Grant Agreement No.: 871808
Research and Innovation action
Call Topic: ICT-20-2019-2020: 5G Long Term Evolution

INSPIRE-5Gplus

INtelligent Security and PervasIve tRust for 5G and Beyond

D3.3: 5G security new breed of enablers

Version: 1.1

Deliverable type	R (Document, report)
Dissemination level	PU (Public)
Due date	28/02/2022
Submission date	03/03/2022
Lead editor	Gürkan Gür (ZHAW)
Authors	Pawani Porambage (UOULU), Yushan Siriwardana (UOULU), Roshan Sedar (CTTC), Charalampos Kalalas (CTTC), Wissem Soussi (ZHAW), Huu Nghia Nguyen (MI), Edgardo Montes de Oca (MI), Vincent Lefebvre (TAGES), Gianni Santinelli (TAGES), Juan Carlos Caja (TID), Antonio Pastor (TID), Chafika Benzaid (OULU), Othmane Hireche (AALTO), Yongchao Dang (AALTO), Tarik Taleb (OULU), Geoffroy Chollon (Thales), Maria Christopoulou (NCSRD), Pablo Fernández, Alejandro Molina Zarca (UMU)
Reviewers	Chafika Benzaid (OULU), A. Selman Bozkurt (ZHAW), Ayed Dhouha (Thales), Wissem Soussi (ZHAW)
Work package, Task	WP3, T3.3
Keywords	AI/ML, security for 5G and Beyond, smart security, attack detection and mitigation

Abstract

This report describes the work done in the WP3 within the Task 3.3 “Exploration of additional AI techniques and solutions”. This task aims to explore the use of AI/ML driven security for 5G and future networks. It focuses on two aspects of “Security by AI/ML” and “Security of AI/ML”, the latter analysed especially from a software security perspective as the network assets including security functions themselves are becoming increasingly part of a distributed software ecosystem in communication networks. The security enablers investigated and elaborated in this deliverable aim to alleviate security issues in 5G and Beyond networks in a wide range from application layer to hardware-provided security primitives.

Document revision history

Version	Date	Description of change	List of contributors
v0.1		First draft	Gürkan Gür (ZHAW)
v0.2		Initial contributions	All partners
v0.3		Additional contributions	All partners
v0.5		Additional contributions	All partners
v0.6		Additional contributions	All partners
V0.7		Reviewers' comments and corrections	Chafika Benzaid (OULU), A. Selman Bozkurt (ZHAW), Ayed Dhouha (Thales), Wissem Soussi (ZHAW)
v0.8		Final partner revisions	All partners
v0.9		Final revisions and formatting	Gürkan Gür (ZHAW)
v0.91	23/02/22	Final editing, sent for GA approval	Uwe Herzog, Anja Köhler (Eurescom)
v0.92	03/03/22	Final editorial improvements	Gürkan Gür (ZHAW)
v1.0	03/03/22	Final check and submit	Uwe Herzog (Eurescom)
v1.1	21/03/23	Corrections to address final Project Review Recommendations	Gürkan Gür (ZHAW)

List of contributing partners, per section

Section number	Short name of partner organisations contributing
Section 1	ZHAW, TSG
Section 2	ZHAW
Section 3	AALTO, CTTC, MI, NCSRD, OULU, TAGES, TID, TSG, UMU, ZHAW
Section 4	AALTO, OULU, TAGES, ZHAW
Section 5	AALTO, CTTC, MI, NCSRD, OULU, TAGES, TID, UMU, ZHAW
Section 6	AALTO, CTTC, MI, NCSRD, OULU, TAGES, TID, UMU, ZHAW
Section 7	ZHAW

Disclaimer

Acknowledgment

The research conducted by INSPIRE-5Gplus receives funding from the European Commission H2020 programme under Grant Agreement No 871808. The European Commission has no responsibility for the content of this document.

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Executive Summary

This deliverable presents the set of AI/ML (Artificial Intelligence/Machine Learning) driven enablers developed in Task T3.3 of the INSPIRE-5Gplus project. We have focused on main security use cases and integration aspects of AI/ML for cognitive security functions in 5G and Beyond 5G (B5G) networks: Detection and mitigation, data support for cognitive security, robust AI/ML and the security of AI via software security techniques and resilience schemes. Accordingly, this deliverable shows that AI/ML techniques in the pursued research improve the identification and detection of network traffic related to malicious activities, such as Botnet traffic, Distributed Denial of Service (DDoS) attacks, V2X misbehaviour, and Global Positioning System (GPS) spoofing. Additionally, an AI/ML driven approach optimizes security mechanisms in the form of Moving Target Defense (MTD), which requires proactive and efficient decision making.

In the same security context, ML models should be protected against attacks affecting the training or the inference of the model. This aspect is particularly valid when dealing with distributed ML systems as with Federated Learning (FL), a key research topic for 5G/B5G networks. The study and work done on improving FL security, as well as the leveraging of Trusted Execution Environment (TEE) protection of ML models gives promising results in terms of feasibility and performance, highlighting the direction of future research and work. To this end, the developed enablers are presented and discussed in this document.

Essentially, the enablers presented in this deliverable, achieved by the INSPIRE-5Gplus T3.3, contributes to the main objective of the project, identified as obtaining an automated security management and orchestration aligned with the Zero touch network & Service Management (ZSM) closed-loop specification, serving the end-to-end 5G/B5G infrastructure.

Table of Contents

List of Figures

List of Tables

Abbreviations

5G-AKA	5G Authentication and Key Agreement
5G-PPP	5G Infrastructure Public Private Partnership
A2C	Advantage Actor-Critic
A3C	Asynchronous Actor-Critic
AAA	Authentication, Authorization, Accounting
AAE	Adversarial Autoencoder
ADF	Anomaly Detection Framework
AES	Advanced Encryption Standard
AI	Artificial Intelligence
AIaaS	AI-as-a-Service
AML	Adversarial Machine Learning
API	Application Programming Interface
B5G	Beyond 5G
BIM	Basic Iterative Method
BS	Base Station
CAM	Connected and Automated Mobility
CDN	Content Delivery Network
CDNaaS	CDN functionality as a service
CNN	Convolutional Neural Network
CS	Cosine Similarity
CTI	Cyber-Threat Intelligence
CVSS	Common Vulnerability Scoring System
DBN	Deep Belief Network
DDoS	Distributed Denial of Service
DevOps	Development Operations
DoA	Direction of Arrival
DNN	Deep Neural Network
DP	Differential Privacy
DPI	Deep Packet Inspection
DQN	Deep Q-learning Network
DRL	Deep Reinforcement Learning
DRM	Digital Right Management
DTLS	Datagram Transport Layer Security
EPC	Enclave Page Cache
E2E	End-to-End
FGSM	Fast Gradient Sign Method
FL	Federated Learning
FN	False Negative

FP	False Positive
gNB	gNodeB
GCM	Galois Counter Mode
GMM	Gaussian Mixture Models
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GTP	GPRS Tunnelling Protocol
HLA	High-Level Architecture
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IoT	Internet of Things
IPR	Intellectual Property Rights
ITU	International Telecommunication Union
JSMA	Jacobian-based Saliency Map Attack
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
MaaS	Machine Learning as-a-Service
MAE	Mean Absolute Error
MANO	Management and Orchestration
MDP	Markov Decision Process
MEC	Multi-access Edge Computing
ML	Machine Learning
ML5G	Machine Learning for Future Networks including 5G
MLOps	Machine Learning Operations
MLP	Multi-Layer Perceptron
MPS	Mobile Positioning System
MSPL	Medium-level Security Policy Language
MTD	Moving Target Defense
MVSK	Mean Variance Skewness Kurtosis
NDT	Network range-Digital Twin
NFV	Network Function Virtualization
NIDS	Network Intrusion Detection Systems
NMA	Navigation Message Authentication
NS	Network Service
OODA	Observe Orient Decide Act

