5g-eve.eu/

[3] https://5gdrones.eu/

[4] https://github.com/my5G/my5G-RANTester

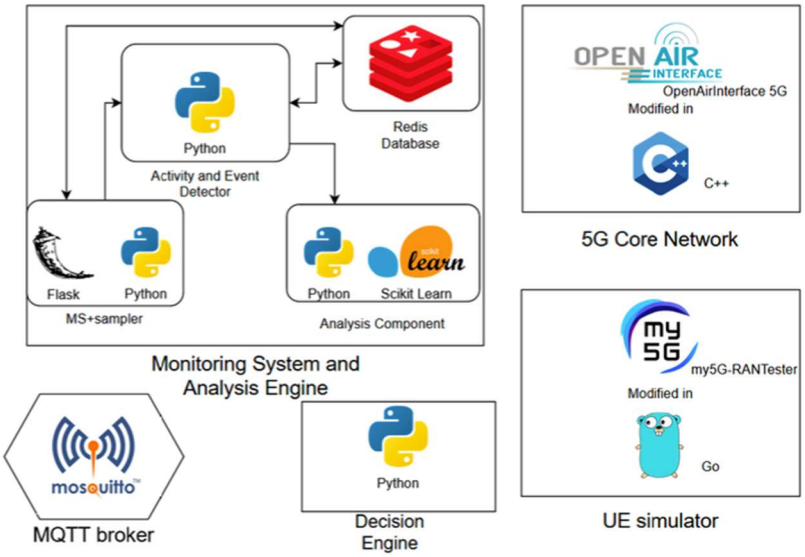

Figure 10: Test platform and technological components.

As described earlier, AE and DE use several parameters that need to be tuned to optimize the different steps of the event analysis. These parameters are summarized in Table 3 and defined as follows:

Table 3 : The configurable parameters in our system.

Parameter Name	Default Value	Component
INTERVAL_LENGTH	**6.0**	Sampler
UNCHANGED_INTERVALS_COUNT	**4**	Sampler
REQUEST_THRESHOLD	**1**	Detector
MAX_EVENT_DURATION	**600**	Detector
DE_DETECTION_THRESHOLD	**0.35**	DE

- INTERVAL_LENGTH or Delta (Sec): Length of the sampling interval. A request is sent from the sampler every INTERVAL_LENGTH seconds, with the number of UEs that connected in the last interval of time of this length.
- UNCHANGED_INTERVALS_COUNT: Number of intervals without activity after which to mark an event as finished. This parameter is used to detect the end of an event. For instance, if its value is equal to 4, and INTERVAL_LENGTH is 6, this means that, if there is very little to no activity for 24 seconds (or four intervals), and an event was ongoing, we consider it has ended.
- REQUEST_THRESHOLD: If there are less requests than this value in an interval of time, consider that there is no activity.

- MAX_EVENT_DURATION (Sec): The maximum duration for an event, if the end of the event is not detected after this duration, we consider it is an attack, and we ban every device taking part in the event.
- DE_DETECTION_THRESHOLD: a threshold used by DE to determine if a device should be banned from the network or not, any detection rate higher than this value leads the system to ban the device.

Regarding the testing conditions, as we do not have datasets that include both positives and negatives, we generate traffic corresponding to the non-malicious using the β(3, 4) probability distribution, while for abnormal traffic, any other distribution can be used. Besides, we are considering timestamps as our only feature, as hence we cannot differentiate between two Attach Requests received at a moment when traffic is dense.

To test our system on realistic circumstances, we leveraged the my5G-RANTester with a script that allows emulating real UE traffic (control plane) to communicate with 5G Core Network components. We used the emulator to simulate an event with different chosen SUPI values. The generated traffic is similar to what real UEs would generate. It follows the 3GPP specifications 15. Algorithm 2 generates the traffic featuring:

- Simulating an event: Sending attach requests that follow the Beta-distribution
- Simulating an attack: Sending attach requests that follow a uniform distribution
- Simulating the registration of a single UE

Algorithm 2 Event simulation

```
procedure SIMULATE_EVENT(int duration)
    num_equipments = floor((duration×3)/7)
    timestamps is an array of num_equipments reals
    for i ← 1 To num_equipments do
            timestamps[i] ← random_β3,4() × duration
    end for
    currSUPI ← MCC + MNC + "0000000001"
    Sleep(timestamps[0])
    for i ← 1 To num_equipments − 1 do
    SendAttachRequest_ASYNC(currSUPI)
    currSUPI ← str(int(currSUPI) + 1)
    Sleep(timestamps[i])
    end for
    SendAttachRequest_ASYNC(currSUPI)
end procedure
```

It is used for simulating an event, it's very similar to the algorithm that generates data. The complexity of this algorithm is $O(n^2)$, where **n** operations are required for input, and the outer loop runs **n** times. Here, n corresponds to the number of devices involved in an event. Figure 11 and Figure 12 show a visualization of traffic likely corresponding to a normal (four events and attack) and malicious one, respectively.

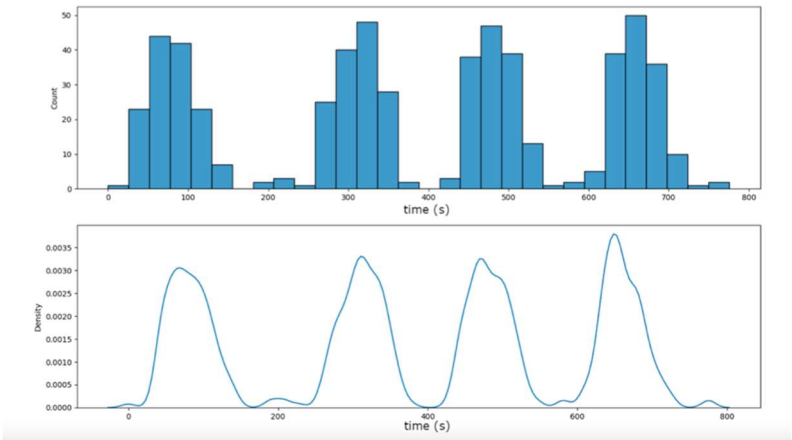

Figure 11: Normal traffic - four events.

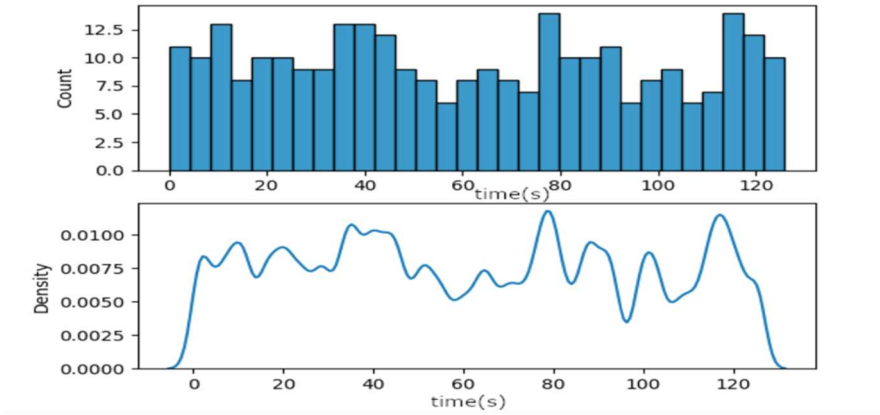

Figure 12: Malicious traffic.

Last but not least, readers may see a video[1] showing a demonstration of all the components, i.e., closed-control loop as well as AMF and UEs, working together to detect DDoS attacks. The following section provides some results on the model accuracy obtained via the experiments.

3.1.5.2 TEST RESULTS

To evaluate the performance of the proposed framework, we focus mainly on the performance of the attack detection algorithm, which is the key function of the closed-control loop. To this end, we evaluated the Gradient Boosting algorithm to detect DDoS attacks accurately and compared its performance with a statistical method. Like the Gradient Boosting solution, the statistical method is applied at the end of the event. By using the duration of the event, the statistical method defines a limit function using the mathematical function of β(3,4) to compare the different points in the report to this limit; all the points

[1] https://www.youtube.com/watch?v=QzCmGfwtDLAt=7s

exceeding this limit are considered as a possible attack. The main differences compared to Gradient Boosting are: (i) it needs the exact duration of the event; (ii) it uses the β(3,4) function to deduce the limit.

3.1.5.2.1. AE performances

Gradient Boost

To measure the accuracy of the Gradient Boosting algorithm, we evaluated the accuracy in normal and malicious traffic samples. In normal traffic, it denotes how often the system yields a detection rate of UEs that is greater than zero. This does not imply that these devices will be banned, but ideally, a value of 0 should be returned for all the devices emitting normal traffic. To evaluate our model on normal traffic (True Positive (FP) = False Negative (FN) = 0), we generate data for 500 normal events and run our detection algorithm on each of them. We then count how many UEs, we obtained a greater-than-zero detection rate versus the total number of UEs in all the events. We generate event durations randomly (between 30 and 250 seconds. Regarding malicious traffic (True Negative (TN) = False Positive (FP) = 0), we also ran 500 tests, but this time, between 7 and 15 Attach Requests were received every 6 seconds, for a total duration that is random between 30 and 250 seconds.

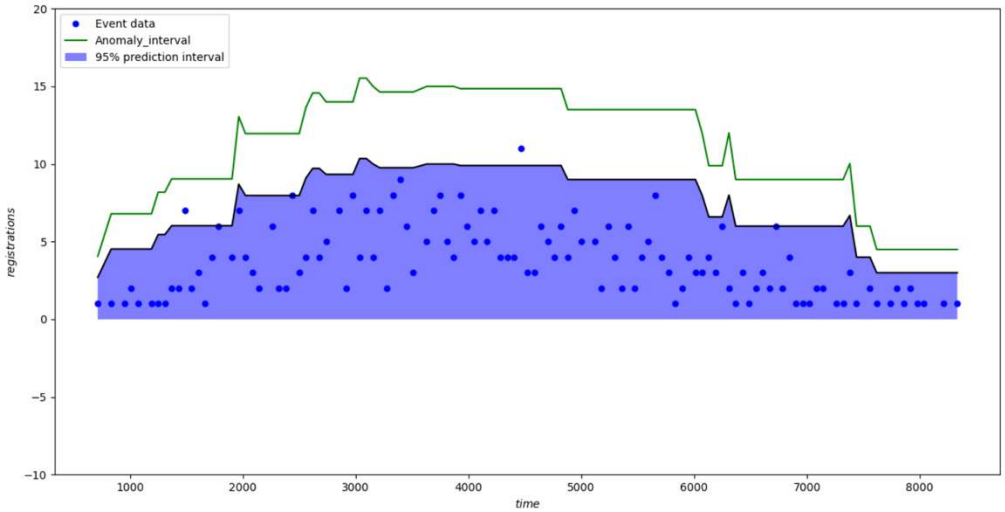

Figure 13 : The result of the detection algorithm over normal traffic.

Figure 13 allows visualizing the results for normal traffic. The points correspond to the event data, while the green line is the anomaly interval. If a point is outside the limit (in green), it will be assumed as an attack. For normal traffic, the accuracy is computed as 1-(FP/(TN+FP)). Hence, the results show an Accuracy on normal traffic of 96.76859478052322%. We expected this result as the interval used in our training data includes around 95% of the data in the training dataset, as depicted in Figure 13.

Figure 14 illustrates the results of malicious attacks. For this case, the accuracy is computed as (1-(FN/(FN+TP)). Hence, the results show an accuracy of 83.63319140762557%. This represents an excellent result as banning a relevant part of the devices taking part in a DDoS attack is enough to mitigate it. Indeed, this type of attack requires a high detection rate (detection_i) of requests to overload the target system with

traffic. Also, this is just the detection rate calculated by the AE component, the final decision on which devices should be banned is taken by the DE component, which can be configured as needed.

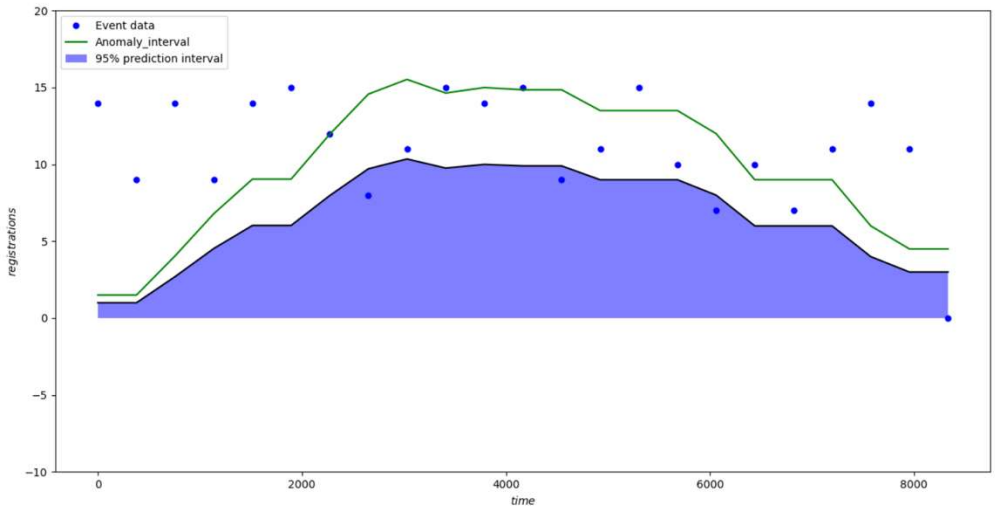

Figure 14: The result of the detection algorithm over abnormal traffic.

Table 4 shows the performance of the Gradient Boosting-based solution when modifying the AE_DETECTION_THRESHOLD value. It is worth recalling that this value is used to derive the detection rate and corresponds to a protecting gap to reduce the impact of the ML prediction and hence reduce the FP value. We remark that the value that allows reducing both FP and FN is 2.0. When this value increases, FP is reduced by the FN increases, whereas when it is reduced, both FP and FN increase.

Table 4 : Impact of the AE detection threshold on the Gradient Boosting accuracy.

	AE_DETECTION_THRESHOLD		
	1.2	2.0	3.0
Normal event	82.98654	96.76859	97.64853
Abnormal event	81.91305	83.6331914	61.86956

Statistical Method

For the sake of comparison, we used the same scenarios as for Gradient Boosting to generate normal and abnormal traffic. Then, we applied the statistical method and verified its accuracy in detecting attacks. Figure 15 illustrates the usage of the statistical method in case of an abnormal event. The discontinued green line shows the β(3,4) curve obtained according to the event duration. The β(3,4) curve allows us to have a limited

path from which all the outside points are considered anomalies and hence potential attacks. The statistical method's results show that 36.0% of the Attach requests do not follow the β(3,4) distribution (they are out of the limited path), which can then be considered as potential attacks.

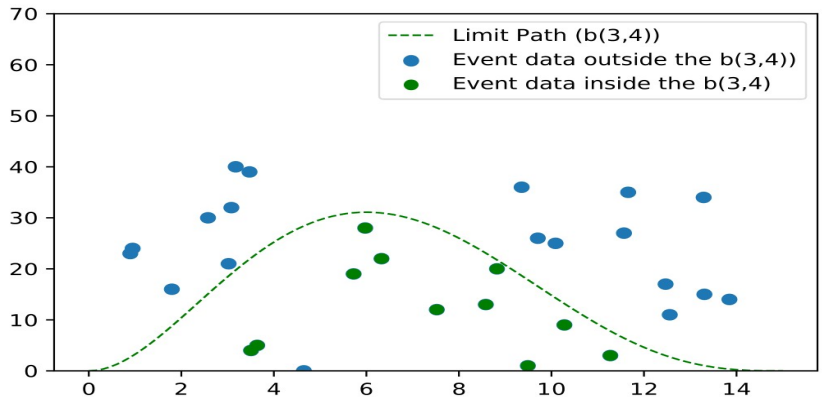

Figure 15: The result of the statics method over abnormal traffic.

On the other hand, Figure 16 presents a test of a normal event. The static results show that 90.0% of the Attach requests follow the β(3,4) distribution. The accuracy of the statistical method for detecting anomaly detection are: 1-(FP/(TN+FP)) = 84.21052 % (normal traffic), 1-((FN/(FN+TP)) = 57.142857 % (abnormal traffic))). We remark that these values are weaker than the Gradient Boosting algorithm. We argue these differences by the fact that the duration estimation has a strong impact on the statistical solution.

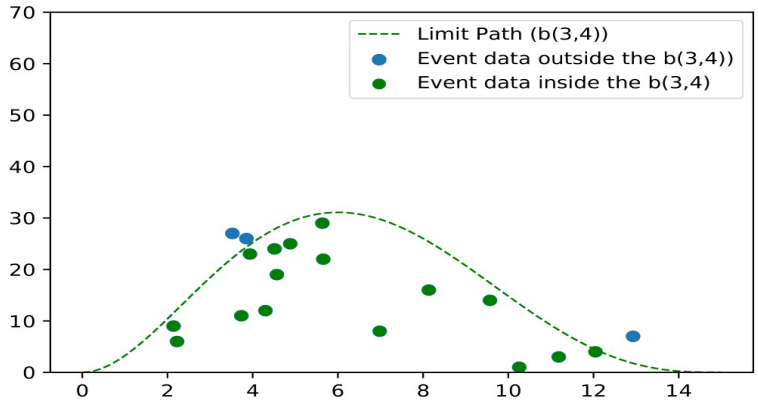

Figure 16: The result of the statics method over normal traffic.

The shape of the β(3,4) curve changes in a drastic way according to the duration (noted D), which impacts the accuracy seriously. For instance, we change the duration by +/- ϵ= 2 sec, and the obtained results are summarized in Table 5. We see clearly from this table the impact of the duration on the accuracy as a small error on the duration drastically yields a drop in the accuracy. In particular, if the duration is less than the real one, many points will be out of the curve. In the Gradient Boosting algorithm, we do not have this concern, as the latter normalizes the sample period duration and uses the trained model to detect the limit.