OptSFC	Optimizer for Security Functions
OS	Operating System
OSI	Open Systems Interconnection
OSM	Open-Source MANO
PCA	Principal Component Analysis
PoC	Proof of Concept
PPO	Proximal Policy Optimization
RAN	Radio Access Network
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RNN	Recurrent Neural Networks
SA	Security Agent
SAE	Stacked Autoencoder
SAF	Security Analytics Framework
SBA	Service Based Architecture
SDC	Security Data Collector
SDN	Software-Defined Networking
SDR	Software Defined Radio
SotA	State-of-the-Art
SSLA	Security Service Level Agreement
SGD	Stochastic Gradient Descent
SGX	Software Guard Extensions
SMD	Security Management Domain
SOM	Self-Organizing Map
TCB	Trusted Computing Base
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TN	True Negative
TP	True Positive
TPM	Trusted Platform Module
TPU	Tensor Processing Unit
UPF	User Plane Function
V2X	Vehicle-to-Everything
vEPC	Virtual Evolved Packet Core
VM	Virtual Machine
VIM	Virtualized Infrastructure Manager
VNF	Virtual Network Function
VXLAN	Virtual Extensible LAN
UAV	Unmanned Aerial Vehicle
UE	User Equipment

USRP	Universal Software Radio Peripheral
WD	Wasserstein Distance
XML	Extensible Markup Language
XRL	Explainable RL
YAML	Yet Another Markup Language
ZSM	Zero touch network & Service Management

1 Introduction

This document is a technical report for Task T3.3 of INSPIRE-5Gplus project. It provides a description and achievements of the WP3 T3.3. T3.3 elaborates on AI techniques by describing how to apply them to secure 5G services and infrastructure. In this deliverable, we organize our contributions for cognitive security into four main categories: Detection and mitigation, data support for cognitive security, robust AI/ML, and the security of AI. To this end, the developed enablers are presented and elaborated.

In Section 2, we discuss the main topic of AI/ML and how it can be used as an inherent part of security in network security. This discussion aims to provide a balanced view on this topic. Then in Section 3, we describe our cognitive security enablers for 5G security via AI/ML. Then in Section 4, we take the alternate point of view and discuss the security of AI/ML itself. This is a crucial discussion since AI/ML controlled functions (including security enablers) can become attack vectors themselves. As a forward-looking project considering Beyond 5G networks, the INSPIRE-5Gplus project has also elaborated on this aspect.

From the implementation and interoperability perspective, the specification of APIs for D3.3 enablers is listed in Section 5. D3.3 enablers will be utilized in Demo 1 and 3 as part of WP5 work in the INSPIRE-5Gplus project. Therefore, we describe this connection to our project demonstrations in Section 6. Subsequently, we provide an overview of how our enablers exploit AI/ML for cognitive operation and how they are embedded in the overall INSPIRE-5Gplus landscape – to be detailed in T3.4. Finally, in Section 8, we conclude with a discussion on future perspective and a summary of our T3.3 work.

2 AI/ML and smart/optimized security techniques for future networks

In this section, we concisely discuss the main topic of AI/ML and its crucial role in network security. This discussion aims to provide a balanced view on the pros and cons of INSPIRE-5Gplus cognitive security enablers for 5G security exploiting AI/ML. Please note that this treatment is not designed to be exhaustive considering the fact there are numerous detailed surveys and tutorials in the literature. Please refer to the sources in the references for such an undertaking. When security via AI/ML is considered, there is also the other side of the coin, which is the security of AI/ML itself, as discussed in Section 4. Moreover, such enablers do not operate in vacuum, but rather as part of a security management and enforcement framework. For 5G and Beyond 5G networks, ETSI ZSM specification is instrumental in that regard. Therefore, we also elaborate on these topics in this section. Since D3.3 enablers will be utilized in Demo 1 and 3 as part of WP5 work, this overview is important for practical aspects of our T3.3 outcomes as well.

INSPIRE-5Gplus has adopted AI/ML driven security enablers, their cognitive operation (in the sense of autonomous and situation-aware operation), and a compliant security management framework as a key focus for 5G and Beyond security in the project work. To this end, WP2 described an HLA architecture for creating a closed-loop system [1]. This loop follows the ETSI Zero touch network & Service Management (ZSM) [2] specification. The ETSI ZSM specification defines the way to automate the management of functions and end-to-end services. In the scope of the INSPIRE-5Gplus project, this management concerns the security applied to services, slices, and VNFs. This automation enables to quickly react to attacks, to dynamically adapt to changes, and eventually to pave the way to predicting self-configuration. Due to a higher complexity and an increase in supported devices or services, it becomes a challenge for a human operator to manage such a system. Moreover, the new 5G ecosystem aims to have a better delivery speed, agility, and adaptability with less operational costs. Autonomous supervision is a key approach to reach this vision.

AI/ML based security solutions are promising due to their various characteristics. Firstly, they allow automation of various security tasks such as malware analysis, log processing, and threat intelligence assimilation for more efficient and rapid operation. Additionally, AI/ML algorithms provide attack detection and classification functions that may evolve for new emerging threats and even alleviate zero-day exploits. They can also deploy optimized countermeasures against security incidents in a proactive and reactive manner without human involvement.

It is worth noting that the integration of AI/ML also brings up some technical challenges. For large-scale and distributed systems like 5G and Beyond 5G networks, a critical issue is the scalability of AI/ML driven security schemes. The computational requirements for training and inference can be quite high. Moreover, providing a continuous integration and continuous delivery supported service through MLOps (Machine Learning Operations) requires different skills expected from data scientists, ML engineers and DevOps (Development Operations) professionals. This is compounded with the communication requirements for distributed and edge/fog computing-based AI/ML schemes. For distributed ML setups such as federated learning, data transmissions themselves should be secured and preserve privacy [3]. Another issue is the data availability for learning schemes. For supervised learning, that may be the quality of training data while for reinforcement learning, that may be the ability to try different actions to get the reward data, which may not be possible due to operational constraints. Therefore, the *policy universe* may not be explored adequately.

The security of AI/ML is another research question that has attracted a lot of attention lately. The role of AI/ML for critical operations such as autonomous management or security in 5G networks brings up the question of trust and reliability of these schemes. Models should be secured and robust in the learning and inference phases (e.g., against poisoning and evasion attacks) [4]. This issue is also related to the loss of control and visibility for security actions once AI/ML driven closed-loop and autonomous security are deployed.

To describe INSPIRE-5Gplus project's contributions via T3.3 efforts, enablers are described and

discussed in the following sections, from the security via AI/ML (from the perspective of specific security functions) to the security of AI/ML. This set of enablers and security use cases is evidently not an exhaustive but yet a critical group for achieving cognitive security for 5G and future networks.

3 Cognitive security enablers for smart 5G and Beyond 5G security

The following sections describe the security enablers developed in the INSPIRE-5Gplus T3.3. These sections explain how each asset is developed and how it goes beyond the state-of-the-art.

3.1 Detection and mitigation via AI/ML in 5G and Beyond systems

3.1.1 MTD control and optimization using AI/ML

3.1.1.1 Problem and challenges

Moving Target Defense (MTD) is a proactive security strategy that changes and moves software and virtual resources to shift the attack surface of a network. Such a strategy aims to increase the difficulty of attackers to perform reconnaissance and forces them to act based on the data gathered in a restricted time frame. However, many challenges are presented against the feasibility of such a plan of action in real scenarios. MTD's main problem is the possibility of disrupting the protected network service, by moving its components on runtime, redirecting traffic, or changing network interfaces. Therefore, MTD strategies should be optimized to reduce their impact and guarantee that the performance requirements are satisfied while ensuring its security utility.

Furthermore, in contemporary telecommunication setups like multi-access edge computing (MEC), edge nodes improve the performance of telecommunication services with better-distributed data and computation made closer to the end-users. However, such nodes have limited resources compared to conventional data centres. Therefore, non-functional components like security solutions should be aware of their resource consumption and minimize the overhead. This is particularly true for MTD operations as, for instance, migrating virtual network functions would require additional resources, multiplying what is already being used.

In order to alleviate these problems and challenges, OptSFC (Optimizer for Security Functions) aims to improve the management of security function configuration, composition, and resource efficiency. Focusing on MTD mechanisms, OptSFC employs ML techniques to train models defining decision policies on the MTD actions to take and considering the constraints mentioned above. Using ML presents further challenges as well. In particular, it can be challenging to explain the choices of a policy generated using deep neural network-based techniques, leading to the concern of explainable AI, though they generally give the best results in terms of performance. This is important as every modification of the network should be "humanly explicable" for accountability reasons. Another challenge is the dynamic nature of 5G/B5G networks, which changes in size and in its characteristics. Hence, the ML model's genericness needs to appraise all possible network conditions and operate with the same performance.

3.1.1.2 State of the art analysis

The optimization of decision problems using ML has been leveraged in many diverse applications using reinforcement learning (RL). Here, an agent interacts with a defined environment using the operations that are made available and learns based on the rewards and penalties taken from the environment's output [5]. The environment is mathematically modelled using the Markov decision process (MDP), a discrete-time stochastic control process used to model the set of states, actions, rewards, and penalties. A model can further define the specific behaviour of the agent, reducing its exploration tree during training and consequently reducing its learning phase cost. This is called *model-based RL* and can be efficient when learning complex tasks, but it can also have limited performance in case the modelled constraints are biased and limit the solution to a local optimum. The so-called *model-free RL*, on the other hand, has greater costs on the training phase, due to its larger exploration space, but generally leads to a model with greater performance by computing the Bellman-optimality [6].

Today, model-free RL techniques are being used the most, being more accessible owing to greater computational resources available, coupled with state-of-the-art ML techniques used to approximate the Bellman-optimality using deep neural networks (DNNs), instead of the classic dynamic programming approach. These gained the name of deep RL (DRL) and include algorithms such as deep Q-network (DQN) [7], Dueling agents [8], asynchronous actor-critic (A3C) [9], NoisyNet [10], or combinations of them (e.g., the DeepMind's Rainbow [11]).

The application of DRL for cybersecurity is gaining widespread attention and has found early usage for MTD optimization. However, these applications are limited to specific environments such as in-vehicle networks, web applications, enterprise networks, or theoretical game-theory models [12] [13]. The MTD optimization in telecommunication networks such as 5G and beyond is still limited and to be explored.

3.1.1.3 Solutions descriptions and advancements

OptSFC is designed as an offline training tool that generates an optimized policy of a security function involving the management and orchestration of virtualized network resources. The RL model resulting from the offline training allows deploying the security function in the 5G network with autonomous and efficient prevention and mitigation actions.

AI and ML contributions

The training phase occurs in a 5G testbed which uses different NFV infrastructures in a Multi-access Edge Computing (MEC) setup. Here, an MDP model collects and processes the testbed's data in near real-time. Such data is also collected at the deployment phase, used by the RL agent for its decision process. The MDP defines the environment as a tuple (S, A, P, R, γ), where:

- S is the set of all possible states. In practice, at each time t, a state S_t is defined by the status of the resources to be protected, i.e., 5G core VNFs and network services (NS) such as the user plane function (UPF), the vEPC, the gNBs at the different base stations, and other NSs used for the network slices management and the functional NSs hosted in the network slices and used by the operator's clients. The status of a resource is defined by its runtime condition (running, idle, voluntarily stopped, or accidentally stopped), the resource consumption (CPU, RAM, and disk), its network metrics (I/O frequency, bandwidth, latency, etc.) and whether an anomaly detection system found it as a target of an attack or an anomaly. The total number of possible states in the environment is $|S| = N_{resources} \times (1 + N_{consumes} + N_{net\_perf} + N_{attacks})$, where $N_{resources}$ is the number of resources to be protected and on which we can perform MTD operations, 1 formally denotes the constant set of features like the runtime condition and other characteristics whereas $N_{consumes}$ and N_{net_perf} are respectively the number of resource consumption metrics and network metrics, which are checked against established requirements. Finally, $N_{attacks}$ is the number of attacks detectable by an anomaly detection system running in the infrastructure. This endows the OptSFC ability to act reactively, against detected attacks, and proactively, based on the measured performance and other passive metrics.
- A is the set of actions $\mathbf{a}_i$, $I < N_A$, the RL agent can take. In practice, in this MDP, the actions are the different MTD operations available for each resource to be protected. Thus, the action space of $|A|$ is upper-bounded by $N_A = N_{MTD} \times N_{resources} + 1$, where N_{MTD} is the number of MTD operations applicable on a network resource, and 1 represents the action of "doing nothing".
- P is the transition probability matrix representing the probability that an action a_i changes a specific state s to the state s', i.e. $\forall\, \mathbf{a} \in A,\ P^{a}_{ss'} = P[S_{t+1} = s' \mid S_t = s, A_t = a]$. This matrix is updated during training and represents the uncertainty of the RL agent in reaching its goal

with a specific action: For instance, the OptSFC agent can decide to perform the action a_i in order to mitigate an ongoing attack, knowing this action could not succeed with a probability of $1-P^{a}_{ss'}$.

- R is the reward function that defines the reward obtained at time *t+1* when performing an action a_i from a state s at time t, i.e., $R^{a}_{s} = E[R_{t+1} / S_t = s, A_t = a]$.In practice, the reward/penalty will be given based on **four factors**:

  1) the status of the protected resource (e.g., a penalty is given when the resource stops accidentally after an agent's action);

  2) the distance of the measured metrics from the established minimum and maximum requirements (e.g., the bandwidth used by a resource is less than what is needed, which can be computed based on the frequency of I/O operations, and inferior to the requirement);

  3) the mitigation of an attack, which is rewarded proportionally to the threat that the attack poses, defined based on the *Common Vulnerability Scoring System* (CVSS) [14], an industry standard. Vice-versa, when an attack occurs, the MDP scores a penalty, i.e., a negative reward. Finally;

  4) the cost of an agent's action, as each MTD action has a different cost in terms of resource consumption and "time to enforce", translated into a negative reward (i.e., a penalty) when the action is performed. Another factor that is considered is the importance of each protected resource with respect to the 5G infrastructure. For example, the vEPC is a core function upon which all network slices depend; thus, it is more critical than other network services. This can be introduced in the MDP through higher rewards or penalties compared to when the same action is performed on a less critical resource;

- γ is the discount factor used to define the importance of the immediate reward with respect to future rewards. This parameter has a value between 0 and 1: a value close to 0 indicates that immediate rewards are more relevant, vice versa, if γ is close to 1, then the future rewards are more relevant than the immediate ones. In practice, our use case is a continuous task, thus, this hyper-parameter will be close to 1, giving more relevance to the long-term rewards, and can be fine-tuned during training.