Table 5: The accuracy of the statistical model considering different Duration values.

	GB	Static method		
		D - Δt	D	D + Δt
Normal traffic	96.76859	45.18924	84.21052	80.24568
Abnormal traffic	83.633191	30.4156	57.142857	51.86854

3.1.5.2.2. DE performance

Regarding DE performances, we evaluated the first version that relies on a single threshold DE_DETECTION_THRESHOLD. Accordingly, in this section, we evaluate the DE_DETECTION_THRESHOLD impact on the number of blocked devices. Table 6 shows the number of banned UEs for three values of the DETECTION_THRESHOLD. As expected, we remark that lower threshold values (ex. 0.1) are very conservative, allowing to block more UE while a higher threshold value (ex. 0.8) may be less strict and reduce the number of banned UE. We recommend two different approaches addressing the DE_DETECTION_THRESHOLD issue. The first one considers the performance limit of the element to protect against DDoS attacks. In our case, we computed the response time of the AMF to attach requests while increasing their number. After a certain number of Attach requests, we noticed that the AMF started to be very slow, which be caused by a DDoS attack. Therefore, after some tests, we found that the value of DE_DETECTION_THRESHOLD equal to 0.35 allows avoiding reaching the Attach request number that yields bad AMF performance. Another solution is to use a dynamic threshold value that decreases or increases over time according to the number of consecutive events classified as an attack.

Table 6: Impact of the DE Detection threshold.

	Detection Threshold DE		
	0.1	0.35	0.8
Nb (Normal event)	3	1	0
Nb (Abnormal event)	21	16	10

3.1.6 Conclusions

In the mMTC scenario, we introduced a zero-touch security management framework that aims to protect mMTC network slices from in-slice DDoS attacks. The proposed framework relies on a closed-control loop that tracks Attach Requests generated by MTC devices to detect abnormal traffic and mitigate possible DDoS attacks. The mitigation process is enforced through disconnecting and banning suspected devices. The attack

detection algorithm uses the ML technique, specifically Gradient Boost, to create a prediction interval that is likely to include normal traffic in our system. It then calculates, for every sample that is outside the interval, a metric that depends on its distance from the bound of the interval; this metric is called the detection rate. The DE uses this metric to take actions, banning and disconnecting devices from the network to prevent them from conducting similar attacks in the future. The proposed framework has been implemented using a 5G testbed. The obtained results indicate the ability of the closed-control loop to predict and mitigate DDoS attacks efficiently.

3.2 Traffic steering and Security VNF instantiation studied in T5.1

3.2.1 INTRODUCTION

It may be decided to include security features in the network slice during the preparation phase in order to ensure that the achieved protection is in line with the security levels expected by the customer. These protection features should be compliant with the baseline measures recommended by 3GPP TS 33.501 [30] and 3GGP TS 33.117 [31] for 5G system. Moreover, particular use of the network can increase the risk of a cyber-attack and requires a more stringent protection strategy. However, any security measure inevitably has an enforcement cost, and the network operator may decide not to implement them all against the acceptance of some residual risks. Thus, ANSSI Cyber for ICS [32] recommends adopting a defence-in-depth strategy to protect against undiscovered threats, reduce the attack surface, and minimise the impact of the incident. This strategy is about defending systems by surrounding them with successive autonomous barriers including preventive protection, surveillance, detection, and response activities. If preventive barriers are not sufficient to block an attack and the attack has been spotted, the incident response activities can reinforce the measures in place by updating the configuration of VSFs and VNF instances, or by deploying additional control functions, thanks to the API for lifecycle management exposed by the domain orchestrator.

In cloud-native architecture, technologies such as containers, service meshes, microservices, immutable infrastructure and declarative APIs provide an alternative approach to physical or virtualised machines for building and running scalable applications. Instead of using layer 2/3 network links to create chains of functions running on host machines, native cloud services interact with each other across a shared fabric built on the use of Layer 7 communication protocols. In the following, we will illustrate with a use case the mitigation of a threat to 5G communications by updating the network service and changing the network configuration of UEs. These mitigations include API calls, on one hand, to the cloud-native infrastructure orchestrator to insert a new security feature into the data plane, and on the other hand to the core network to update security policies and network settings for UEs.

3.2.2 STATE OF THE ART ON EXISTING ATTACK AND SOLUTIONS

3.2.2.1 VNF INSTANTIATION

VNF Instantiation is the process of deploying an instance of a VNF through an NFV platform, and having it ready to handle traffic. The instantiation process includes two steps: provisioning the needed virtual resources, and configuring the templates bundled in the same VNF.

VNFs may have inherent software or system vulnerabilities, or malware designed to perform attacks, therefore they could either be the source or the target of a threat or attack. Attacks from within VNF are

possible due to software vulnerability flaws, where a malicious attacker could exploit a flaw (using snort[1] for example) to bypass firewall restrictions or to take advantage of a buffer overflow to execute a malicious code.

Furthermore, VNFs have virtualization vulnerabilities, according to ENISA's threat Landscape [33] :

1. **Network Virtualisation Bypassing**
   Improper configuration along with bad network slicing implementation can result in loss of data which can harm both confidentiality and privacy. As the networks are shared between different users, it should be assured that only legitimate traffic enters or leaves a network slice. Unless traffic is isolated, slice trespassing will happen.
2. **Abuse on data Centers Interconnect (DCI) protocol**
   Whatever system relies on virtualisation is usually deployed within data centers, thus inheriting their threats and vulnerabilities. This threat refers to the lack of authentication and encryption. Under those circumstances, potential attackers can create spoofed traffic in a way that makes it possible to traverse DCI links or creating a DoS attack on DCI connections.
3. **Virtualised Host Abuse**
   In a virtualised environment, physical resources are shared between the different applications and their users running on virtualised host. Thus, there is a vulnerability that some of those users want to overcome the boundaries and the limitations of their virtualised space, invading the neighbouring spaces, scavenging information. This threat permits cross-inspection of various tenant's data flow, neighbouring attacks which allow the mapping of topology, thus serving as the initial step for DoS attack.
4. **Abuse of Cloud Computational Resources**
   In NFV networks, as the compute nodes are outside of the core, it requires the operators to loosen the security rules between the controller and the compute nodes. This slackening of the security can open up the whole environment to a malicious attacker and thus compromise it, leading to data loss, breaches, and loss of service.

Additionally, ETSI's GS [34] suggests that VNFs' lack of management authentication is a prime vulnerability, and that should be addressed. The report includes the specifications and the requirements for the aforementioned problem. Management authentication is essential for any real-world deployment and affects every phase of the VNF lifecycle. The existence of an authentication API, that can allow only authorized users to get access and manage to operate the VNF, will mitigate this vulnerability.

As it is evident that most VNF-related vulnerabilities lead to DoS disruption, we refer to a detailed survey [35] of SDN-based DDoS attack detection and mitigation solutions that aggregated DDoS (flooding) attacks into three categories by the authors:

i. Reflection-based Attacks

1. https://www.snort.org/

These are reflection-based volumetric attacks where the attacker overwhelms the target network by injecting a large number of ICMP packets. Two infamous examples of this type of attack are the Smurf[1] and Fraggle[2] attacks.

ii. Protocol Exploitation Attacks

Where attackers can exploit the synchronize (SYN) protocol and send large UDP packets to consume more bandwidth. Examples are the SYN flooding and UDP fragmentation attacks.

iii. Amplification-based Attacks

Where attackers are able to generate a large volume of DNS and Network Time Protocol (NTP) requests to jam their servers, rendering the target and surrounding infrastructure inaccessible to regular traffic.

3.2.2.2 SOLUTIONS AND MITIGATIONS

A recent report [36] describes a novel mitigation experiment that was performed under the **EU H2020 SoftFIRE project**[3]. The solution designed is called **BotsOnFire**, and within its report the authors specify the implementation of two VNFs and an SDN application, and their testing as SDN/NFV applications.

Their 3-tier solution is described as follows:

I. Botnet detection

A Snort Deep Packet Inspection (DPI) VNF is deployed for detecting botnets at a low level. Snort is one of the most widely used Network Intrusion Detection Systems (NIDS) for misuse detection, which is 5G compliant due to its ability to analyze GPRS Tunneling Protocol (GTP) packets natively without having to de-capsulate them first.

II. Botnet mitigation

A Honeynet VNF is created to emulate the malicious behaviors of compromised UE. The Honeynet is a security component chosen to isolate bots in the compromised UEs by emulating their behavior, thus preventing real bots from executing attacks. This mechanism works because honeynets can be used to continuously analyze how botnets evolve over time. The results of the analysis can then be used to advise monitoring and detection modules on how to adapt their internal processes to the observed changes.

III. Enabling detection and mitigation procedures in SDN

An SDN application (called FlowT) has also been implemented to enable security functions of the Snort and Honeynet VNFs. Effectively mirroring suspicious flows to the Snort VNF for inspection and diverting network flows exchanged between bots and the server for botnet emulation by the Honeynet VNF. FlowT enables

[1] https://www.sciencedirect.com/topics/computer-science/smurf-attack

[2] https://www.radware.com/security/ddos-knowledge-center/ddospedia/fraggle-attack/

[3] https://www.softfire.eu/

both features by selectively applying network flow mirroring and diversion rules to the virtual switches (vSwitches) being managed by the SDN Controller.

3.2.3 MITIGATION USING MONB5G AI-DRIVEN SECURITY TECHNIQUES

In the previous section, we discussed the need to adopt the strategy of defence in depth by erecting successive concentric barriers around the system to be protected. The rationale for such a strategy is related to the risks that the protection measures in place contain vulnerabilities such as software or hardware defects or are ineffective against unknown attacks or are implemented with tolerance of residual risks. Preventive measures alone are not sufficient in this strategy, which leads to the inclusion of other mechanisms such as surveillance, intrusion detection, and incident response activities.

To prevent aLTEr attacks (see deliverable D5.1 for details of this attack), the PDU session integrity protection exists in 5G and it detects any attempt by the MitM to redirect DNS request messages. However, the over-consumption of the battery due to the calculation of the integrity code and the low likelihood of the attack being implemented because of its complexity do not motivate the network operator to deploy such a measure. The risk can then be accepted as it is mitigated by the mechanisms to detect and remedy such an attack provided for in the defence in-depth strategy. In this context, we have employed the distributed and autonomic architecture of the security orchestrator proposed in D2.4 [37] as defensive barriers. These barriers cover actions of different scopes, from the local vision (intra-slice) managed by the L-ASO to be more reactive and precise, to the global vision (inter-slice) elaborated by the D-ASO from consolidated information to better coordinate activities.

If the PDU session integrity protection option is not enabled, any modification of user data by the MitM cannot be detected at the gNB, Therefore the aLTEr attack will succeed in redirecting DNS requests to the malicious DNS server. To counter this, we have proposed the defensive line implemented by the L-ASO using Monb5G sublayer components MS, AE, DE:

- To detect a potential aLTEr attack, DNS traffic is continuously collected, prepared and analysed to detect anomalies. Indeed, the attacker's modus operandi consists in swapping the address of a known DNS server with his own by XOR operations while adapting the TTL value to preserve the checksum of the IP packet. The presence of these packets containing these fields with unusual values leads to the hypothesis of an aLTEr attack. As part of the implementation, the MS performs the network security monitoring that unobtrusively collects raw network traffic from the user traffic. We observed this at the N6 interface. User traffic is copied and filtered to extract only DNS, then sent over a secure communication link to the Zeek monitoring tool, which translates it into high-level transaction logs to ease the detection function.

The detection process consists of three main steps: the anomaly detection from the DNS logs using AI/ML techniques, then the creation of the aLTEr attack hypothesis, and finally the resolution of the hypothesis based on inference machine.

Anomaly detection, sometimes referred to as outlier detection, is the identification of rare items, events or observations that raise suspicions by differing significantly from the majority of the data. Typically, anomalous items will translate into certain problems such as bank fraud, medical problems, or cyber security incidents. Anomalous items might be referred in the literature as outliers, novelties, noise, deviations and exceptions. In our use-case, we try to detect anomalies in DNS log records that are generated by Zeek. We

assume that we are unaware of the attack pattern and characteristics of the current attack, and the common approach is to use an unsupervised or semi-supervised method for detecting outliers. In the literature, there are three popular algorithms that are applied in these kinds of situations, namely One-Class SVM (OcSVM), Isolation Forest (IForest) and Local Outlier Factor. In the scope of this project, we were able to try the OcSVM and IForest methods. From the plot of the DNS server IP address distribution, we can deduce the hypothesis that anomalies are more likely to occur in areas where we have low density. We tried to plot the anomaly score of the iForest fitted model along with the outlier regions and we obtained the results shown in Figure 17. iForest results validated our hypothesis, although there are many normal points that were decided as anomalous and vice-versa.

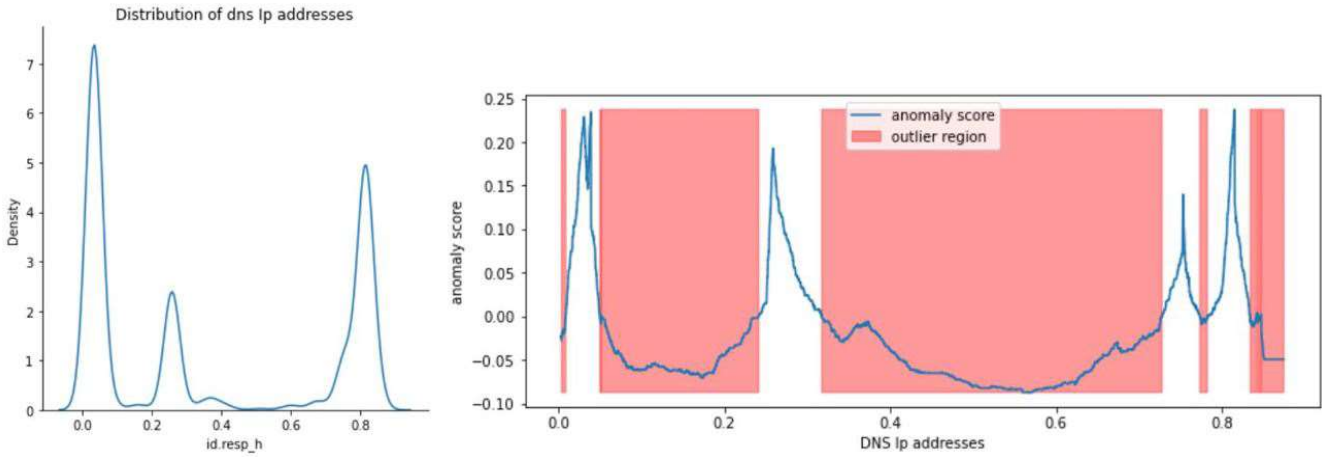

Figure 17: Anomaly score and outlier regions output of the iForest model.

We trained a One-Class SVM model on the DNS logs dataset, more precisely on the destination IP address feature, supposing that it doesn't contain outliers, meaning that the values taken by this feature are all legit. We tried many combinations of hyper parameters until we found a suitable one. Figure 18 represents a plot of the decision boundaries. As we can see in the plot, many regions that are susceptible to contain anomalies were not detected. We can deduce that the DNS IP address distribution makes it harder for the OcSVM model to find the right outlier regions.

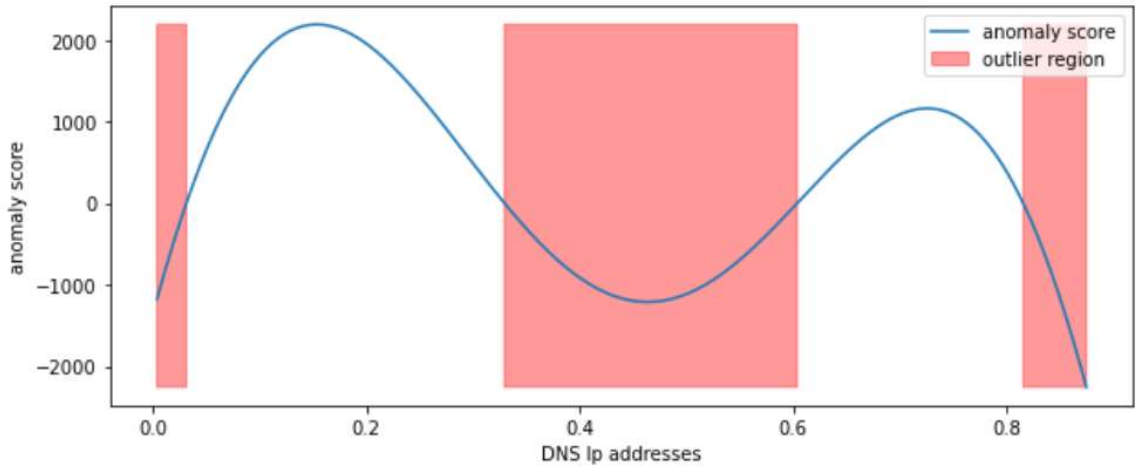

Figure 18: OcSVM anomaly score and outlier regions.

From the evaluation in Table 7 we can deduce that the iForest model gives more precise decisions, meaning that most of the data samples that were detected as outliers were actually real ones. On the other hand, the OcSVM model gives a higher value of the recall metric, and it rarely makes a decision about a sample being an outlier.

Table 7: Evaluation of Isolation Forest and One-Class SVM models applied to the DNS logs dataset.