The policy π will define the behaviour of the agent, i.e., the probability distribution of the actions in *A* for every state in *S*. To optimize the policy, this first needs to be evaluated. Therefore, we have to compute how good it is to take an action *a* when in a certain state *s*. This is done using the Q-function, that gives to each couple (*s,a*) the Q-value defined as follows:

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

which is a recursive function decomposed by the immediate reward (R_{t+1}) of the actual action *a* in the state *s,* summed to the discounted Q-value of the next state (where the next action A_{t+1} is defined by the policy π). Intuitively, the optimal policy π_{max} is the policy that has the greatest Q-value for all states.

In our model, the optimization translates into finding the best policy that compensates for the action cost of the MTD and gets the highest reward based on: 1) the mitigation of an attack, 2) the reduction of a threat, 3) the increased performance of a service/resource, and 4) the cost of the MTD operation in terms of resource usage.

It is important to note that for the reactive functionality, corresponding to mitigating detected attacks, the MDP would rely on an anomaly detection system. Such systems are never 100% accurate, thus, there might be cases where an attack is ongoing or successful but it has not been detected. This situation is represented in our game-theory model as the discrepancy between the network state observed by the RL agent and the real state of the 5G network. We are dealing indeed with an incomplete-information Markov process.

During the training phase in a testbed, as attacks are being simulated, it is possible to measure the

effectiveness of the OptSFC policy against undetected attacks by scoring a "real reward" R' in parallel, differing from the agent's observation as it also scores the rewards/penalties of non-detected attacks. This would help in spotting policies that are well-performing at the proactive point of view.

When generalized or used with a different set of actions A', the MDP model will enable the deployment of autonomous RL agents for various security orchestrations in 5G and NFV architectures, not only limited to MTD.

Last but not least, the explainability of the RL model is another fundamental challenge found in many ML studies related to critical environments, including 5G/B5G networks. RL is concerned as state-of-the-art RL algorithms use DNNs as well, which are not explainable by nature (as stated in Section 3.1.1.1). OptSFC (presented in the previous two Sections) models the MDP and the reward system through expert knowledge and processing of monitoring data. Thus, it is possible to implement different explainable RL (XRL) methods present in the literature [15], such as the XRL via reward decomposition [16]. In this method, rewards can be classified according to semantically meaningful reward types (e.g., resource consumption rewards, QoS rewards, and security rewards), allowing to "explain" the RL agent's decision in terms of trade-offs among these reward types.

Current Implementation

A Proof of Concept (PoC) has been implemented to investigate the potential of DRL for smart control and security decision schemes in 5G environments. This simulates a simplified drone network slice hosted by a 5G operator, which uses two different NFV infrastructures, a core node and an edge node. The limitation of this first simulation is that it omits the integration of network metrics and resource consumption data, focusing on the proactive and reactive functionalities, and the mitigation of detected and undetected attacks.

We train and evaluate six agents using various DRL algorithms: a deep Q learning network (DQN), an advantage actor-critic (A2C), and a proximal policy optimization (PPO), each having two variants based on the deep neural network (DNN) used, namely a multi-layer perceptron (MLP), and a convolutional neural network (CNN). A third variant using long short-term memory (LSTM) was used and then dropped for its inefficient and unprofitable results. The MLP used has 64 layers each having 64 perceptrons each and using the ReLU activation function at each layer. The agent receives the environment's observation as a vector of $|S|$ x c bits, where $|S|$ is defined by the formula in the previous section (excluding $N_{consumes} + N_{net\_per}$) and c is a constant representing the length in bits of a single state s. Each bit represents a type of attack and the targeted resource, and it is set to 1 if the attack is successful and not mitigated, while it is zero otherwise. On the other hand, the CNN takes as input an image, which is the conversion of the observation vector into a B&W image with $|S|$ x c pixels. It performs 3 convolutions, and extracts 512 linear features. To use CNNs, the observation vector is converted into an $|S|$ x c pixel image, each pixel representing a bit in the vector representation. The MDP model has been implemented with OpenAI Gym, and the DRL agents using Stable-Baselines [17] and Pytorch.

3.1.2 Lightweight and space-efficient vehicle authentication enhanced with misbehaviour detection

3.1.2.1 Problem and challenges

Pervasive Vehicle-to-everything (V2X) connectivity and the emergence of effective data-driven methods based on AI/ML drive a paradigm shift towards Connected and Automated Mobility (CAM) services and applications [18]. A key functionality in vehicular systems that can benefit from AI/ML is security, which is essential for ensuring road safety in CAM environments. V2X security threats and attacks can either originate from malicious outsiders which are vehicles/users exogenous to the original system, or insiders which are already authenticated and possess valid credentials to interact with other legitimate entities in the system.

While outsider attacks can be efficiently addressed even in highly dense V2X scenarios with a proper extension of the 5G Authentication and Key Agreement (5G-AKA) procedure, as shown in [19], insider attacks are often difficult to detect and contain, particularly when attackers behave intelligently while conforming to normal system behaviour. For example, an already authenticated vehicle may be able to intentionally transmit false kinematic information (e.g., position, speed, acceleration, heading-angle data) in its broadcast messages and cause disruption in the network. Seemingly abnormal vehicular activity originated from malicious actors (e.g., vehicles) may take the form of highly sophisticated attacks. Such malicious/selfish behaviours from such rogue insiders are commonly referred to as misbehaviours in V2X, and they pose a serious threat when transmitting erroneous/incorrect data in safety-critical situations. Ensuring the semantic correctness of exchanged V2X information is thus of paramount importance.

3.1.2.2 State of the art analysis

In existing literature, several data-driven approaches similar to conventional intrusion-detection systems have been proposed for detecting misbehaving entities in V2X systems [20]. For example, ML-based detection techniques have demonstrated promising results in misbehaviour detection while leveraging vehicular data generated in V2X scenarios [21] [22] [23]. However, current misbehaviour detectors are not designed to dynamically improve their detection experience according to evolving attack patterns in rapidly changing V2X environments. In addition, the use of security thresholds in detectors (e.g., anomaly score-based methods or reputation/trust assessment criteria) limit their applicability to very specific V2X scenarios. For example, when a vehicle exceeds the maximum beacon frequency threshold, its behaviour is considered as a potential denial-of-service attack. The detection threshold must be carefully selected, since a large value may result in excessive losses due to high detection delays, while a small value may result in wasting operational resources on investigating false alarms. Finding, thus, an optimal threshold which optimally balances the detection delay-false positive trade-off, is a challenging problem. To this end, RL can be identified as a highly effective tool that can consistently improve the detection experience over time while interacting with unknown environments without relying on security threshold values [24]. Unforeseen changes in exchanged measurement streams which may go beyond any anticipated variability in traffic behaviour (e.g., due to either naturally drifting mobility patterns or unpredicted malicious activity patterns) in conjunction with insufficient training data pose a challenge (e.g., model over-fitting) to conventional deep-learning-based attack identification methods.

3.1.2.3 Solutions descriptions and advancements

AI and ML contributions

Misbehaviour detection constitutes a sequential decision-making process that can be modelled as an MDP [25]. The action of misbehaviour detection will change the environment based on the decision (i.e., label resolution for malicious V2X information) of either normal or anomalous behaviour at time t; subsequently, the next decision at time step t+1 will be influenced by the changing environment at previous time-step t. In the context of V2X, vehicle's mobility data constitutes a time-series-based report consisting of periodic beacon messages. Each beacon message includes information on the vehicle's speed, position, heading angle, etc., and this information is evolving over time along the vehicle's trajectory. Hence, misbehaving vehicles can be potentially detected by sequentially analysing their mobility patterns using an RL model. In what follows, we briefly discuss the components pertaining to the RL model and applied for misbehaviour detection.