Model	*Precision*	*Recall*	F_1 *score*
iForest	0.96	0.48	0.64
OcSVM	0.92	0.74	0.82

The analysis of alerts is a difficult task because, as our case shows, ML models can produce false positives and in large proportion. Even when an alert is a real positive, it does not necessarily indicate that an incident has occurred. For example, a DNS request with an unusual server address does not necessarily mean that the address has been modified, as the user may simply have configured a private DNS on his terminal in order to be able to sort out the case of an attack from a legitimate one. Therefore we hypothesise an aLTEr attack as the most likely scenario known that could explain the indicators. We then use policy engines such as ONAP APEX which have been previously provisioned with inference rules to solve the aLTEr hypothesis. The prerequisite for the attacker is to know the address of the target DNS server so that he can replace it with his own by a simple XOR operation. The policy engine issues then actions to update the DNS configuration on the UE with a temporary random DNS server address and waits to receive a new DNS request from the UE to check the following four subcases. These subcases are combinations of two Boolean variables: whether the aLTEr attack is taking place or not, and whether the UE is configured with a network operator or a private DNS server.

1. aLTEr attack is not taking place, UE is configured with a network operator DNS server
    - The DNS query is normal, security alert should not be raised
2. aLTEr attack is not taking place, UE is configured with a private DNS server

- The ML/AE will detect the anomaly, the alert analysis will clear the status of the private DNS by challenging the UE with a random DNS address configuration. As a response, the action plan triggered by the DE (policy/inference engine) will consist of:
    - the PDU session modification procedure defined in 3GPP TS 23.502 [38] and
    - security policy update on the firewall to allow the private DNS address
    - security monitoring update to no longer send this traffic to the detection function

3. aLTEr attack is taking place, UE is configured with a network operator DNS server
    - The ML/AE will detect the anomaly, the alert analysis will clear the status of the private DNS by challenging the UE with a random DNS address configuration. This outcome of the challenge will confirm the aLTEr attack. To contain the attack, the action plan triggered by the DE (policy/inference engine) will consist of:
        - security policy update on the firewall to deny the private DNS address from the UE
        - reporting the incident as an aLTEr attack is on going
        - update the User Plane security policies for a PDU session in UDM to require PDU Session UP integrity protection
4. aLTEr attack is taking place, UE is configured with a private DNS server
    - The attacker is manipulating the address without knowing that is a private address. As a response, the action plan triggered by the DE (policy/inference engine) will consist of
        - reporting the incident as an aLTEr attack is on going
        - update the User Plane security policies for a PDU session in UDM to require PDU Session UP integrity protection

The DE is based on a rule engine such as the projects ONAP APEX and DROOLS, where the rules have been elaborated initially by the experts to decide on the actions to be taken for:

- the analysis of the detection alert: making the assumption of the most plausible scenario that can correlate events, and resolve it; if the attack is confirmed, the alert becomes a cyber-security incident;
- to respond to the incident: to limit the impact of the attack, eradicate the threat and use the lessons learned to improve the preventive measures.

The DE based on rule engine is stateful, it follows up contexts when handling events as shown in Figure 19.

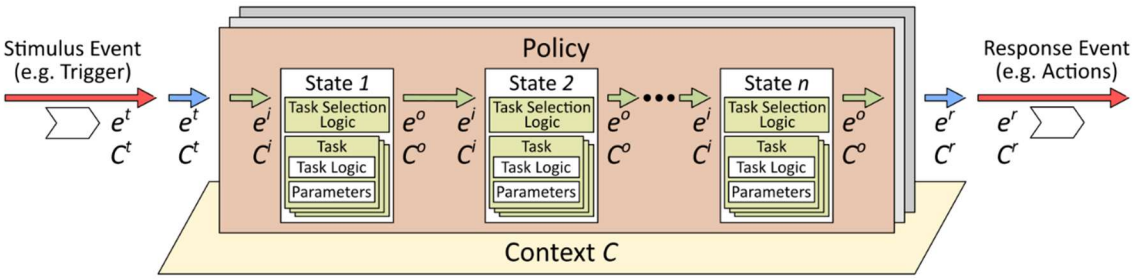

Figure 19. States and contexts in the policy engine APEX.

Before measuring the end-to-end KPIs such as MTTD (Mean Time to Detection, which is the average time it takes for an issue to be identified as one requiring attention) and MTTR (Mean Time to Respond, which is the average time it takes to begin the work associated with a service ticket), we have assessed the performance of the rule engine APEX by performing some load tests. The results in Figure 20 and Figure 21 of the testing show that the number of rules and the frequency of queries affect response times from about 100 milliseconds up to 2 seconds, and significant delays and losses start around the rate of 100 requests per second.

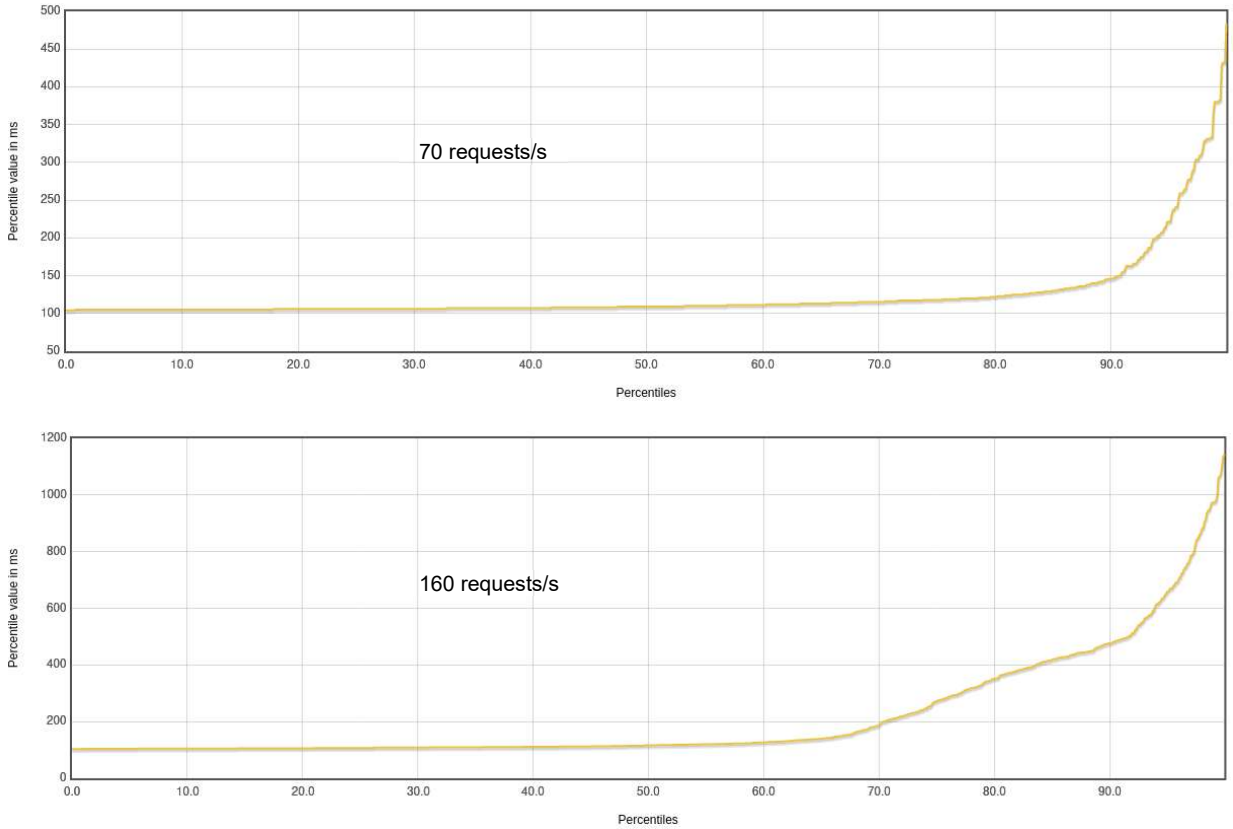

Figure 20: Performance measurement of APEX: response time vs event rate of APEX.

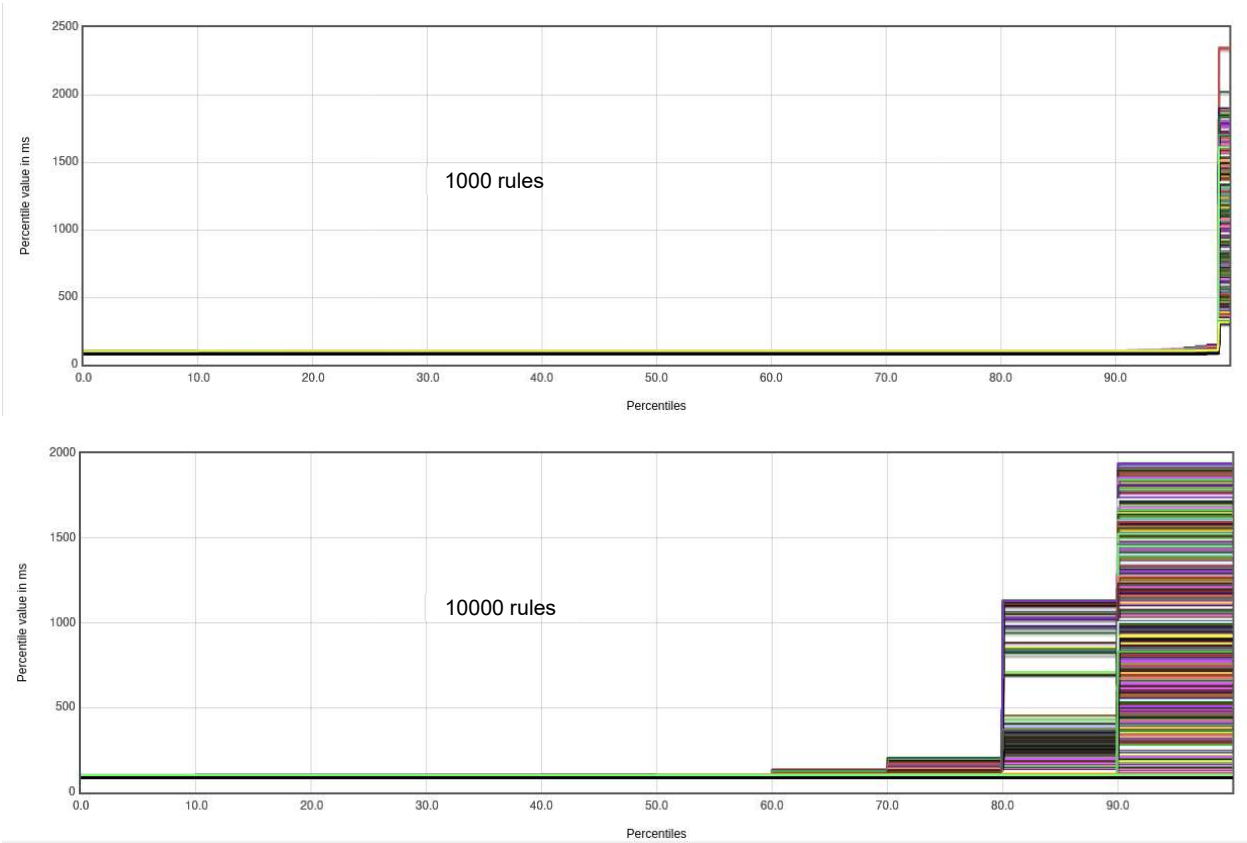

Figure 21: Performance measurement of APEX: response time vs number of policies at the rate of 160 request/s.

The DE via the ACT uses the limited management API exposed by:

- The security functions to update security policies: for instance to whitelist or blacklist an IP flow. These actions are not necessarily intended for firewalls, as in a cloud-native environment like Kubernetes, a *GlobalNetworkPolicy* object provided by the Container Network Interface (CNI) Calico Custom Resource Definition (CRD) can be used to control external connectivity from the cluster.
- The network functions to update configurations of the 5G system such as requesting the SMF to send the updated DNS server address to the UE
- The orchestrator (NFVO, CISM) to manage the VNF/CNF lifecycle and update network services. In the proposed security management architecture, only the D-ASO is responsible for requesting the DMO to modify a network service. In this case, when the PDU Session IP option is not relevant, switching to DNS over TLS mode might be the best solution. The D-ASO instructs the DMO to deploy the function DNS over TLS between the UE and the DNS server, and then instructs the SMF to update the UE with the new secure DNS configuration. According to 3GPP TS 33. 501 [30] , the UE can be pre-configured with the DNS server security

information, or the core network can configure the DNS server security information to the UE. The Protocol Configuration Options (PCO) is a component of NAS message. Its purpose is to transfer external network protocol options IE using the message structure in Figure 22.

. . .	octet x+1 octet y
Container ID n	octet y+1 octet y+2
Length of container ID n contents	octet y+3
Container ID n contents	octet y+4 octet z
Container ID n+1	octet z+1 octet z+2
Length of container ID n+1 contents (see NOTE)	octet z+3 octet z+4
Container ID n+1 contents	octet z+5 octet za

NOTE: If the container ID is:
- 0023H (QoS rules with the length of two octets);
- 0024H (QoS flow descriptions with the length of two octets);
- 0030H (ATSSS response with the length of two octets); or
- 0031H (DNS server security information with length of two octets);

for network to MS direction, then the octet z+3 and octet z+4 indicate the length of container ID contents.

When the *container identifier* indicates DNS server security information with length of two octets, the *container identifier contents* field contains one of the parameters: **security protocol type, port number, authentication domain name, SPKI pin sets, root certificate, raw public key**.

Figure 22: Protocol configuration options information element.

4. Alternate AE algorithms for mMTC and aLTEr attacks

4.1 Objective

In this section, we aim to build an AE that will detect the multiple different threats that will appear within the network. The data of slices was gathered and sent from different kinds of sources, such as servers that are receiving and sending the above network slices. For the implementation and development of the aforementioned engine, the necessary research has been done in order to understand the data and how to use these data in supervised and unsupervised ML models to identify possible anomalies.

Moreover, to successfully develop the alternate AE, it was necessary to create a plan. Therefore, managing software was used to separate the tasks into smaller procedures. In light of this, architecture was designed in a graphical view (see Figure 23) to illustrate the flow of the engine. This has been observed as a supportive way to set goals and deadlines without causing delays to the final output that will be presented. In addition, the usage of the above procedure has proved a necessity to always plan ahead for possible ways to improve the effectiveness and efficiency of the whole procedure.

4.2 ALTER ATTACK – mMTC Attack

We used 2 bunches of datasets that BCOM has provided for testing purposes. BCOM's dataset was about aLTEr attacks. The first bunch of data contained 2000 observations, and the second excel file contained around 800,000 operations. Thus, the partners ensured that the data structure would not change, the data was gathered directly from the DNS server, and the investigation into the data was initialized.
It is essential to mention that the dataset was not provided with identification of the threads (Labels). In other words, it was real-time data in a raw form without any processing. In light of this, an engine for processing them in order to be imported at a later stage into the machine learning models was a necessity to be constructed.

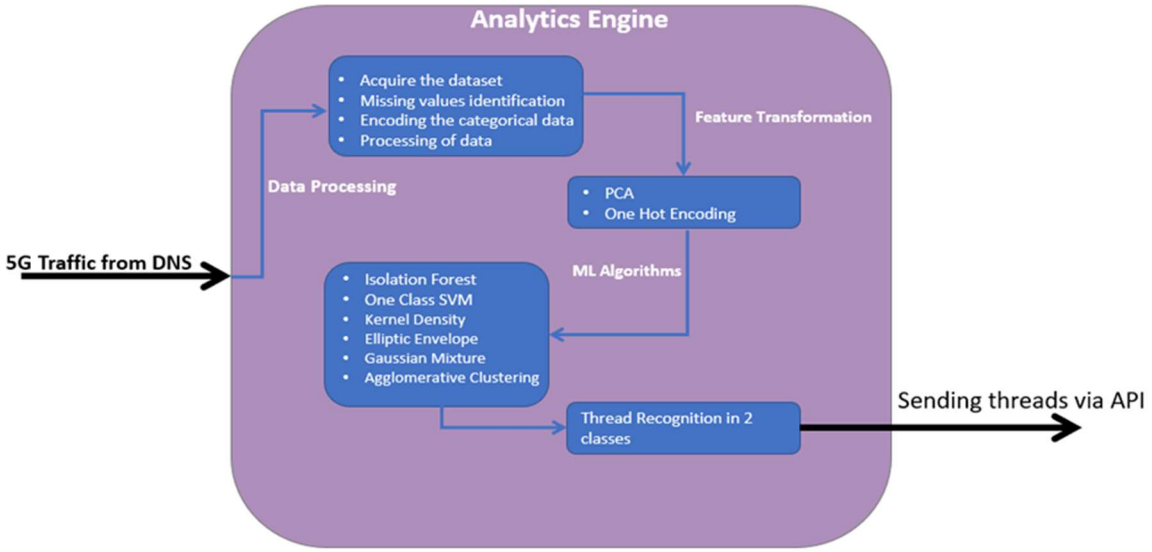

Figure 23: The flow and architecture of the AE.

The goal of the task is to process the provided dataset using simple processing techniques such as cleaning null values and removing unnecessary columns and rows. Then a transformation procedure is used to encode the ID columns into logical integer numbers and reduce dimensionality. Finally, ML models have been implemented to identify data points that differ significantly from other observations. Plots and tables will be presented below, illustrating the results for each task.

4.3 Data Processing

4.3.1 IMPORT TOOL

An import engine has been created in order to receive the data from an API. This procedure can be changed according to the requirements in a future stage when the responsible partner identifies the exact way to send the data. However, for testing purposes, BCOM sent us datasets from CSV. In light of this, we imported the data with a delimiter ',' in order to be able to continue to the next sections.

It was observed that the provided dataset contained a huge number of null values. It was also checked that the columns with null values contained more than half of the range of data, and under these circumstances, the entire columns were ignored. Therefore, missing values were handled by deleting the rows or columns having null values. The rows which had one or more column values null were also ignored.

By deleting the aforementioned columns and rows, a lot of information from the dataset was lost. However the robustness of the model will be achieved when the training session will be executed. In addition, the

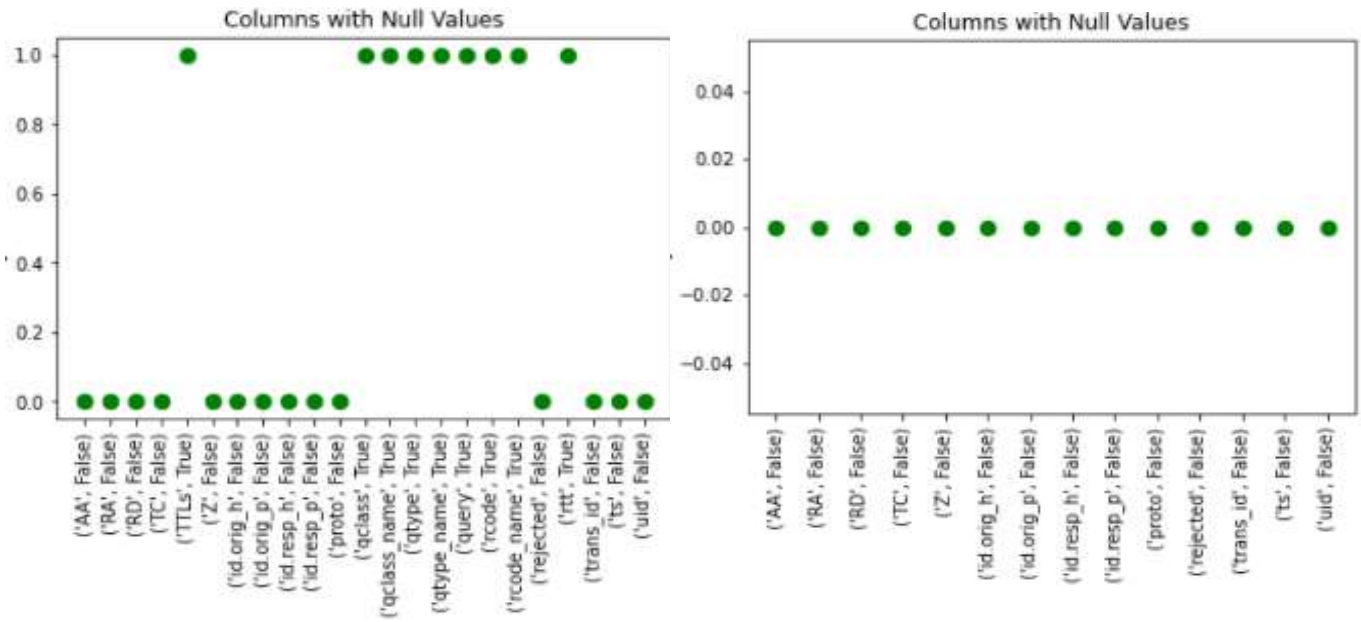

Figure 24: The above plots show the initial vs after cleaning of the dataset

accuracy of the results will be higher without the complete information. Furthermore, the dataset contained ID columns that did not present any meaningful information. These columns were also deleted in order to avoid the misleading detection of anomalies (see Figure 24).

Further, it was observed that many features contained exactly the same values as others. This can be considered a waste of computational performance and a high memory footprint. These features have been deleted to achieve one of the task's goals, which was to improve effectiveness and efficiency.

4.3.1.1 ONE-HOT ENCODING TECHNIQUE

Some of the features were represented as IDs, but it was considered significant for the algorithms that will be used in the machine learning section. Some of these features were the origin IP and destination IP. It is

logical enough that the detection of anomalies is based on their origin and destination IP, so under these circumstances, an encoding technique has been implemented called One-Hot encoding.

This is a process of converting categorical data variables so they can be provided to machine learning algorithms to improve predictions. This procedure converts all ID codes into binary format in order to be accepted and meaningful for all ML algorithms. In light of this, a function has been created to identify all the ID features that remained and map them to integer binary numbers. One-hot encoding is a crucial part of feature engineering for machine learning.

4.3.1.2 CORRELATION TABLE

After the processing stage is finalized, the correlation between the dataset features is required to be able to observe the behaviour of the remaining features. In other words, each feature is correlated positively or negatively with the other. This procedure was constructed to find the pairwise correlations of all columns in the Dataframe.

4.3.1.3 PRINCIPAL COMPONENT ANALYSIS (PCA)

Due to the high dimensionality of the dataset, the PCA technique has been implemented to reduce the components to two. This algorithm transforms the columns of a dataset into a new set of features called Principal Components. Hence, a large chunk of the information across the full dataset is effectively compressed into fewer feature columns. This enables dimensionality reduction and gives the ability to visualize the separation of classes or clusters if any.

The provided dataset has been tested in several components. However, the most accurate for ML was the 2 components which are illustrated below in Figure 25. It is observed that the 2 components suit our purposes as the anomalies are easiest to be clarified.

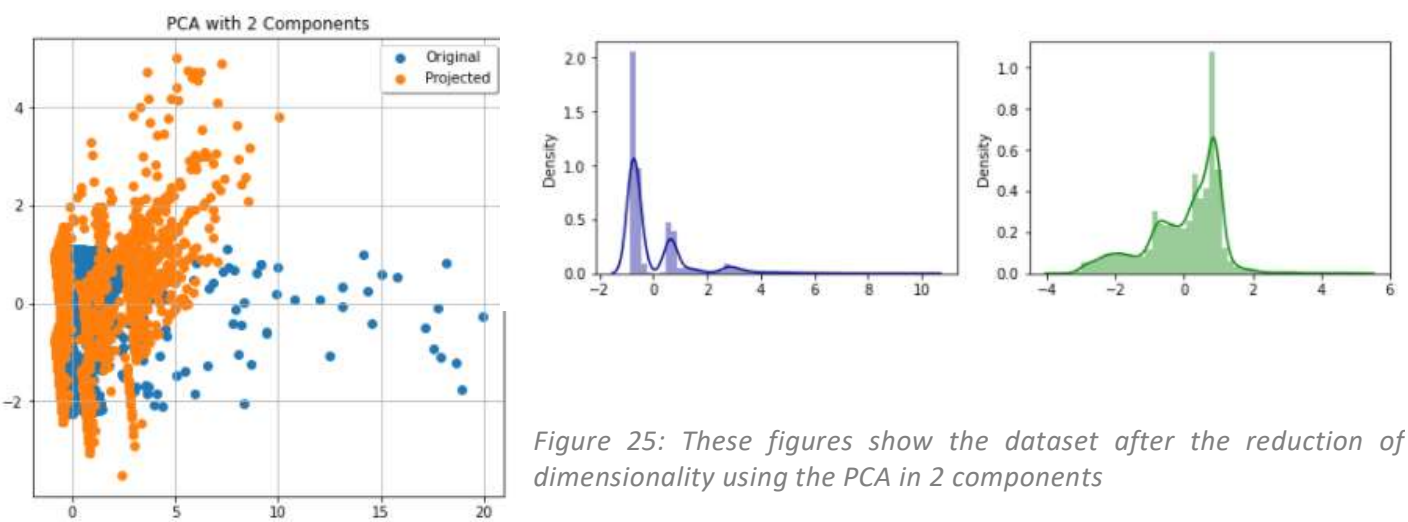

Figure 25: These figures show the dataset after the reduction of dimensionality using the PCA in 2 components

4.4 Machine Learning Algorithms

4.4.1 ALGORITHMS' BRIEF EXPLANATION

Relying on the available dataset, several unsupervised ML algorithms have been implemented. Unsupervised Learning is a machine learning technique in which the users do not need to supervise the model. Instead, it allows the model to work on its own to discover patterns and information that were previously undetected. It mainly deals with unlabelled data.

4.4.2 k-MEANS

k-means is an iterative clustering algorithm that helps to find the highest value for every iteration. Initially, the desired number of clusters is selected. In this clustering method, the data points are grouped into clusters.

4.4.2.1 AGGLOMERATIVE CLUSTERING

This type of k-means clustering starts with a fixed number of clusters. It groups the data into an exact number of clusters. This clustering method does not require the number of clusters k as an input. Agglomeration process starts by forming each data as a single cluster. This method uses some distance measure and reduces the number of clusters (one in each iteration) by a merging process. Finally, we obtain one big cluster that contains all the objects.

4.4.2.2 ISOLATION FOREST

It is a tree-based algorithm (Figure 26) built around the theory of decision trees and random forests. When presented with a dataset, the algorithm splits the data into two parts based on a random threshold value. This process continues recursively until each data point is isolated. Once the algorithm runs through the whole data, it filters the data points that took fewer steps than others to be isolated.

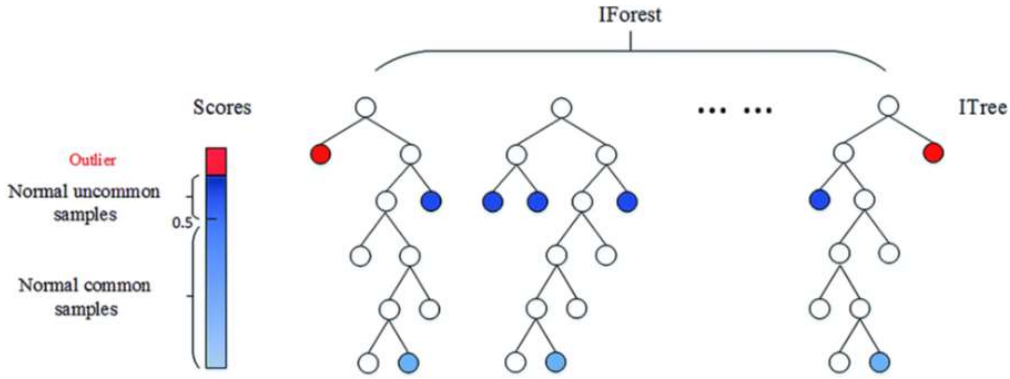

Figure 26: Diagram indicates the procedure that IForest distinguish the anomalies.

4.4.2.3 ONE CLASS SVM

One class SVM separates all the data points from the origin (in feature space FF) and maximizes the distance from this hyperplane to the origin. This results in a binary function that captures regions in the input space where the probability density of the data lives. Thus, the function returns +1 in a “small” region (capturing the training data points) and −1 elsewhere.

4.4.2.4 KERNEL DENSITY

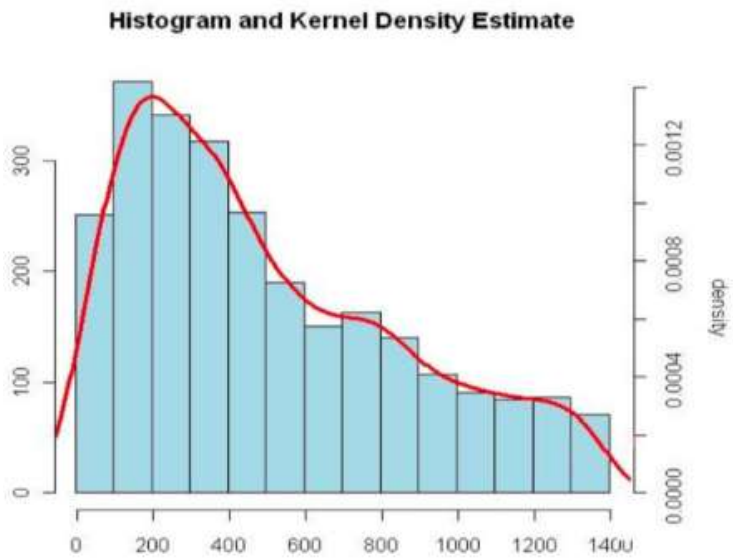

Figure 27: Indicates the Histogram that is created by the Kernel Density Estimation algorithm.

Kernel density estimation is the process of estimating an unknown probability density function using a kernel function $K(u)$. While a histogram counts the number of data points in somewhat arbitrary regions, a kernel density estimate is a function defined as the sum of a kernel function on every data point (Figure 27).

4.4.2.5 ELLIPTIC ENVELOPE

This algorithm's primal purpose is to detect outliers in a Normal distributed dataset. In light of the above, this algorithm creates an imaginary elliptical area around a given dataset (green circle in Figure 28). The values that fall inside the envelope are considered normal data, and anything outside the envelope is returned as outliers. So, naturally, the red data points in the below diagram should be identified as outliers by this algorithm. As evident from this figure, the algorithm works best if data has a Gaussian distribution.

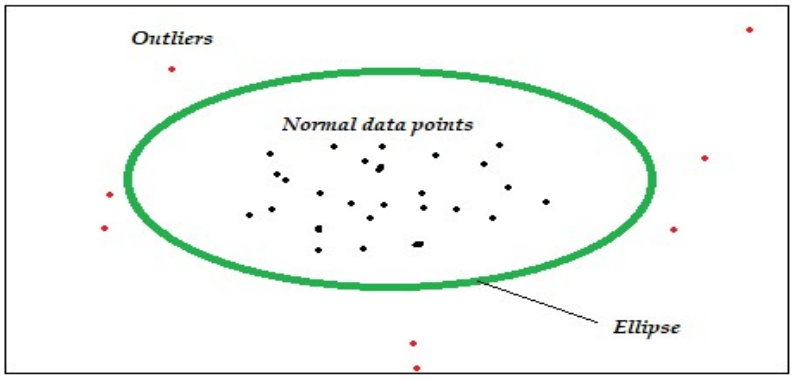

Figure 28: Diagram which illustrates the procedure that the Elliptic envelope algorithm follows

4.4.2.6 GAUSSIAN MIXTURE CLUSTERING MODEL

Gaussian Mixture Models are probabilistic models and use the soft clustering approach for distributing the points in different clusters (Figure 29). In addition, Gaussian mixture models are a probabilistic model for representing normally distributed subpopulations within an overall population. Mixture models in general don't require knowing which subpopulation a data point belongs to, allowing the model to learn the subpopulations automatically.

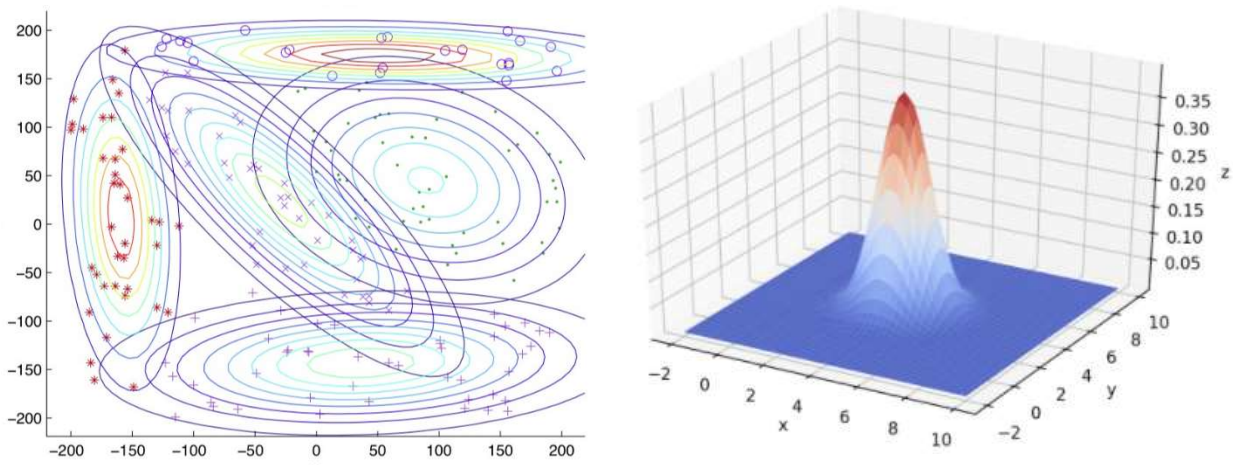

Figure 29: Plots which both show the Gaussian Mixture algorithm clustering method.

4.4.2.7 INTERQUARTILE RANGE (IQR)

The interquartile range defines the difference between the third and the first quartile. Quartiles are the partitioned values that divide the whole series into 4 equal parts. So, there are 3 quartiles (Figure 30). The first Quartile is denoted by Q1, also known as the lower quartile, the second Quartile is denoted by Q2 and the third Quartile is denoted by Q3, also known as the upper quartile. Therefore, the interquartile range is equal to the upper quartile minus the lower quartile.

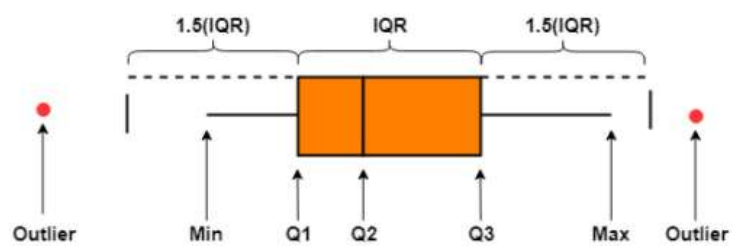

Figure 30: Diagram indicates how the outliers defined by the Interquartile range algorithm.

4.5 Machine Learning Algorithms Results

Several models that were implemented showed interesting results for anomaly detection. All models are plotted in 2-D scatter plots (Figure 31) in order to identify the best-case scenarios. In addition, the dataset contained a huge number of observations, so under these circumstances several tests were executed with a

smaller random range of observations. In this case the models clearly identify the anomalies between the slices of the network.