The *agent* is the core part of the RL model. It takes the time-series (i.e., vehicle's mobility data) and prior related decisions as inputs (i.e., state s), and generates the new decision made (i.e., action a) as output. Each action made by the agent is rewarded (i.e., reward r) as feedback, and the agent subsequently updates its model in order to improve the accuracy in decision-making. The iterative model update is performed through Q-learning [26], i.e.,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)),$$

where α and γ denote the learning rate and discount factor, respectively. The *environment* of the RL model controls the training of the agent. It takes the action a performed by the agent as its input, and consequently generates a reward r and the next environment state s for the agent. The environment is a time-series repository of vehicles' mobility data and contains a large population of periodic beacon messages with attack labels. The *state* contains two sequences: the sequence of previous actions and the current vehicle's time-series data. According to the state design, the next action taken by the agent is dependent on the previous actions and the current vehicle's information. The action space is defined as $A = \{0,1\}$, where 1 indicates the detection of an attack and 0 represents the normal behaviour. In a given state s, the agent selects the action as

$$a = arg \max_{a} Q(s, a).$$

The *reward* r is offered to the agent when an action a is taken in state s. In particular, the agent is given a positive reward for correctly identifying the misbehaviour, i.e., True Positive (TP), or a normal state, i.e., True Negative (TN); otherwise, a negative reward is given to the agent for incorrect identification of a normal state as an attack, i.e., False Positive (FP), or an attack as a normal state, i.e., False Negative (FN). In safety-critical V2X scenarios, FNs are more hazardous than FP alarms; thus, an agent is penalized more for FN actions than for FPs. The reward function can be expressed as

$$r(s, a) = \begin{cases} A, if\ the\ action\ is\ a\ TP, \\ B, \quad if\ the\ action\ is\ a\ TN, \\ -C, if\ the\ action\ is\ a\ FP, \\ -D, if\ the\ action\ is\ a\ FN, \end{cases}$$

where A, B, C, D >0, with A>B and C<D. The effectiveness of our proposed RL-based approach applied for V2X misbehaviour detection is demonstrated in [24] by performing extensive experiments on specific parts of the open-source VeReMi dataset [27].

Current implementation

In this section, we evaluate the effectiveness of the RL-based approach applied for V2X misbehaviour detection by performing experiments using the VeReMi dataset. The VeReMi dataset includes 19 misbehaviour attack types and models two road traffic densities under each attack scenario: high-density (37.03 vehicles per square km) and low-density (16.36 vehicles per square km). For each attack scenario, a log file per vehicle is generated, which contains information transmitted by neighbouring vehicles over its entire trajectory. Each scenario contains a ground truth file to record the observed behaviour of all participating vehicles. The exchanged messages constitute a three-dimensional vector for position, speed, acceleration and heading angle features. The proportion between misbehaving and genuine vehicles is set to 30% - 70%, respectively, for all the simulations. Table 1 depicts the results of the RL-based detection per attack type.

Type	Attack	Accuracy	Precision	Recall	F1-Score
1	Constant Position	0.9868	0.9588	0.9984	0.9782
2	Constant Position Offset	0.9981	0.9629	0.9982	0.9803
3	Random Position	0.9886	0.9642	0.9985	0.9810
4	Random Position Offset	0.9886	0.9632	1	0.9812
5	Constant Speed	0.9988	0.9968	0.9995	0.9982
6	Constant Speed Offset	0.9923	0.9766	0.9978	0.9871
7	Random Speed	0.9985	0.9987	0.9963	0.9975
8	Random Speed Offset	0.9915	0.9774	0.9945	0.9858

9	Sudden Stop	0.9811	0.9274	1	0.9623
10	Disruptive	0.9896	0.9664	1	0.9829
11	Data Replay	0.9894	0.9656	0.9999	0.9825
12	Delayed Messages	0.9666	0.9012	0.9976	0.9470
13	DoS	0.9999	0.9999	1	0.9999
14	DoS Random	0.9997	0.9996	1	0.9998
15	DoS Disruptive	0.9991	0.9984	1	0.9992
16	Traffic Congestion Sybil	1	1	1	1
17	Data Replay Sybil	0.9938	0.9981	0.9809	0.9894
18	DoS Random Sybil	1	1	1	1
19	DoS Disruptive Sybil	0.9972	0.9998	0.9940	0.9970

Table 1: Detection performance per attack type

3.1.3 Anomaly detection in 5G networks

3.1.3.1 Security Analytics Framework

Problem and challenges

The rapid advancements of digital technologies have resulted in considerable increase of new technologies, such as SDN/NFV, network slicing, multi-tenancy, edge and cloud computing. These advancements have also led to an increased attack surface that constitutes a rich source of vulnerabilities that can be exploited by malicious threat actors.

Anomaly detection is a significant branch of network security that detects abnormal traffic flows in the presence of an attack in a network. It relies on the assumption that normal data follow a particular and unknown distribution, whereas anomalies originate from different (also unknown) distributions. In traditional security management, intrusion detection takes place after several weeks that the incident was initiated, causing severe damage in the meantime. ML is a promising technique that can address some of the challenges concerning the analysis of encrypted traffic and detection of unknown attacks (e.g., zero-day attacks and advanced persistent threats). ML can help to automatically and efficiently detect anomalies that are not detected by traditional intrusion detection techniques (e.g., Deep Packet Inspection (DPI) and signature-based techniques).

State of the art analysis

Deep Learning (DL) is a ML branch that relies on neural networks that can be used for detecting such anomalies. This section provides an insight on the current State-of-the-Art. Neural networks have been used successfully in several fields, such as computer vision, medical diagnosis, fraud detection and network intrusion detection.

Neural networks are composed of an input layer, multiple hidden layers and an output layer. Neural networks can have different structures of their hidden layers, such as fully connected layers, convolutional layers, pooled layers and many others. Use of these different components give rise to different neural networks architectures, such as MLP, CNN and LSTM networks.

In the literature, all these methods have been applied successfully in network intrusion detection management, but they are applied to solve very specific problems and rely on public datasets for training and tests. Important challenges remain to be able to obtain a more generic solution and deal with real network traffic that is constantly changing (i.e., less predictable). In an article published in 2018 [28], the authors describe the use of sequential LSTM autoencoders on detecting intrusions found

on the ISCX IDS 2021 dataset [29]. Their experiments demonstrate that LSTM autoencoder with mean or max pooling achieve the highest F1-Score metric (0.8538) compared to other unsupervised solutions, such as LSTM, Gated Recurrent Unit (GRU) and Bi-LSTM. F1-Score is a formula that measures the model's accuracy on a dataset. In [30], the authors use improved CNNs for detecting intrusions in the KDD99 dataset [31], demonstrating higher accuracy (99.23%) than Support Vector Machines (SVM) and Deep Belief Network (DBN) algorithms. In [32], the authors describe a revised version of the LeNet-5 model for network threat classification (originally designed by LeCun et al. in 1998), achieving higher accuracy (97.53%) compared to the existing LeNet-5 model (95%) and malicious code identification in other deep learning schemes. In [33] the authors use the CIDDS-001 dataset[34] to evaluate the performance of an LSTM model, achieving higher accuracy (0.8483), recall (0.8834) and precision (0.8514) compared to other models, such asIe Bayes, SVM and MLP. The classification accuracy, precision, recall and F1-Score are the most popular metrics used for evaluating a neural network model.

Solutions descriptions and advancements

AI and ML contributions: In the course of INSPIRE-5Gplus project, there are different enablers for conducting anomaly detection on network traffic: i) the MMT Security Monitoring Framework [93] and ii) the Security Analytics Framework. Both enablers will be integrated on a 5G Standalone environment, comprising cloud and edge computing nodes that will provide the source data for training the models. In the "features" definition phase, we compare features used in known intrusion detection datasets and define the features that we are going to use in the training phase.

MMT Security Monitoring Framework: Regarding the model architecture, we are conducting an analysis of existing neural network models in the literature in order to design an appropriate architecture that provides an acceptable trade-off between time complexity and performance. The AI discipline offers variety of structures and we particularly focus on DL since the use of multiple layers to progressively extract higher-level features from the raw input is well adapted to network traffic analysis. The choice of a particular structure to solve real-world problems is crucial in order to obtain precise and efficient results. In the scope of intrusion detection, there are several widely popular models. The DBN is a model created by stacking multiple Restricted Boltzman Machines. While DBNs are fast to train in an unsupervised way, they present limitations in the training due to the use of approximated gradients [35]. Another example of DL technique is the Deep Autoencoder (or Stacked Autoencoder, SAE). It is an unsupervised method with an encoder-decoder structure and a hidden space that allows obtaining a better representation of latent features and an overall dimensionality reduction of the input. However, as SAEs are made in order to reconstruct the input, in the case of intrusion detection they can only be modelling normal traffic, as it is shown in several other works [36]. This means providing a big training dataset containing only normal traffic, which in many cases is not a realistic possibility. CNN is yet another DL method containing a multi-layered feed-forward neural network (convolutional layers) capable of learning hierarchical features. Whereas the structure of the model tends to be more complicated than other techniques, it generally tends to give highly accurate results. Recent works [37] [38] have shown the advantage of combining different techniques to create hybrid models. In the scope of all the above, we strive to adopt a combination approach that uses SAEs and CNN, similar to what is found in the literature [38]. Thanks to such approach, we utilize the latent feature discovery and dimensionality reduction of SAEs, with the good performance of CNN.

Security Analytics Framework: While initially, the Security Analytics Framework (SAF) was based upon the open-source project Apache Spot, due to its inactivity, we decided to begin from scratch an updated enabler for anomaly detection (in the form of an anomaly detection framework), based on Tensorflow, Keras and Tensorflow Servings. These solutions are more manageable from deployment perspective. As the core component of the SAF, we have selected a deep learning-based model, considering two facts: i) deep architecture models are able to capture the underlying data distribution more effectively than swallow models, and ii) they are able to learn useful representations in high-dimensional data. To this end, autoencoders are suitable for identifying outliers in the network traffic, that could be generated due to a potential attack. In intrusion detection, autoencoders are trained using normal traffic, so they are able to flag a network flow as suspicious, since the characteristics of

the network flow lead to exceeding the specified thresholds. The threshold is specified by the infrastructure owner or the service provider and it can take the form of a decision criterion based on two metrics: i) reconstruction error, and ii) energy-based score. Whenever a sample is above/below a selected value/threshold, then it is flagged as an outlier. The reconstruction error is based on the mean absolute error or the mean square error. In our case, our fully-connected autoencoder has been trained on the NSL-KDD dataset, achieving a performance of ~73%. However, since this is still lower than what has been reported in the literature by similar approaches, we continue processing the architecture, in order to increase performance. The size of the latent layer seems to be important in the overall performance.

3.1.3.2 Advanced Encrypted Traffic Analysis

Problem and challenges

There are various challenges that need to be addressed for analysing encrypted traffic, e.g., traffic clustering, classification, anomaly detection, etc. [39] With respect to both traffic classification and anomaly detection, there is a particular need to perform malware and attack detection and identification. The growing popularity of traffic encryption increases user security at the individual level, but at the same time it has become a big challenge for traffic analysis, imposing the need to explore improved analysis techniques based on other criteria, such as behaviour analysis.

Compared to ordinary traffic, most of the information within encrypted traffic is meaningless, as it is the payload that is encrypted (e.g., TLS at OSI layer 4) or even the header is encrypted (e.g., IPsec at OSI layer 3). It thus implies that there are significantly less features that a tool can gather when using DPI techniques. With the introduction of network encryption techniques, such as the TLS protocol, the accuracy and efficiency of conventional Network Intrusion Detection Systems (NIDS) that were using rule and signature-based monitoring detection methods is greatly reduced, in some cases rendering them completely useless. Moreover, the variety and dynamicity of the traffic malware poses a significant challenge on traffic monitoring tools in terms of flexibility and generalization of their algorithms. New encrypted traffic analysis tools need to be adaptable not only to ever evolving attacks, but also to new protocols, policies and potentially to new breakthrough technologies that are constantly being developed. It is also important to note that encrypted traffic analysis, similarly to non-encrypted one, is most useful when done in an on-line, real-time manner. Independent of the particular use-case, most applications and security functions require the ability to directly react to what is happening in the network on the fly, especially when it involves the detection of malware. Finally, one of the most obvious but equally important challenges of encrypted traffic analysis is the precision of the results. While high accuracy of the analysis is expected, simultaneously the system cannot be prone to high number of false positives either.

State of the art analysis

In order to tackle the aforementioned challenges, numerous different approaches for executing analysis of encrypted traffic have emerged during the last years.

A common way to deal with encrypted communications is to use a decryption platform, which decrypts the traffic in real time and inspects its contents. For instance, in case of HTTPS malware, these are interceptor proxies (e.g., TLS interception). Such a strategy however has multiple drawbacks. Firstly, it leads to a drop in network performance, increasing the latency. Moreover, it involves high computational complexity and constant updates of certificates and protocol keys, while simultaneously having a potential of violating different policy standards, such as privacy.

Another rapidly developing field used for encrypted traffic analysis is AI. While many ML methods give decent results, they are not without some limitations. Depending on the particular ML algorithm, the mode of learning can be divided into three categories: supervised, semi-supervised, and unsupervised. The first two classes, where we can find methods such as Hidden Markov Models, RIPPER, AdaBoost, SVMs, C4.I Naive Bayes [40] for supervised group, and k-means or k-nearest neighbour for semi-supervised [41], are relying on hand-crafted traffic features and involve the need for voluminous

labelled datasets. This can be time consuming, costly or even impossible in some cases. Moreover, many of the important characteristics that can imply malicious traffic that are known to experts at a given time can quickly become outdated. The combination of labelled dataset requirements with the reality of rapid and constant growth of completely new attacks, as well as class imbalance problems for the dataset can significantly affect the final ML algorithm accuracy. Concerning the above-mentioned ML methods categories, only unsupervised algorithms, such as Principal Component Analysis (PCA) do not require labelled datasets. However, in the case of these methods, accuracy is still not high enough, and the features still need to be hand-crafted. That being said, recently developed DL methods are especially gaining in popularity in the field of encrypted traffic analysis. DL methods do not require hand-crafted features, which makes them possibly much more appropriate for the analysis of encrypted traffic. One of the big advantages of using such techniques is that they are able to work on the dataset without feat–re mining - it is the algorithm itself that "chooses" the most important features, as well as latent features, directly from input data. DL methods used for encrypted traffic analysis apply various models [42][43], from MLP, CNNs, Recurrent Neural Networks (RNNs), to Stacked and Variational Autoencoders. Such models are quite flexible, thanks to the fact that they are able to work on packet-level features, flow features, or raw data. Moreover, some of the DL techniques (e.g., Autoencoders), instead of trying to learn and categorize normal and malicious traffic, a technique that can quickly become obsolete, they aim at modelling only the normal traffic. As they are able to reconstruct very adequately the normal traffic (with a low error rate between the input sample and reconstructed one). In this way, we can set a threshold of error rate of the reconstructed sample and consider any sample that exceeds it as possibly anomalous. This makes these techniques particularly flexible and adaptable for detecting so-called zero-day attacks. On the other hand, DL techniques are known to be both computationally challenging and require a large volume of data in order to achieve decent accuracy of the results.