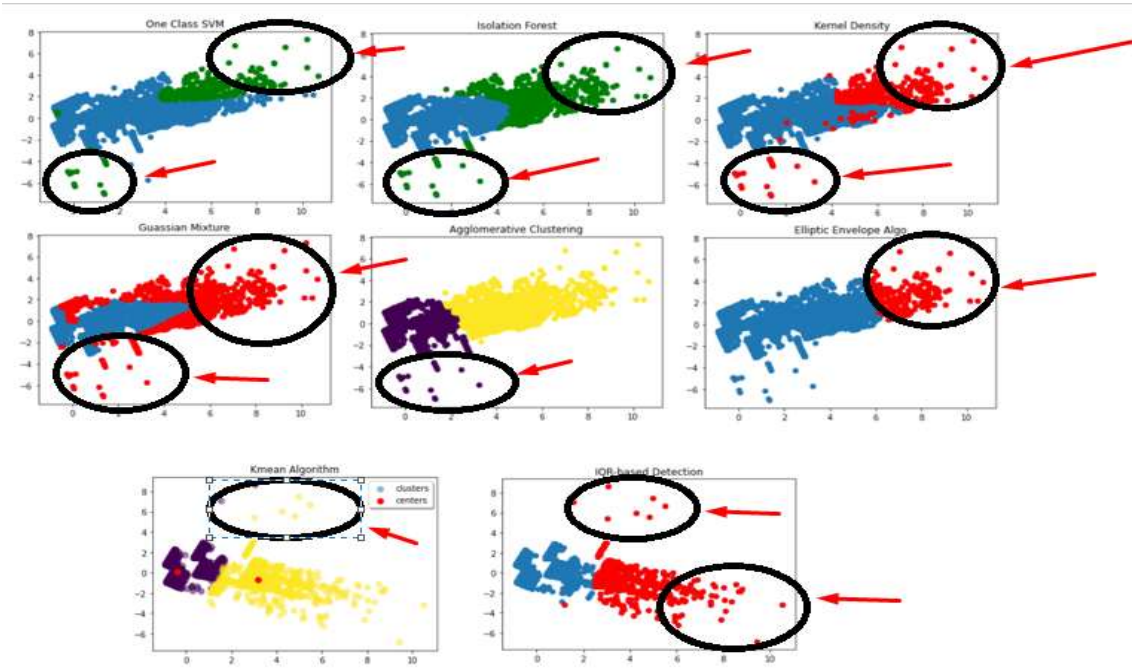

Figure 31: 2D scatter plots for each model.

However, in some cases, the centroids of the datasets confused the clustering and thus the anomaly detection due to the nature of the dataset. Still, it was expected that some of the algorithms, such as k-means, would not be effective in the case of the provided dataset. As we mentioned earlier, the provided data was gathered from an actual real-time DNS server, so under these circumstances, the labeling of threads was not provided.

4.5.1 EURECOM DATASET – MMTC ATTACK

The dataset obtained from EURECOM testbed is a dataset of connections between sources and destinations. Each connection requires a protocol and it logs the source/destination IPs, the execution time, the length of the connection and some basic information. Table 8 below shows a sample of our data:

Table 8: Sample Data set used.

Time	Source	Destination	Protocol	Length	Info
0	192.168.70.134	192.168.70.133	PFCP	60	PFCP Heartbeat Request
0.000029748	192.168.70.134	192.168.70.133	PFCP	60	PFCP Heartbeat Request
0.000326028	192.168.70.133	192.168.70.134	PFCP	60	PFCP Heartbeat Response
0.000346125	192.168.70.133	192.168.70.134	PFCP	60	PFCP Heartbeat Response
0.703005739	192.168.70.132	192.168.70.136	SCTP	100	HEARTBEAT
0.703033581	192.168.70.132	192.168.70.136	SCTP	100	HEARTBEAT

0.703051036	192.168.70.136	192.168.70.132	SCTP	100	HEARTBEAT_ACK
0.703054485	192.168.70.136	192.168.70.132	SCTP	100	HEARTBEAT_ACK

There are different types of protocols, each with its own characteristics for specific operations. Since each protocol is different, they cannot be modelled at the same instant. Each protocol needs to be modelled individually. For this reason, we split our dataset based on those protocols to obtain the following list as shown in Figure 32:

```
$GTP
# A tibble: 88 x 6
   ExecutionTime Source         Destination    Protocol Length Info
           <dbl> <chr>          <chr>          <chr>     <dbl> <chr>
 1     0.000303  192.168.70.136 192.168.70.134 GTP          58 Echo request
 2     0.000269  192.168.70.136 192.168.70.134 GTP          58 Echo request
 3     0.00653   192.168.70.134 192.168.70.133 GTP          54 Echo response
 4     0.00651   192.168.70.134 192.168.70.133 GTP          54 Echo response
 5     0.0000665 192.168.70.136 192.168.70.134 GTP          58 Echo request
 6     0.0000545 192.168.70.136 192.168.70.134 GTP          58 Echo request
 7     0.00165   192.168.70.134 192.168.70.133 GTP          54 Echo response
 8     0.00164   192.168.70.134 192.168.70.133 GTP          54 Echo response
 9     0.0000731 192.168.70.136 192.168.70.134 GTP          58 Echo request
10     0.0000614 192.168.70.136 192.168.70.134 GTP          58 Echo request
# ... with 78 more rows

$HTTP
# A tibble: 2 x 6
  ExecutionTime Source         Destination    Protocol Length Info
          <dbl> <chr>          <chr>          <chr>     <dbl> <chr>
1    0.00000879 192.168.70.129 192.168.70.136 HTTP        168 GET /trigger_event/428/1000 HTTP/1.1
2    0.0000231  192.168.70.136 192.168.70.129 HTTP        191 HTTP/1.1 200 OK  (text/plain)

$NGAP
# A tibble: 414 x 6
   ExecutionTime Source         Destination    Protocol Length Info
           <dbl> <chr>          <chr>          <chr>     <dbl> <chr>
 1     0.204     192.168.70.136 192.168.70.132 NGAP        100 InitialContextSetupResponse
 2     0.0000716 192.168.70.136 192.168.70.132 NGAP        120 PDUSessionResourceSetupResponse
 3     0.203     192.168.70.136 192.168.70.132 NGAP        120 PDUSessionResourceSetupResponse
 4     0.204     192.168.70.136 192.168.70.132 NGAP        100 InitialContextSetupResponse
 5     0.201     192.168.70.136 192.168.70.132 NGAP        120 PDUSessionResourceSetupResponse
 6     0.202     192.168.70.136 192.168.70.132 NGAP        100 InitialContextSetupResponse
 7     0.201     192.168.70.136 192.168.70.132 NGAP        120 PDUSessionResourceSetupResponse
 8     0.201     192.168.70.136 192.168.70.132 NGAP        100 InitialContextSetupResponse
 9     0.204     192.168.70.136 192.168.70.132 NGAP        120 PDUSessionResourceSetupResponse
10     0.203     192.168.70.136 192.168.70.132 NGAP        100 InitialContextSetupResponse
# ... with 404 more rows
```

Figure 32: The created list derived from splitting the dataset based on the protocol.

Since we have execution time and length of the connection, we will utilize those two features and build an anomaly detection methodology. The methodology will rely on a distance metric between the two variables, described by the 'Mahalanobis' function. This function is applied in multi-dimensional datasets and it uses the variance and covariance between the variables in order to extract a 'distance' metric. The derived metric is then modelled via an anomaly detection algorithm to account for any outliers. The function is defined as follows:

$$D^2 = (x - m)^T \cdot C^{-1} \cdot (x - m)$$

Where,

- D^2 is the square of the Mahalanobis distance,
- x is the vector of the observation (row in a dataset).
- m is the vector of mean values of independent variables (mean of each column),

- C^{-1} is the inverse covariance matrix of independent variables.

The output of Mahalanobis is a somewhat normalized variable. Applying a transformation like taking the natural logarithm of that metric, it normalizes it even more. As an example, we can see the difference in density for the 'PFCP' protocol before and after applying the logarithm as show in Figure 33 below:

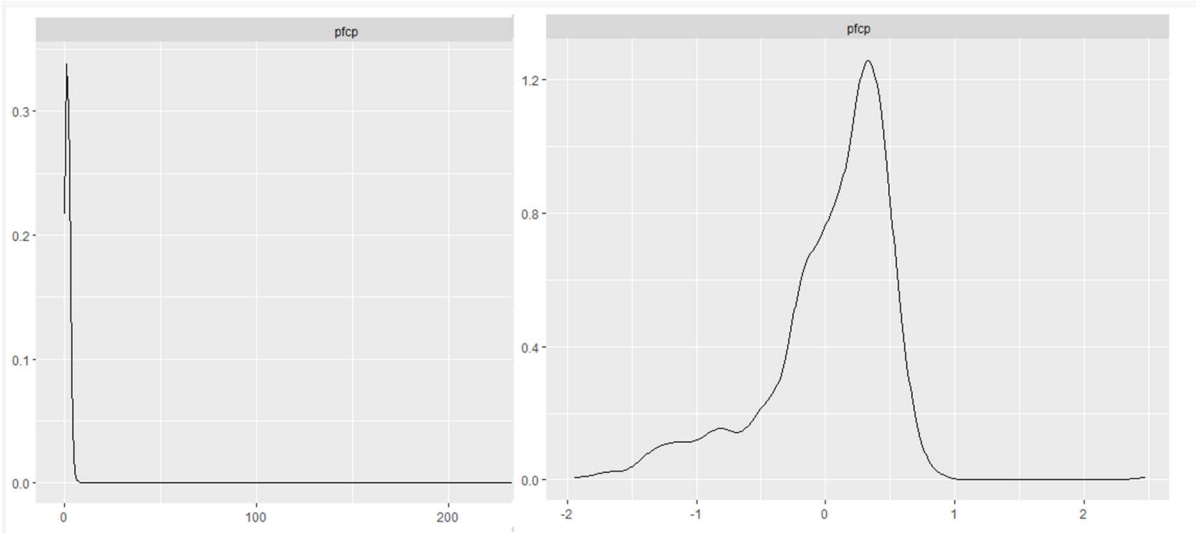

Figure 33: Density plot of PFCP before and after the logarithmic transformation.

In practise, this normality allows us to easily apply a Gaussian filter for our anomaly detection part of the module.

The filter is constructed based on the distance of the points from the mean, counted as standard deviations. As a rule of thumb, this methodology adds (or subtracts in the case of seeking to also find negative outliers) two standard deviations on the mean of the data. However, since only two standard deviations can yield some false positives, we add one more to it and so our anomaly detection function becomes as follows:

$$Threshold = Average(x) + 3 \times (standard\ Deviation)$$

The following plot shows the distances from the mean as per the number of standard deviations away from it,

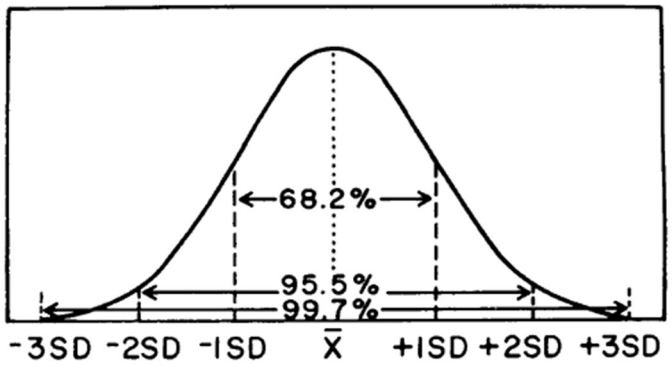

Figure 34: Distances of points from the mean in standard deviations

Applying this methodology for each different protocol, we get the following plots:

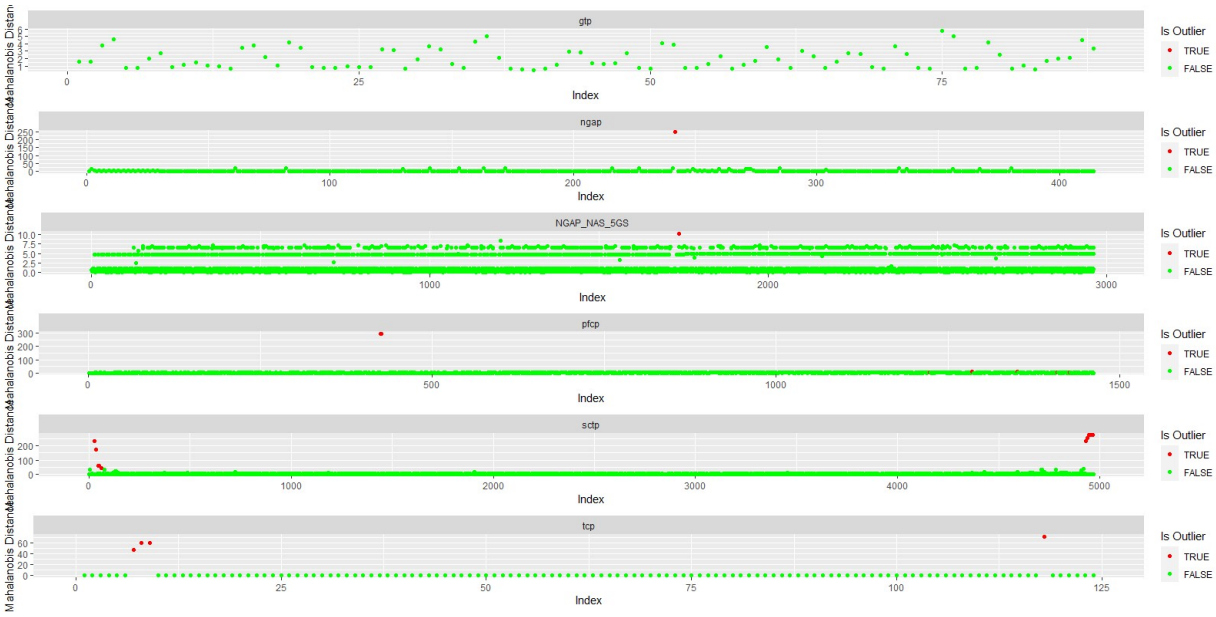

Figure 35: Anomaly Detection applied for all available protocols

We can see that the algorithm seems to be working well without outputting false positives thus creating too much noise. Table 9 below summarises the anomalies:

Table 9: The anomalous points from each protocol as per the model.

Protocol	Execution Time	Length	mahal	anomaly	Info
tcp	6.16E-06	168	47.46	1	NA
tcp	0.022983676	68	60.25	1	NA
tcp	0.022983676	68	60.25	1	NA
tcp	2.31E-05	191	72.37	1	NA
pfcp	0.00639746	60.56898176	294.5410178	1	PFCP Heartbeat Request

pfcp	0.006390159	59.02574863	292.997947	1	PFCP Heartbeat Request
pfcp	0.024958192	115.7751375	7.33032367	1	PFCP Session Establishment Response
pfcp	0.024936189	116.0844674	7.80161582	1	PFCP Session Establishment Response
sctp	30.30003808	63.52952972	234.2804661	1	NA
sctp	30.29999175	64.64177031	234.2873334	1	NA
sctp	25.98480843	99.90206021	171.6257704	1	NA
sctp	25.98480528	100.347178	171.628364	1	NA
sctp	15.31036831	63.83236712	58.12706399	1	NA
sctp	15.31031838	63.48435265	58.12622009	1	NA
sctp	13.45442956	63.94586255	44.56062908	1	NA
sctp	13.45439018	63.95487215	44.56036613	1	NA
sctp	30.16403165	64.62923297	232.1569123	1	NA
sctp	30.16401652	63.40031926	232.1483475	1	NA
sctp	31.74400363	100.3435008	257.7900844	1	NA
sctp	31.74400119	99.81786603	257.7858183	1	NA
sctp	32.76790745	99.34379431	274.9313991	1	NA
sctp	32.76790255	100.2812324	274.9392008	1	NA
sctp	32.77196393	99.90212969	275.0051358	1	NA
sctp	32.771961	99.54585925	275.0020892	1	NA
sctp	32.76396042	99.50220694	274.8655598	1	NA
sctp	32.76395725	100.308901	274.8722915	1	NA
ngap	0.001062125	243.7150432	245.4257304	1	NA

5. Attack on Federated Learning

5.1 Introduction and general solution

Fifth-generation and Beyond (B5G) networks are exponentially growing as key enablers of various applications related to multiple vertical industries [39]. These emerging applications are characterised by heterogeneous requirements, including ultra-low latency, high bandwidth and communication reliability, support for massive device density, etc. To achieve this, B5G systems are based on the network slicing concept, which relies on network softwarization, i.e., the creation of flexible and virtual networks tailored to services, to allow building various applications on top of common physical resources (radio, computation, and network). Indeed, network softwarization is built on top of three new technologies: NFV, SDN, and Cloud computing (central and edge). In addition, a network slice is composed and described by a set of physical and virtual Network Functions (PNF and VNF), interconnected with each other, and deployed on top of a common physical infrastructure.

MonB5G aims to achieve ZSM to automate the management and orchestration of running network slices [40]. This requires heavy usage of advanced Deep Learning (DL) techniques in a closed-loop way to auto-build suitable decisions and meet those requirements. Specifically, the traffic/data generated by network slices are first monitored and then used to build analytic functions (using DL mechanisms) to run in AE. In general, the analytic functions aim to monitor network slice performances or Service Level Agreement (SLA) to predict/detect any degradation or violations. Noting that the KPI of a network slice can be computation-oriented, network-oriented, or service-oriented [41]. Both computation and network-oriented KPIs can easily be monitored, through the NFVI manager and SDN controller, to build analytic functions. However, service-oriented KPIs are difficult to share due to their confidentiality and private nature since they are directly linked to running vertical industry's applications and services, such as the IP address allocation, response/processing time of a particular VNF, and statistics on handled packets by a router.

In this context, FL is playing a vital role in training deep learning models in a collaborative way among thousands of network slice participants while ensuring their privacy, hence network slice isolation. In addition, FL allows to avoid transferring large amounts of monitoring data across the network. This is especially interesting in the ZSM context, in which some autonomic loops are installed on the edge and would be likely to cause perturbations if they report all of their monitoring to a remote location.

Rather than uploading their data to train their models, running network slices share only their local model parameters, during several rounds, with a central entity, e.g. Inter Domain Slice Manager, to aggregate them and build a global model. Thus, the central entity does not have direct access to the training data. Therefore, FL is more than required to create analytic functions about service-oriented KPIs for running network slices while ensuring their confidentiality and privacy. However, FL is vulnerable to poisoning attacks [42] , where an insider participant, a malicious network slice, may upload poisoning updates to the central entity so that it can cause a construction failure of the global model. For instance, a malicious participant may consider poisoned latency values in building its local model in such a way that the aggregated global model cannot then detect or predict latency-related SLA violations. Another example is the analytic function. It implements an intrusion detection system that detects the poisoning models (launched by one or several participant(s)). Thus, poisoning attacks may affect the global performance of FL-based models for running network slices as well as the whole system. Therefore, it is crucial to design security tools to detect and mitigate such threats.