Solutions descriptions and advancements

AI and ML contributions: The solution that is being developed in INSPIRE-5Gplus is based on the integration of numeric feature extraction techniques, the MMT-Probe (part of the MMT Security Monitoring Framework) for data capture and analysis, and algorithms based on a combination of DL methods. Currently, we have created a DL model that consists of combining SAEs and CNNs in order to examine and classify encrypted traffic.

In our case, we experiment with different configurations of Autoencoders utilizing different hidden layer numbers, namely: **i)** symmetric Deep Autoencoder with 3 layers; and; **ii)** symmetric Deep Autoencoder with 5 layers.

Regardless, in both cases, the use of the feature extraction and translation done by the MMT-Probe is the same. The role of the MMT-Probe is to capture and translate raw traffic (online or offline) to calculate the statistic features in the form of numeric values. Our system focuses on gathering traffic packets into flows, i.e., a flow sequence of packets from a source entity to a destination and vice-versa. Using flows, we then calculate different statistics (e.g., mean or standard deviation of flow lengths, times). These statistics will vary depending on the particular protocol (e.g., TLS) that are being used. The full set of features is then fed in the form of a numeric matrix input into the DL component. Currently, we are using a total of 60 features as input to the DL module. The general procedure followed is shown in Figure 1. The encrypted traffic is processed by the MMT-Probe and the results are fed to the DL Module that will provide a verdict to the Application (e.g., the security management function).

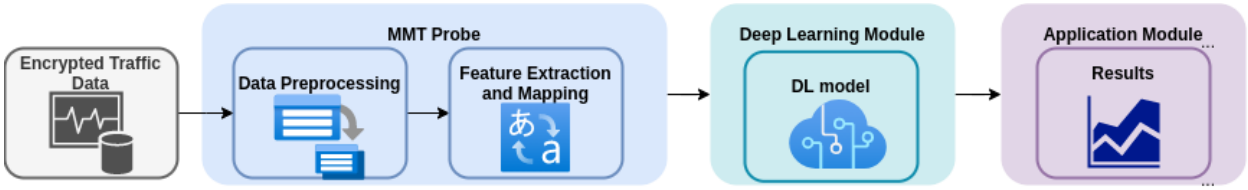

Figure 1: Deep Learning processing schema

In general, regardless of the particular utilisation of the DL module (which is described below), the module has two modes of working: training and evaluating modes.

In the training mode, the DL learning algorithm is executed in order to create the desired model. The input data is used to learn the weights that need to be applied based on a back propagation technique. The model is tuned and optimised, before it is finally validated on not yet analysed data. Once the desired accuracy is reached, the model is ready to be used.

In the evaluation mode, the DL model is used in order to generate its evaluation of new data flows. The final outcome of the solution is the evaluation of a flow being malicious or normal.

In the current stage, we are utilising a structure containing two parallel Autoencoders and a one-dimensional CNN that is used sequentially after the SAEs. The autoencoders are used in order to transform normal (Autoencoder 1) and malicious (Autoencoder 2) traffic. Here, the idea is to use only the encoding parts of the Autoencoder. By using the encoding, the model attempts to compress the input information into a reduced dimensional space. Thanks to this, we obtain a compact version of the input. Then, the output of both autoencoders is concatenated and fed into 1d CNN module that is learning the class representation (using the compressed inputs) and, hence, the performance of this structure is significantly improved. Figure 2 shows the structure of the used model.

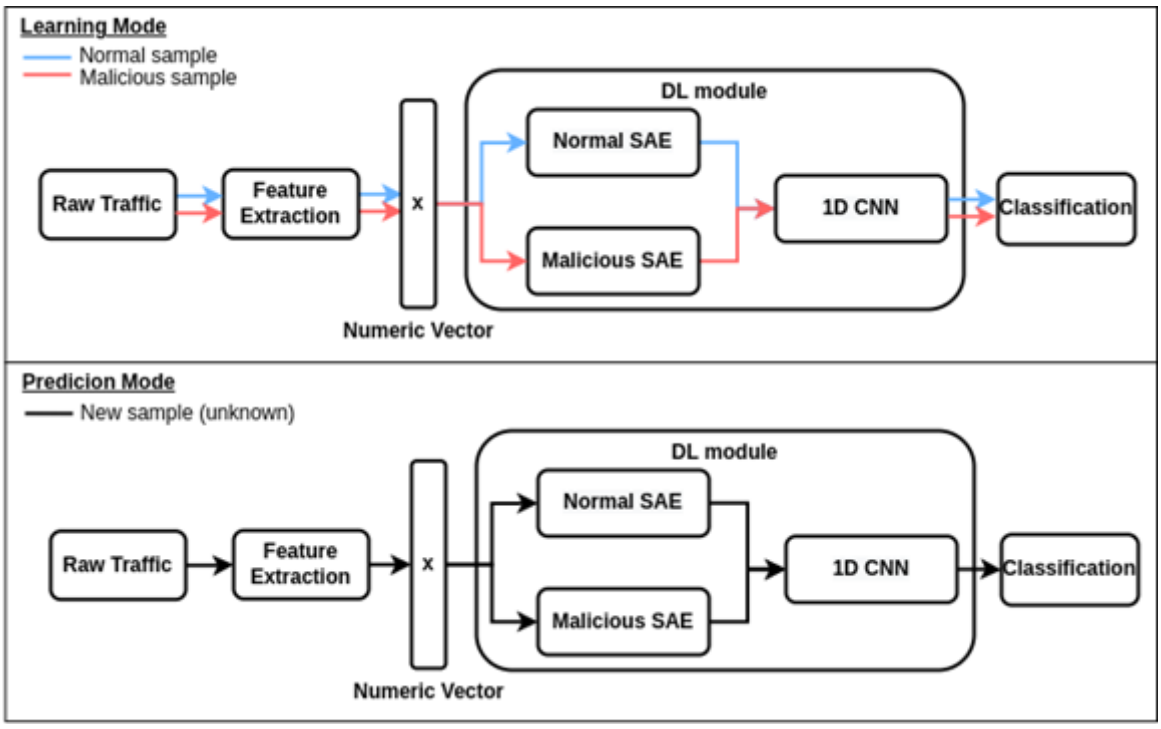

Figure 2: Structure of the used DL model

In our model, the input layer takes as input the single matrix of 60 features. Both SAEs use dense hidden layers, meaning that they are deeply connected with a connection between each neuron from the preceding layer to the current one. Between the dense layers, the dropout layers are included in order to prevent the overfitting of the model. Each autoencoder is trained separately, respectively using normal and malicious traffic. The output of each SAE is used as an input for one dimensional CNN, whose structure is shown in Figure 3. For the CNN, we use two convolutional layers (Conv1D in

the figure) followed by max pool layer (MaxPooling1D) and a dropout. Such a set is stacked three times and followed by a flattening layer (Flatten) and two dense connected layers (Dense). This last layer provides the predicted classification of the sample. In case of SAE, the mean squared error loss function is used. CNN part is using binary cross entropy.