We designed a novel framework to automatically detect malicious participants in the FL process. First of all, based on a real test bed, we generate a realistic dataset that focuses on the latency as a service-oriented KPI of running network slices. We focus here on the latency experienced by the key element of 5G CN on-boarded in a network slice, namely Access & Mobility Management Function. This dataset is then exploited to build an analytic function about latency prediction in a federated way. In addition, the basic idea of our framework is to select dynamically one participant as a trusted entity by leveraging deep reinforcement learning. The trusted participant will be in charge of identifying poisoning model updates using unsupervised machine learning. The main contributions of this work are summarized as follows:

- We use OAI to generate a real dataset about the latency KPI of the AMF running as a VNF. This service-oriented KPI measures the response time for handling UE attach requests when different configurations, such as available RAM memory and the number of CPUs, are taken into account.

- Exploiting EURECOM dataset, we build a DL-based model in a federated way between several running network slices. Our model enables us to predict the latency of the AMF function and hence anticipate any latency-related SLA violations.
- We also build an online Deep Reinforcement Learning (DRL) model that dynamically selects a network slice as a trusted participant at each federated learning round, i.e., when participants send their local models towards the central node, based on their reputations (of the FL nodes).
- At each FL round, the trusted participant applies a dimensionality reduction scheme and unsupervised machine/deep learning to detect the poisoned model(s) and hence malicious participant(s).

5.2 Background and related work

Few solutions have been designed to deal with poisoning attacks when building learning models relaying on FL. These works can be classified into two main categories: works dealing with poisoned local models [43] , [44] and those dealing with data poisoning of malicious participants [45] , [46].

The objective of model poisoning attacks is to poison the local model updates of the FL clients before sending them to the FL server or inserting hidden backdoors into the FL global model. This attack impacts the performance of the global model by giving misclassifications or wrong predictions. In [45], the authors demonstrated how the FL model could be poisoned. They showed that any malicious client could introduce hidden backdoor functionality into the joint global model, e.g., to ensure that an image classifier model can predict labels for some input data, which were introduced (labels) by the malicious client. The authors designed a new model-poisoning attack based on model replacement and evaluated it on top of several assumptions on the standard FL.

Another scheme is proposed with the study of the resilience of distributed implementations of Stochastic Gradient Descent (SGD) against Byzantine failures, including network asynchrony, software bugs, and attackers aiming to compromise the whole system as well as biases in local datasets. The authors first showed that current approaches do not tolerate Byzantine failures. Then, they proposed a resilience property of the aggregation rule that can ensure model convergence despite Byzantine participants.

Although FL introduces new application scenarios in B5G networks, such as edge computing and on-device learning, it inherits the same critical threats, such as the poisoning and membership inference attacks, as in the other contexts. A novel blockchain-based FL scheme was designed in [47] to deal with model poisoning attacks. This scheme is based on blockchain to create smart contracts and hence prevent malicious clients from being involved in the FL process. Therefore, the central server may identify unreliable clients by executing smart contracts to defend against model poisoning attacks. However, the proposed solution is highly expensive.

5.3 The TRUST federated deep learning Framework

In this section, we describe our framework to secure federated learning in B5G networks against poisoning attacks, named TQFL for “Trust deep Q-learning Federated Learning”. The design of our framework comprises four main steps, starting from generating a realistic dataset to design a detection scheme for poisoning attacks: (i) the generation of a realistic dataset about the AMF function's latency of running network slices

and its (latency) related parameters. (ii) Building a deep learning model to predict the AMF function's latency of each running network slice in a federated way in order to prevent any latency-related SLA violation. (iii) Building an online DRL model that dynamically selects a network slice as a trusted participant at each federated learning round. (iv) After the first FL rounds, the trusted participant applies a dimensionality reduction scheme and unsupervised machine/deep learning to detect the malicious participant(s). Before we proceed, we first give an overview of our framework in the next subsection.

5.3.1 OVERVIEW OF TQFL FRAMEWORK

As depicted in Figure 36, we consider n running network slices that may be initiated by different vertical industries, such as intelligent transportation, Industrial IoT, and eHealth verticals. The running network slices are interconnected to an Inter-Domain Slice Manager (IDSM), which is in charge of the management and orchestration of network slices. To enable ZSM, the IDSM side includes an AE for building learning models and a DE to make suitable decisions based on AE's outputs. On the other side, each running network slice is managed locally by a Domain Slice Manager (DSM), which also includes a MS for monitoring data and in-slice traffic (i.e., KPIs) from different VNFs, and an AE for building learning models.

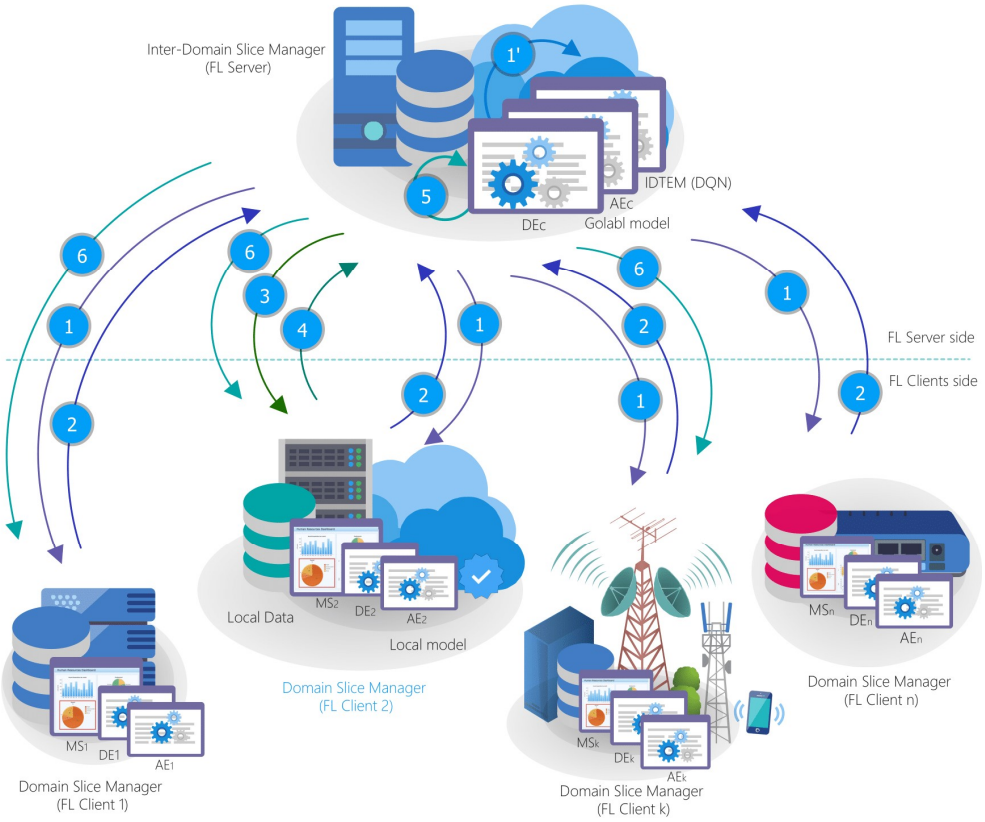

Figure 36 : Overview of TFQL Architecture.

5.3.2 GENERATION OF REALISTIC DATASET

Machine/Deep learning algorithms require data to create learning models. However, the more the datasets are realistic and large, the more the deep learning models are accurate and adaptable for various situations.

Hence, the first critical step toward developing accurate learning models is the data set collection (data acquisition or monitoring).

With the lack of a real dataset, we conducted a real testbed using the Eurecom OAI platform to generate a realistic dataset called EARCD for the Eurecom AMF Resource Consumption Dataset. OAI implements 5G radio access and core networks as open-source software. We emulated ten instances of AMF, running as VNFs inside ten isolated network slices. The network slices differ from each other in terms of their AMFs' configurations. For instance: the AMF of network slice 1 has 1GB of memory, and 1 CPU, the AMF of network slice 2 has 2GB of memory and 2 CPUs, etc. In addition, we used my5G-RANTester [1] tool to emulate UEs and one gNB. my5G-RANTester enables to emulate data and control planes of UEs and gNBs. my5G-RANTester relies mainly on the release 15 of the 3GPP standard for NG-RAN (Next Generation-Radio Access Network). We used my5G-RANTester tool to generate attach request packets, which will be then handled by the different network slices' AMFs. By increasing the number of UEs, we are able to generate up to 560 attach request per second, covering different traffic densities.

Besides, it is clear that the latency values increase as the number of attach request increases. However, the latency values depend greatly on the configurations of network slices' AMFs. Figure 37 depicts the average latency values according to the received number of attach requests for two different configurations of AMFs' VNFs. We clearly observe that AMF with 2048 MByte of memory and 2 CPUs succeeds in decreasing the latency when compared to AMF with 1024 MByte and 1 CPU. Therefore, we generated ten local datasets (ten network slices (n=10)), by varying the number of handled attach request/s, when each dataset contains 2813 samples (rows). In addition, each local dataset contains five features as input data, including RAM capacity ("$ram\ -limit$"), CPU capacity ("$cpu\ -limit$"), RAM used ("$ram\ -usage$"), CPU used ("$cpu\ -usage$"), the number of attach requests ("n"), and latency ("$mean$") in terms of average duration of UEs attachment as output data. The latter corresponds to the response time (latency in microseconds) to handle UE attach requests by the network slices' AMFs. Table 10 shows a Sample Data set used.

Table 10: Sample Data set used.

time	ram_limit	cpu_limit	ram_usage	cpu_usage	n	mean
1636553178	2048	2.000000	75722752.000000	0.003700	10	0.194402
1636553188	2048	2.000000	140255232.000000	0.003700	20	0.301621
1636553199	2048	2.000000	184238080.000000	0.052663	30	0.465659
1636553210	2048	2.000000	217681920.000000	0.052663	40	0.592062
1636553221	2048	2.000000	305606656.000000	0.120889	50	0.862765
1636553232	2048	2.000000	356622336.000000	0.120889	60	0.930602
1636553243	2048	2.000000	412884992.000000	0.249958	70	1.089852
1636553254	2048	2.000000	417267712.000000	0.249958	80	1.123452
1636553267	2048	2.000000	421892096.000000	0.399306	90	2.462652

[1] https://github.com/my5G/my5G-RANTester

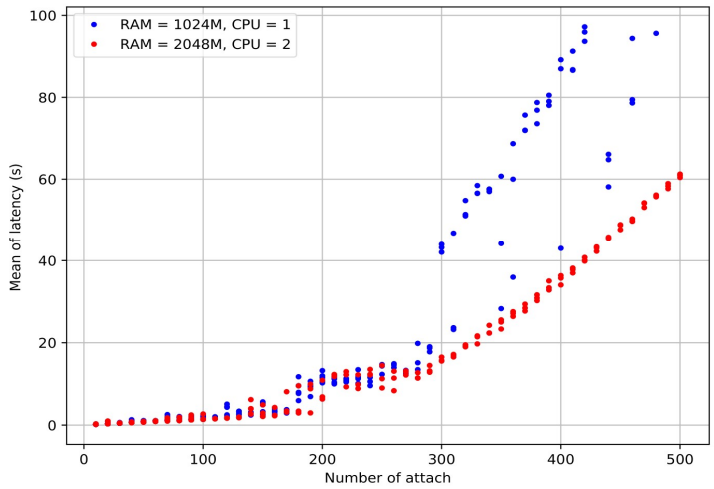

Figure 37: Latency of multiple Attach with different configurations of CPU and RAM.

5.3.3 LATENCY KPI PREDICTION IN FEDERATED WAY

To enable zero-touch management of the latency KPI related to the UEs attach requests, we built a deep learning model that enables each DSM to predict the average needed latency for UEs attach requests in a federated way. As depicted in Figure 36, each running network slice includes MS, AE, and DE; Those elements allow the DSM to monitor needed information (used CPU, used memory, etc.), build local learning models, and make the suitable decisions based on made predictions. Thus, we create a deep learning model in a federated manner that comprises four main steps:

- **Data Pre-Processing**: it is important to pre-process the collected data before feeding the deep learning model since the output of this step may directly affect the performance of the learning model. This step checks if the data is on the same format and scale, does not include null and redundant values, and includes all needed features for the training step.
- **Learning Initialization**: In this step, the IDSM, as the central node, creates an initial global model and defines the learning hyper-parameters in terms of batch sizes, number of epochs, learning rate, neural network architecture, etc. These parameters are then sent to the network slices' DSMs, as clients. Noting that the network slices implement an artificial neural network (ANN), which comprises one input layer of 5 neurons (the five input features), seven hidden layers of 20 neurons, and one layer of 1 neuron (Latency value). In addition, the activation function of the neuron nodes is rectified with a linear function, and two different optimizers are used: Stochastic Gradient Descent (SGD) and ADAM optimizers.
- **Training of Local Models**: each network slice's DSM, through its AE, updates the parameters of its local model. Indeed, each DSM gathers needed data through its MS and splits it into many batches. Then, the DSM's AE performs the average gradient on each batch, with respect to the current model,

during several epochs and with a particular learning rate. Once done, the network slices' DSMs send their local models back to the central IDSM.

- **Building of Global model**: the central IDSM aggregates the DSMs' local models, which is based on distributed SGD, as weights optimizer. The aggregated global model is then sent back to the DSM nodes.

5.3.4 POISONING ATTACK DETECTION

In this section, we present our poisoning attack detection scheme, which begins with the election of a network slice participant as a trusted entity. The latter will then be in charge of detecting malicious clients by leveraging unsupervised learning.

5.3.4.1 TRUST PARTICIPANT SELECTION USING DEEP REINFORCEMENT LEARNING

In each FL round, the central IDSM of running network slices selects a running network slice (DSM) as a trusted node. To do so, we design a new deep reinforcement learning-based model to derive an optimal policy of trust node selection while considering several criteria related to such nodes, such as their reputation, detection rate of malicious nodes, and their accuracy in building learning models.

Deep Reinforcement learning is a process that enables one or a set of agents to learn how to make suitable decisions through error and trial and based on their (agents) previous experiences. Specifically, each agent interacts with the environment to receive either penalties or rewards for actions it makes. Hence, the main objective of deep reinforcement learning is to derive an optimal policy about agents' actions that maximizes agents' cumulative reward.

In our study, the central IDSM is the agent that interacts with running network slices' DSM (environment) in discrete time steps, as shown in Figure 36. In the next section, we first formalize our problem using Markov Decision Process (MDP). Then, we apply the Deep Q-Network algorithm (DQN) to predict the best trusted participant to select at each federated learning stage.

5.3.4.2 MARKOV DECISION PROCESS MODEL

The problem is often modelled using a MDP, where, at every timestamp t, the central IDSM manager, is in a state s_t, takes an action a_t. Then, it (IDSM) will receive a scalar reward from the network slices' DSM (environment). In addition, the system passes from the state s_t to a state s_{t+1}, according to environment dynamics $p(s_{t+1}|\, s_t, a_t)$. Therefore, the central IDSM manager attempts to learn a policy $\pi\ (a|s)$, that helps to map from observations to actions, and maximizing its rewards.

In our study, a state s in S is a three-sized tuple (MC; TC; GMA), where:

- MC_i is the number of malicious models (participants), that are detected by network slice's DSM_i;
- TC_i is the number of trust models, that are detected by network slice's DSM_i;
- GMA_i is the accuracy of the global model, after detecting and removing malicious models, by network slice's DSM_i ;

At every timestamp *t* and when aiming to take an action a, the central IDSM can select a network slice *i*, among the *n* running slices, that has the highest reputation (*Rep)*. Then, the system may transit to a new

state s_{t+1} , that corresponds to the number of detected malicious clients MC_i, the number of trust clients as well as the new accuracy of the new global model.

Furthermore, when the system moves to a new state s_{t+1} , a reward $r_{t+1} = R(s,a)$ is associated with the transition $p(s_{t+1}| s_t, a_t)$. For this end, we model the reward that the central IDSM expects to get from the selected network slices' DSM, as follows:

$$r_{t+1}=\begin{cases} GMA_{t+1} & if\ MC_i \geq 1\ or\ GMA_{t+1} \geq GMA_t \\ 0 & otherwie \end{cases}$$

It is clear that the received reward from a participant *i* affects directly its reputation in the federated learning process. Additionally, the reward increases, when the reputation of the network slice *i* increases as follows:

$$Rep_i = Max(0, r_{t+1})$$

Based on equations 1 and 2, the central IDSM's reward will be affected not only by the number of observed malicious participants but also by the accuracy of the new global model. Therefore, the IDSM manager aims to derive the optimal policy in selecting a trusted participant, which optimizes the learning model's accuracy and its detection of poisoning attacks.

5.3.4.3 DEEP REINFORCEMENT LEARNING

As mentioned before, our reinforcement scheme is based on DQN, which combines Q-learning and a deep neural network to learn an optimal policy from input data. Q-learning is a reinforcement learning algorithm that aims to identify the optimal policy of action selection, maximizing the total reward, called Q-value, for any finite MDP. The Q-value of each action is calculated and stored on a table named Q-table. In addition, at every timestamp (learning episode), the Q-values are updated using the following formula:

$$Q(s_t, a_t) = Q(s_t, a_t) + \Theta(r_{t+1} + \lambda\, max_{a \in A}\, Q(s_{t+1}, a))$$

With Theta is the learning rate and lambda is the discount factor, indicating the importance of future rewards. Besides, the basic idea of DQN is to use a neural network to predict the Q-value of all possible actions based on the current state. In particular, DQN comprises two neural networks: target network Q'(s'; a; theta') and a prediction network Q(s; a; Θ). The prediction network is updated after each learning epoch, while the target network is directly updated from the prediction network after every several iterations. Hence, DQN aims to reduce the loss function between both neural networks as follows:

$$L = \left(r + \lambda\, max_{a' \in A}\, Q'(s', a', \theta') - Q(s, a, \theta)\right)^2$$

Where theta is the learning weights of the Q-network that is updated using gradient back propagation optimizer. In our study, we implement a fully connected neural network. It comprises one input layer with three neurons that correspond to the three states (MC, TC, GMA), two hidden layers with 24 neurons each, and one output layer to predict the Q-values of the running network slices. Noting that, we tried several configurations in terms of the number of intermediate layers and their number of neurons. We selected the best configuration, which provided better performance.

5.3.4.4 DIMENSIONALITY REDUCTION OF MODEL UPDATES

Once a trusted client is selected, it will be in charge of detecting whether the received updates include a malicious model or not. First of all, the selected trust client receives the n model updates of network slices from the central IDSM, and then applies dimensionality reduction techniques to be able to present the model

updates in 2D dimensions. Indeed, dimension reduction is the transformation of a dataset from a high-dimensional space into a low-dimensional space; in such a way, the low-dimensional representation retains the meaningful properties of the original data. In our study, to design an efficient detection scheme, we are based on two different techniques: Principal Component Analysis (PCA) as an unsupervised technique (ignores class labels) and Linear Discriminant Analysis (LDA) as a supervised technique.

5.3.4.4.1 PRINCIPAL COMPONENT ANALYSIS (PCA)

PCA is a statistical technique that helps explain data of high dimensions by extracting only some principal components of such data. The process of PCA comprises four main steps:

- *Standardization of model updates' values:* the trusted client receives **n** model updates, where each one contains five values that correspond to the five input features, including RAM capacity, CPU capacity, RAM used, CPU used, and the number of attach requests. The first step of standardization consists of putting the feature values in the same range and format in order to make sure that they will equally contribute to the final analysis, using the following equation:

$$Str(value) = \frac{value - mean}{standarddeviation}$$

- *Covariance Matrix:* The second step is the calculation of the "Covariance Matrix" between the input features of our dataset. Notably, the aim is to understand how these input features vary from the mean with respect to each other. The covariance matrix is a m × m symmetric matrix (where m is the number of input features, 5 in our case). In fact, the covariance matrix describes the correlations between all the possible pairs of our input features as follows:

$$Cov(input\ features) = \begin{pmatrix} Cov(x_1,x_1) & Cov(x_1,x_2) & \dots & Cov(x_1,x_m) \\ Cov(x_2,x_1) & Cov(x_2,x_2) & \dots & Cov(x_2,x_m) \\ Cov(x_3,x_1) & Cov(x_3,x_2) & \dots & Cov(x_3,x_m) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ Cov(x_m,x_1) & Cov(x_m,x_2) & \dots & Cov(x_m,x_m) \end{pmatrix}$$

The correlation between the two features depends mainly on the covariance sign. If it is positive, both features increase or decrease together, which means they are strongly correlated. Nevertheless, if it is negative, one increases whereas the other decreases, which signifies that they are inversely correlated.

- *eigenvectors and eigenvalues of the covariance matrix*: eigenvectors and eigenvalues are the core of PCA, enabling us to identify the principal components. These parameters are new variables that are constructed as mixtures or linear combinations of the initial features. Indeed, the eigenvectors (principal components) describe the directions of the new feature space, while the eigenvalues show the variance of the data along the new feature axes. Both parameters are calculated using the following formula:

$$Cov(inputfeatures) * v = \lambda * v$$

Where:

- *Cov* (input features) is the covariance matrix

- v is the eigen vectors
- λ is the eigen value

Therefore, eigenvectors and eigenvalues enable us to create new g-dimensional data that gives g principal components.

- *Computation of feature vector:* This step aims to select which principal components to keep and which ones to remove (those that have lesser significance or lower eigenvalues). The output of this step is a matrix of vectors named "Feature Vector", which has the selected components as columns. Finally, the initial data are aligned with the new principal component, using:

$$FinalData = FeatureVector^T * StandardizedData^T$$

In addition, the feature vector with Eigenvectors are used to align/reorient the data from original axes to the new principal component axes.

5.3.4.4.2 LINEAR DISCRIMINANT ANALYSIS (LDA)

LDA is also a dimensionality reduction technique that aims to find not only the component axes, maximizing data variance (PCA), but also a feature subspace that maximizes class separability. Thus, LDA provides the ability to project a feature space (a dataset with *nb* dimensional samples) into a smaller subspace *k* while maintaining the class-discriminatory information.

The LDA process also comprises different steps. First, we consider a matrix of n classes of model updates that correspond to the n clients (network slices) as follows:

$$C = \begin{pmatrix} C_1 \\ C_2 \\ \dots \\ C_n \end{pmatrix}$$

And each class comprises five input features, as follows:

$$F = \begin{pmatrix} F(1,c_1) & F(2,c_1) & \dots & F(5,c_1) \\ F(1,c_2) & F(2,c_2) & \dots & Cov(5,c_2) \\ Cov(1,c_3) & F(2,c_3) & \dots & Cov(5,c_3) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ Cov(1,c_n) & Cov(2,c_n) & \dots & Cov(5,c_n) \end{pmatrix}$$

- *Computing the d-dimensional mean vectors*: The first step of LDA is to compute a d-dimensional mean vector $M(C_i)$ for the different classes n.
- *Compute the Scatter matrix (in between class and within the class scatter matrix) :* The second step is to compute the "Scatter Matrices" which are equivalent to the variance. In fact, we have two matrices of E x N dimensions to calculate: "The within-class" and the "between-class scatter matrix". The objective of these matrices is to determine the variability within a class (Intra class scatter) as well as between different classes (inter-class Scatter). Both consist of calculating the distance between different points (inter/intra classes). The first matrix (within-class scatter) *Sca* is calculated as follows:

$$Sca = \sum_{j=1}^{n} Ss_j$$

Where:

$$Sca = \sum_{x \in D_i}^{number-of-sa} (x - m_i)(x - m_i)^T$$

Ss_j is a scatter matrix for every class,

M is the mean vector,

$$m_i = \frac{1}{n_i} \sum_{x \in D_i}^{number-of-sample} x_k$$

Likewise, the class-covariance matrices are computed by adding the scaling factor 1/ (N – 1) to the within-class scatter matrix so that the precedent equation becomes:

$$\xi_i = \frac{1}{N-1} \sum_{x \in D_i}^{number-of-sampl} (x - m_i)(x - m_i)^T$$

$$Sc_j = \sum_{i-1}^{n} (N_i - 1)\xi_i$$

The between-class scatter matrix SB is computed by the following equation:

$$S_B = \sum_{i=1}^{n} N_i (m_i - m)(m_i - m)^T$$

- *Computation of the eigenvectors and eigenvalues for the scatter matrices:* After that, the next step is the computation of the eigenvectors and corresponding eigenvalues for the scatter matrices, as we did for PCA.
- *Selecting linear discriminant for the new feature subspace:* Following, the sort of eigenvectors should be applied by decreasing eigenvalues and choosing *k* eigenvectors with the largest eigenvalues to form a *d * k* dimensional matrix (where every column represents an eigenvector).
- *Transforming the samples onto the new subspace:* Next, the selected $d * k$ eigenvector matrix will be used to transform the samples onto the new subspace dimension.

5.3.4.4.3 DETECTION OF THE MALICIOUS NODES USING CLUSTERING ALGORITHMS

Once applying dimensionality reduction techniques and depicting network slices' updates in a 2D plan, the last step consists of grouping the received updates into several clusters, in order to determine malicious updates/models. To ensure an effective detection of malicious updates, we chose to apply two different clustering algorithms. The first is k-means which is an unsupervised learning algorithm. It considers no labeled update models' data. The second is KNN, a supervised learning algorithm. It considers labeled data about model updates. Both algorithms aim to divide the received update models, at each FL round, into k clusters that share similarities and are dissimilar to the model updates belonging to another cluster. We note that we

leverage the model update of the trusted participant as a reference that helps us to make the difference between malicious and trust models.

5.3.5 TEST RESULTS

5.3.5.1 EXPERIMENT SETTING

We developed the main components of our framework in terms of FL-based latency model, DQN-based participants selection model, clustering and dimensionality reduction-based attack detection model, using the TensorFlow Python library and leveraging our EARCD dataset. We evaluate our FL-based model to predict the latency KPI, in terms of Mean Squared Error (MSE) on top of two different weight optimizers (SGD and ADAM) in order to determine a suitable optimizer that improves the prediction accuracy of our both FL-based and attack detection models. We note that we evaluate our FL-based model with and without the presence of malicious nodes to show the impact of poisoning attacks on the performance of our FL model. The malicious nodes (network slices) generate poisoning attacks by introducing new malicious/incorrect data samples and building their local models on top of such data. For example, malicious nodes may have high latency values, even when receiving a low number of attach requests and vice versa. In addition, we trained our DQN model for more than 5000 learning episodes. Once converged, we deployed our DQN-based model at the inter-domain slice manager, which is in charge of selecting a trusted participant at each FL round. Moreover, to evaluate our attack detection scheme, after a given number of FL training rounds (r < R'), the ten network slices' DSMs send their model updates to the inter-domain network slice manager. The latter first selects one network slice as a trusted party before sharing with it the model updates. Table 11 gives more details about the parameter settings used in our simulation.

*Table 11: The parameter **settings.***

The parameters settings	**Value**
Federated Learning	
Number of layers	4
Number of Neurons	20
Optimizer 1	SGD
Learning rate	0.0001
Optimizer 2	ADAM
Activation function	ReLU
Loss function	MSE
Reinforcement Learning (DQL)	
Number of layers	2
Number of Neurons	24
Optimizer	ADAM
Learning rate	$1e^{-2}$

Episodes	5000
Dimensional Reduction Algorithm (DRA)	
Dimension of LDA/PCA	2D
Clustering Algorithm (CA)	
Number of clusters (k)	1,2

5.3.5.2 EVALUATION OF POISONING ATTACK DETECTION

In this section, we evaluate the performance of our combined dimensionality reduction and unsupervised clustering scheme against data poisoning attacks, and on top of the two different optimizers: ADAM and SGD. We note that for the poisoning attacks, the malicious node tries to inject incorrect latency values, e.g., high latency value, even when the node has high resource capacity in terms of memory and computing.

5.3.5.2.1 COMBINING LDA WITH K-MEANS

Figure 38 and Figure 39*Figure 40* depict the detection results when combining LDA and K-means on top of ADAM and SGD optimizers, respectively. We also vary the percentage of malicious network slices' DSM and show the trusted nodes that are selected at each FL round (nodes in green colour). As we see, our scheme can clearly detect the malicious nodes, even with only one malicious node, i.e., only one malicious node. Specifically, the trusted node applies both LDA and K-means, and then all nodes that are in the same cluster are considered correct models, while the nodes (models) that are in the other (s) cluster (s) will be considered as malicious. Therefore, our trust participant selection algorithm helps us not only to select a trusted node but also to determine malicious nodes when performing dimensionality reduction and unsupervised clustering. Moreover, determining the trusted cluster of nodes will also help the FL server (IDSM) to select a trusted participant for the next FL round.

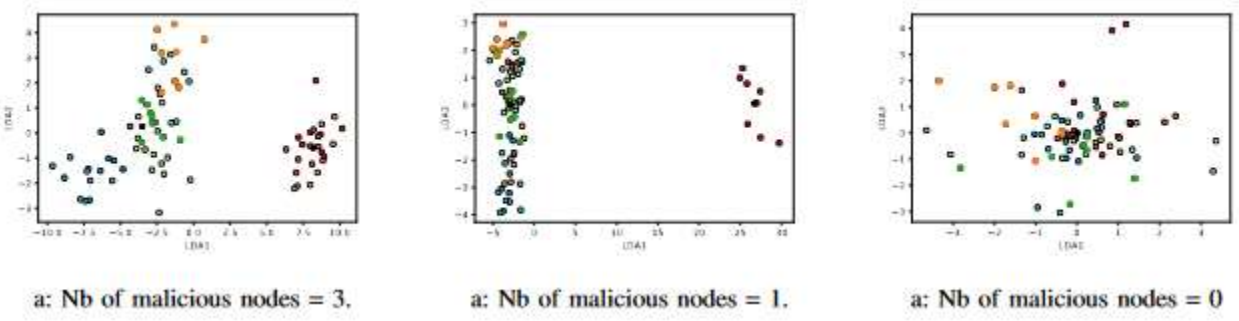

Figure 38: LDA + K-means for different number of malicious nodes (ADAM optimizer).

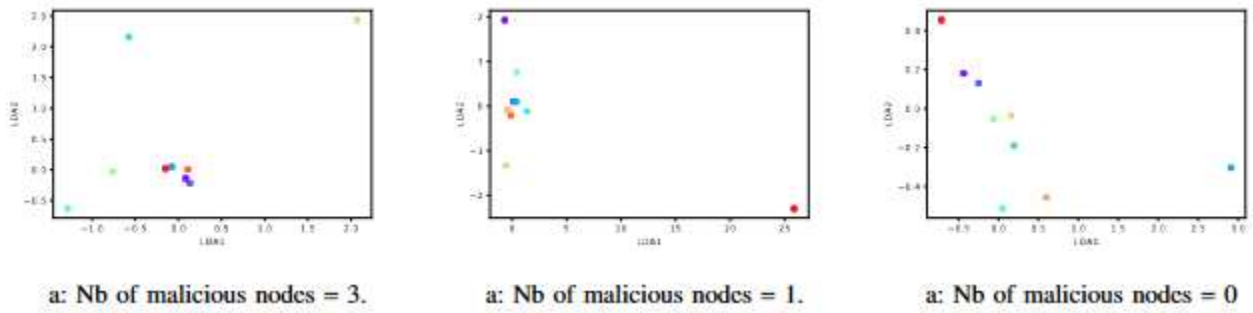

Figure 39: LDA + KNN for different number of malicious nodes (ADAM optimizer).

5.3.5.2.2 COMBINING LDA WITH KNN

Figure 39 and Figure 40 also show the clustering of local models when applying both LDA and KNN on top of ADAM and SGD optimizers, respectively. Whatever the number of malicious nodes, we also observe that there are always some isolated points that represent the infected models sent by the malicious DSMs. However, for the LDA technique, we see that the isolated models (infected) are identified better with the ADAM optimizer than with the SGD optimizer. Hence, the LDA (with KNN or k-means) technique gives better detection on top of the ADAM optimizer. In fact, these last combinations show the clearest clustering (two separate groups) compared to other algorithms.

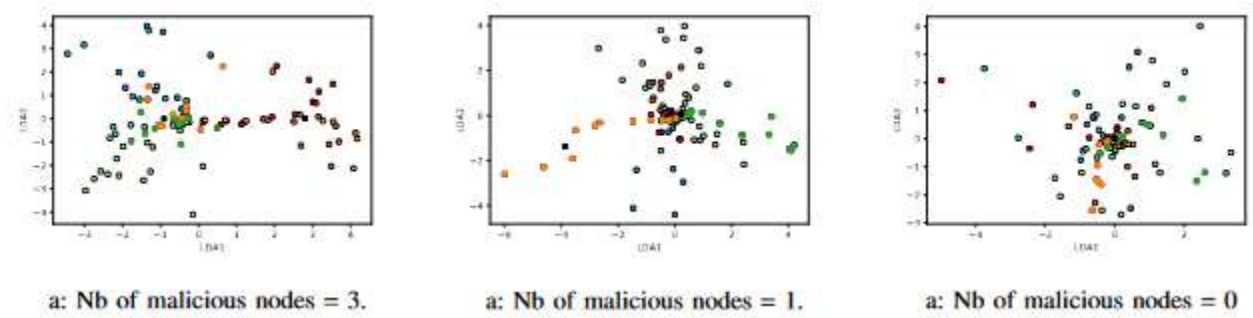

Figure 40: LDA + K-means for different number of malicious nodes (SGD optimizer).

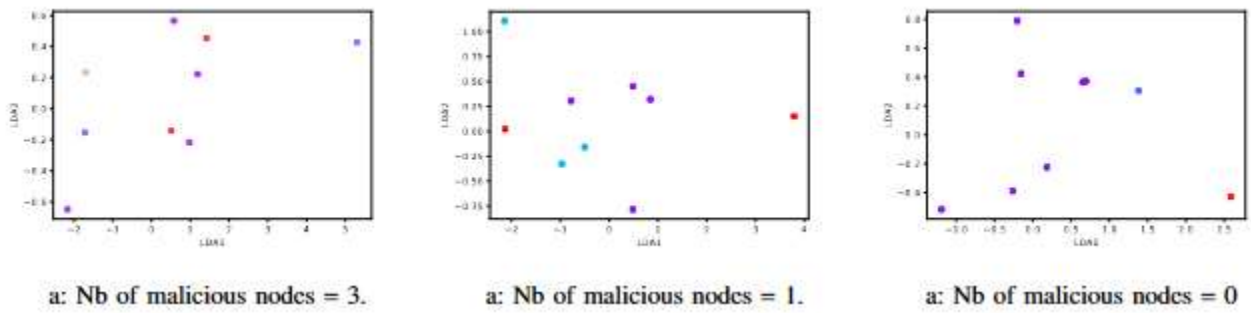

Figure 41: LDA + KNN for different number of malicious nodes (SGD optimizer).

5.3.5.2.3 COMBINING PCA WITH K-MEANS

As we did for LDA, we also evaluated the performance of PCA technique when combined with clustering unsupervised algorithms. Figure 42 and Figure 43 show the clustering detection when combining PCA with K-means, on top of the ADAM and SGD optimizers, respectively. We remark that both optimizers succeed in separating and identifying infected models by incorrect data. However, infected models are better identified on top of the SGD optimizer as compared to the ADAM optimizer. Thus, PCA with K-means gives better performance in detecting infected models on top of the SGD optimizer. Indeed, this last combination shows the clearest clustering (two separate groups) compared to other algorithms.

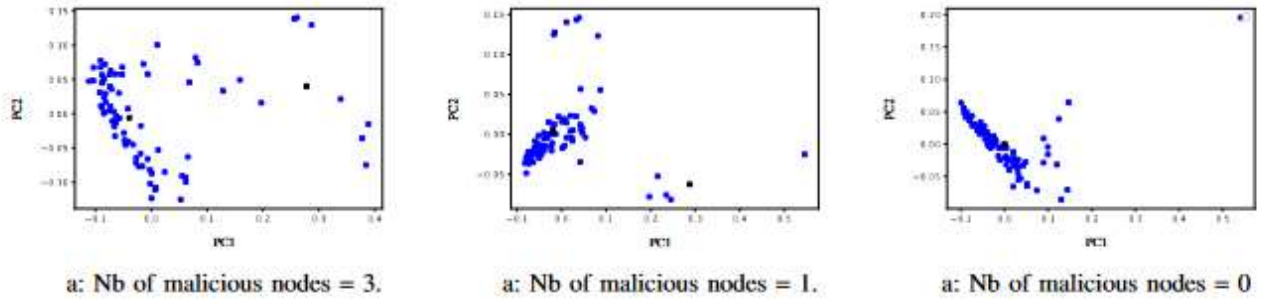

Figure 42: PCA + K-means for different number of malicious nodes (ADAM optimizer).

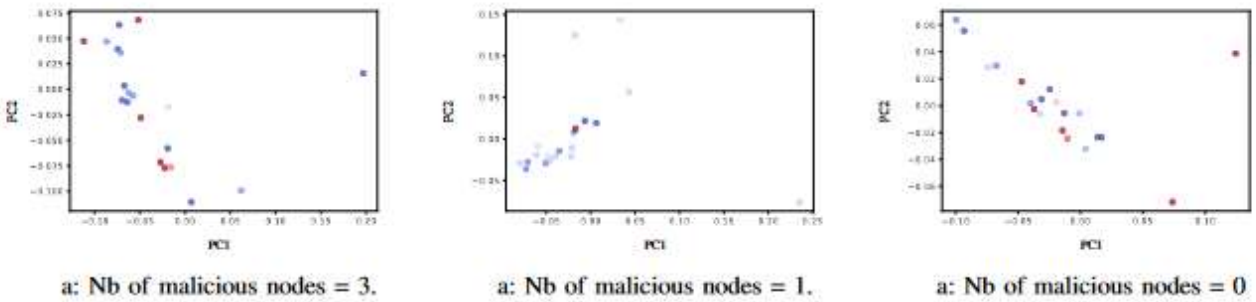

Figure 43: PCA + KNN for different number of malicious nodes (ADAM optimizer).

5.3.5.2.4 COMBINING PCA WITH KNN

Similarly, Figure 44 and Figure 45 depict the detection when combining PCA with KNN on top of the ADAM and SGD optimizers, respectively. As in Figure 42 and Figure 43 PCA with KNN on top of both optimizers clearly separates correct local models from infected ones and thus enables them to detect/identify malicious DSMs. We also see that infected models are better identified when leveraging the SGD optimizer than the ADAM optimizer. Therefore, the PCA technique with either K-means or KNN gives better detection of malicious models on top of the SGD optimizer, which is confirmed in Figure 42, Figure 43, Figure 44, and Figure 45. In fact, these last combinations show the clearest clustering (two separate groups) compared to other algorithms.

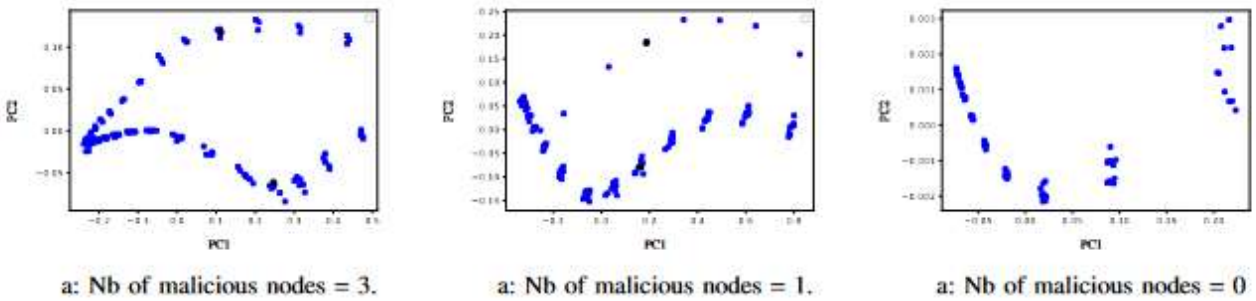

Figure 44: PCA + K-means for different number of malicious nodes (SGD optimizer).

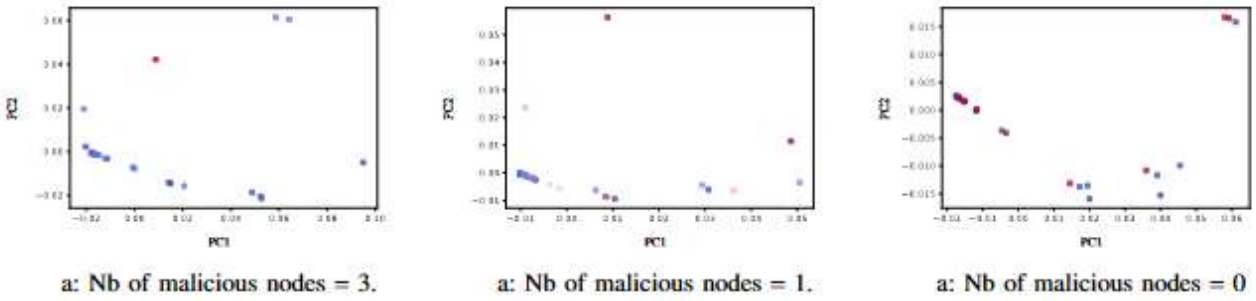

Figure 45: PCA + KNN for different number of malicious nodes (SGD optimizer).

5.4 Conclusions

In the FL attacks scenario, we designed a novel secure federated learning framework, which leverages reinforcement deep learning, dimensionality reduction, and clustering unsupervised learning to deal with both data and model poisoning attacks that can target federated learning-based models in 5G and beyond networks. First of all, we generate a realistic dataset related to network slicing KPIs in order to build a federated learning model for latency KPI prediction. We then generate realistic data and model poisoning attacks on top of our federated learning-based model. To deal with them, a dynamic selection algorithm of a trusted node was first proposed, leveraging reinforcement deep learning. The latter is in charge of detecting and identifying malicious learning models and hence network slices. The numerical results show the efficiency of our framework in dealing with poisoning attacks and thus mitigating such attacks and ensuring stable performances of the federated learning model.

6. Conclusions and next steps

In this deliverable, we detailed the activities conducted by the consortium on zero-touch security management and orchestration of network slices covering the tasks T5.1, T5.2, and T5.4. Besides defining the MonB5G security orchestrator and the SECaaS components, we have devised several solutions demonstrating zero-touch security management using MonB5G key components, i.e., MS, AE, and DE. These solutions were applied to three representative use-cases: mMTC, aLTEr, and poisoning attacks on federated learning training. For each use case, we built a proof of concept in order to show the effectiveness of the proposed solutions to reach zero-touch to detect and mitigate the mentioned attacks, proving a good integration between the different components.

This deliverable complemented the deliverable D5.1 by providing more details on the technical approaches introduced, such as Security Orchestrator, MS, AE, and DE. We presented several AI/ML algorithms that run at the AE to detect and mitigate attacks, while we provided solutions at the DE to mitigate these attacks.

The next step will be to demonstrate, on the EURECOM testbed, in the context of WP6 activities, the solutions developed and validated in WP5 via the three use-case scenarios.

7. References

[1] 5GAmericas , "The evolution of security in 5g, A slice of mobile threats", Tech. Rep, 2019. [Online]. Available: https://www.5gamericas.org/wp-content/uploads/2019/08/5G-Security-White-Paper_8.15.pdf.

[2] Ch. Benzaid, T. Taleb, C.-T. Phan, C. Tselios, G. Tsolis, "Distributed AI-based Security for Massive Numbers of Network Slices in 5G & Beyond Mobile Systems", 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), 2021, pp. 401-406, doi:10.1109/EuCNC/6GSummit51104.2021.9482418.

[3] ETSI, "Management and Orchestration; Sc-Or, Sc-Vnfm, Sc-Vi reference points – Interface and Information Model", ETSI GS NFV-IFA 033, V4.1.1, Aug. 2020.

[4] Arne Holst, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030", January 2021. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/, visited 2021.

[5] 3GPP," Architecture enhancements for 5G System (5GS) to support network data analytics services", TS 23.288 version 16.4.0 Release 16 , July, 2020.

[6] A.Amokrane, A. Ksentini et al., "Congestion control for machine type communications", 2012 IEEE International Conference on Communications (ICC), 2012, pp. 778-782, doi: 10.1109/ICC.2012.6364152.

[7] O. Arouk and A. Ksentini, "Multi-channel slotted aloha optimization for machine-type-communication, In Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems (MSWiM '14). Association for Computing Machinery, New York, NY, USA, pp.119–125, 2014. Available: https://doi.org/10.1145/2641798.2641802.

[8] Ericsson, "A guide to 5G network security 2.0", sep, 2021. Available: https://www.ericsson.com/4a66f8/assets/local/news/2021/09172021-a-guide-to-5g-network-security-2.0.pdf

[9] Jordan Lam and Robert Abbas, "Machine Learning based Anomaly Detection for 5G Networks", IEEE Mobile computing, mar, 2020.

[10] Grant Millar et al, "D2.1: 5G Security: Current Status and Future Trends in Intelligent Security and Pervasive trust for 5G and Beyond", jul, 2020. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2020/05/i5-d2.1_5g-security-current-status-and-future-trends_v1.0.pdf.

[11] Amir Alimohammadifar and Suryadipta Majumdar and Taous Madi et al, "Stealthy Probing-Based Verification (SPV): An Active Approach to Defending Software Defined Networks Against Topology Poisoning Attacks", 23rd European Symposium on Research in Computer Security, ESORICS 2018, aug, 2018.

[12] Antoine Delplace and Sheryl Hermoso and Kristofer Anandita, "Cyber Attack Detection thanks to Machine Learning Algorithms", Computing Research Repository (CoRR), arXiv preprint arXiv:2001.06309, jan, 2020.

[13] João Henrique Corrêa and Patrick Marques Ciarelli and Moises R. N. Ribeiro et al, "On Machine Learning DDoS Attack Identification from Cloud Computing Telemetry", Computing Research Repository (CoRR), arXiv preprint arXiv:2001.06309, apr, 2019.

[14] Rohan DDoShi and Noah Apthorpe and Nick Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices", 2018 IEEE Security and Privacy Workshops (SPW), apr, 2018. pp. 29-35, doi: 10.1109/SPW.2018.00013.

[15] Faisal Hussain and Syed Ghazanfar Abbas and Muhammad Husnain et al , "IoT DDoS and DDoS Attack Detection using ResNet", IEEE 23rd International Multitopic Conference (INMIC), 2020, pp. 1-6, doi: 10.1109/INMIC50486.2020.9318216.

[16] Xiaoyong Yuan and Chuanhuang Li and Xiaolin Li, "DeepDefense: Identifying DDoS Attack via Deep Learning", IEEE International Conference on Smart Computing (SMARTCOMP), ISBN: 978-1-5090-6517-2, 2017, doi: 10.1109/SMARTCOMP.2017.7946998.

[17] Maryam Ghanbari and Witold Kinsner, "Extracting Features from Both the Input and the Output of a Convolutional Neural Network to Detect Distributed Denial of Service Attacks", IEEE 17th International Conference on Cognit, IEEE 17th International onference on Cognitive Informatics Cognitive Computing (ICCI CC), 2018, pp. 138-144, doi: 10.1109/ICCI-CC.2018.8482019.

[18] Roberto Doriguzzi-Corin and Stuart Millar and Sandra Scott-Hayward et al, "Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection", IEEE Transactions on Network and Service Management, 17, ISBN: 1932-4537, June, 2020, 876–889, Available: https://doi.org/10.1109/TNSM.2020.2971776.

[19] Michael Schachter, "DDoS 5G: The Bigger the Pipe, the Stronger the Threat", Allot, June 26, 2018. Available: https://www.allot.com/blog/ddos-5g-the-bigger-the-pipe-the-stronger-the-threat/.

[20] Sean Newman, " Variations on the Mirai Malware Still Feeding DDoS Attacks", Enterprises Beware, 2022. Available: https://www.corero.com/blog/enterprises-beware-variations-on-the-mirai-malware-still-feeding-ddos-attacks/.

[21] Mahmoud Ammar, Giovanni Russello b, Bruno Crispo, "Internet of Things: A survey on the security of IoT frameworks ", Journal of Information, *2018*.

[22] Raja Ettiane, Rachid EL Kouch, "Mitigating Denial of Service Signaling in *Advanced Computer Science and Applications*", International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 12, No. 2, 2021, Available: http://dx.doi.org/10.14569/IJACSA.2021.0120211.

[23] A. S. Mamolar, Z. Pervez, Q. Wang et al, "Towards the Detection of Mobile DDoS Attacks in 5G Multi-Tenant Networks", 2019 European Conference on Networks and Communications (EuCNC), pp. 273-277, doi: 10.1109/EuCNC.2019.8801975, 2019.

[24] Florian Metzger and Tobias Hoßfeld and Andr Bauer et al, "Modeling of Aggregated IoT Traffic and its Application to an IoT Cloud", Proceedings of the IEEE, 1558-2256, doi = 10.1109/JPROC.2019.2901578, mar, 2019.

[25] Markus Laner and Philipp Svoboda and Navid Nikaein et al, "Traffic Models for Machine Type Communications", ISWCS 2013; The Tenth International Symposium on Wireless Communication Systems, aug, 2013.

[26] Hatim Chergui, Adlen Ksentini, Luis Blonco, et al, "Toward Zero-Touch Management and Orchestration of Massive Deployment of Network Slices in 6G", IEEE Wireless Communications, vol. 29, no. 1, pp. 86-93, February 2022, doi: 10.1109/MWC.009.00366.

[27] ETSI, "5G; Security architecture and procedures for 5G System", 3GPP TS 33.501 version 15.1.0 Release 15, jul, 2018.

[28] Mason, Llew and Baxter, Jonathan and Bartlett, Peter et al, "Advances in Neural Information Processing Systems, Boosting Algorithms as Gradient Descent", Advances in Neural Information Processing Systems 12,NIPS 1999, MIT Press, Cambridge, 2000.

[29] Vu, Quang and Ruta, Dymitr and Cen, Ling, "Gradient boosting decision trees for cyber security threats detection based on network events logs", 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 5921-5928, doi: 10.1109/BigData47090.2019.9006061.

[30] 3GPP, TS 33.501; "Security architecture and proccedures for 5G system", 2022.

[31] 3GPP, TS 33.117; "Catalogue of general security assurance requirements", 2021.

[32] ANSSI, National Cybersecurity Agency, "Managing Cybersecurity for Industrial Control Systems ", 2012. ANSSI, 2012. Available: https://www.ssi.gouv.fr/uploads/2014/01/Managing_Cybe_for_ICS_EN.pdf.

[33] ENISA, "Sectoral/Thematic Threat Analysis," 2020.

[34] ETSI, "Group Specifications: Network Functions Virtualisation (NFV) Release 3," ETSI, 2017.

[35] N. Z. Bawany, J. A. Shamsi and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions", Arabian Journal for Science and Engineering, vol. 42, pp. 425-441, 2017.

[36] M. G. Pérez, A. H. Celdrán, P. G. Giardina, G. Bernini, P. Simone, F. J. G. Clemente, G. M. Pérez, G. Festa and P. a. Fabio, "Mitigation of cyber threats: Protection mechanisms in federated SDN/NFV," Concurrency and Computation Practice and Experience, 2019.

[37] MonB5G, "Deliverable D2.4; Final release of the MonB5G architecture (including security) ", MonB5G , 2021.

[38] 3GPP, TS 23.502; “Procedures for the 5G System (5GS)”, 2021.

[39] Shah Zeb, Aamir Mahmood, Syed Ali Hassan, et al, “Industrial digital twins at the nexus of NextG wireless networks and computational intelligence, Journal of Network and Computer Applications”, 2022.

[40] ETSI, “Zero touch network & Service Management (ZSM)”, [Online]. Available: https://www.etsi.org/technologies/zero-touch-network-service-management , visited 2022.

[41] Slawomir Kuklinski and Lechoslaw Tomaszewski, “Key Performance Indicators for 5G network slicing”, IEEE Conference on Network Softwarization (NetSoft), pp. 464–471, June 2019..

[42] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz , et al., “Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning”, the IEEE Symposium on Security & Privacy, 2022.

[43] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov, “How to backdoor federated learning”, Computing Research Repository (CoRR), 2018.

[44] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al., “Machine learning with adversaries: Byzantine tolerant gradient descent”, In NeurIPS, pages 119–129, 2017.

[45] Clement Fung, Chris J.M. Yoon, Ivan Beschastnikh, “Mitigating sybils in federated learning poisoning”, CoRR, 02 Sep 2018.

[46] Gu T., Dolan-Gavitt B., Garg S., “BadNets: identifying vulnerabilities in the machine learning model supply chain”, Computing Research Repository (CoRR), 2017.

[47] Yi Liu, Jialiang Peng, Jiawen Kang, et al., “A Secure Federated Learning Framework for 5G Networks”, IEEE WIRELESS COMMUNICATIONS MAGAZINE, March 2020.