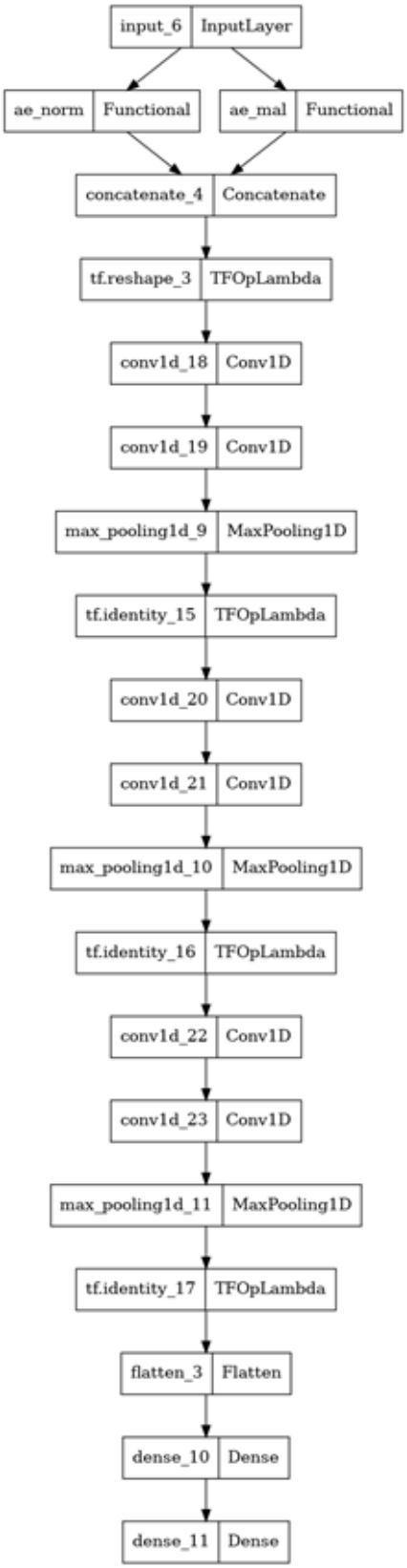

Figure 3: Structure of used one dimensional Convolutional Neural Network (1D CNN)

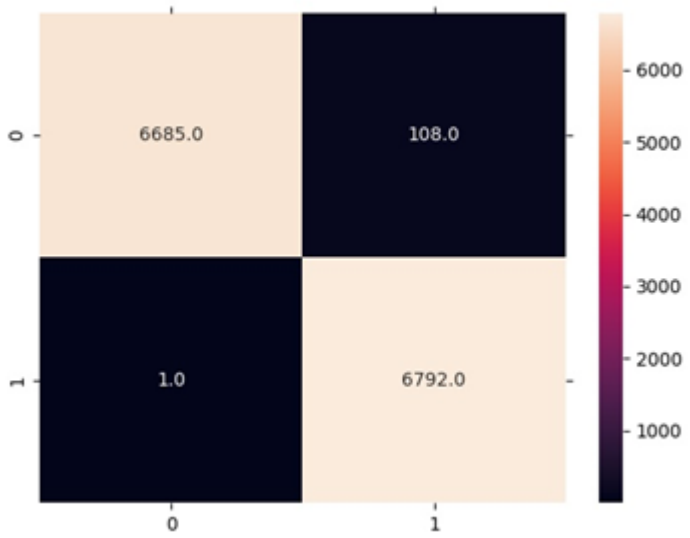

Figure 4: Confusion matrix with the results of classification – X axis shows predicted samples and Y axis signifies ground truth samples, where 0 signifies normal class, and 1 represents the malicious class

Currently, the performed experiments give very good results for the classification of malicious traffic utilizing Ares, a Python based botnet as an example. Table 2 shows the metrics of the trained model. The model was tested using a balanced set containing 13586 samples where half was normal traffic, and half contained infected bot traffic. Figure 4 summarizes the classification results obtained comparing the one predicted by the model and the ground truth.

Metric	Value for Normal Class (%)	Value for Malicious Class (%)
Precision	1	0.98
Recall	0.98	1
F1	0.99	0.99
Metric	Value	
Accuracy	0.99	

Table 2: Metrics obtained by the trained model

3.1.4 Multi-domain/tenant DDoS detection and mitigation

3.1.4.1 Problem and challenges

The advent of 5G and its many advances over previous mobile technologies - much lower latency, huge bandwidth, the possibility to connect many more devices per square meter, and so on an– so forth - will not just bring benefits. It turns out that all these advances in mobile network performance will provide the perfect breeding ground for attacks. DoS attacks, in particular, will benefit the most from this: larger bandwidth will allow much more traffic to be sent per device, and the fact that many more devices can be concurrently connected to the network (proliferation of IoT devices) will allow much larger, and much more powerful botnets to be created in order to carry out these types of attacks much more effectively, especially empowering DDoS attacks.

The main challenge that arises from the previous aspects is an effective detection for traditional DDoS attacks (e.g., flooding attacks) and also for more advanced stealthy DDoS attacks (e.g., SlowDoS attacks). For this purpose, we aim to leverage AI techniques, particularly Deep Learning techniques, for an efficient detection and mitigation of such attacks in 5G environments.

3.1.4.2 Problem and challenges

Many research efforts have been devoted to tackle DDoS attacks leveraging ML and/or SDN. Braga et al. [44] proposed an intelligent method for detecting network-layer DDoS attacks in an SDN environment. The proposed method uses a Self-Organizing Maps (SOM) [45] model, an unsupervised artificial neural network, trained on traffic flow features. The contribution in [46] rely on DNN models to detect intrusion in an SDN network. The authors in [47] devised a ML-based collaborative DDoS mitigation strategy in a multi-SDN controller environment. The detection is performed Naive Bayes classifier based on flow features extracted by the SDN controller. Upon detection of malicious behaviour, the SDN controller in the attacker's network is automatically notified to create a deny IP based flow. Similar to [40], the work in [42] , [43] consider only network-layer attacks. Moreover, the proposed models are trained on NSL-KDD, a relatively old dataset that cannot reflect the current trend in network attacks.

Hong et al.[48] devised an SDN-assisted defence method to detect and mitigate slow HTTP DDoS attacks. The defence solution is deployed as a SDN application and triggered by the web server when the number of open connections that sent incomplete HTTP requests exceeds a given threshold. The major weakness of threshold-based schemes is their lack of accuracy. In fact, threshold-based schemes are unsuitable for detecting application-layer DDoS attacks due to the resemblance between the traffic patterns generated by those attacks and benign activities. The authors in [49] demonstrated the potential of ML techniques in detecting low-rate application-layer DDoS using the characteristics of malicious TCP flows. A detection accuracy of over 97% has been achieved using K-Nearest Neighbour, Decision Trees and DNN techniques.

Some solutions related to the detection of DDoS attacks over 5G multi-tenant networks have been presented in recent years. For instance, Mamolar *et. al* [50] proposed an extension of the well-known Intrusion Detection System (IDS) Snort, capable of detecting DDoS attacks in real time, to support 5G multi-tenant traffic, so it can be deployed in a multi-tenant 5G environment. However, they do not leverage any AI technique, so we consider this approach too static and inappropriate for such dynamic network environments as those found in 5G.

Furthermore, very few contributions have focused on addressing the issue in 5G network slicing environment leveraging mainly the resource isolation concept (e.g., [51]). However, the new shift towards cloud-native architecture where virtual network functions are deployed as containers makes the complete isolation hard to achieve.

In addition, detecting DDoS attacks by only analysing the network traffic may not always be possible, especially with the emergence of stealthy application-layer DDoS attacks which aim at exhausting the server's resources while generating a traffic that mimic the legitimate one. Thus, using new sources of information, such as resource usage and/or performance of service under attack, is vital to discriminate malicious behaviour due to DDoS attack.

3.1.4.3 Solutions descriptions and advancements

To fill the aforementioned gaps, we leverage the potential of DL to build three solutions for tackling the stealthy DDoS attacks in a 5G environment, considering two settings (i.e., network slicing and multi-domain/multi-tenancy) and performing malicious behaviour detection either by using data collected from network layer (i.e., characteristics of network flows) or application layer (i.e., resource usage and service performance metrics).

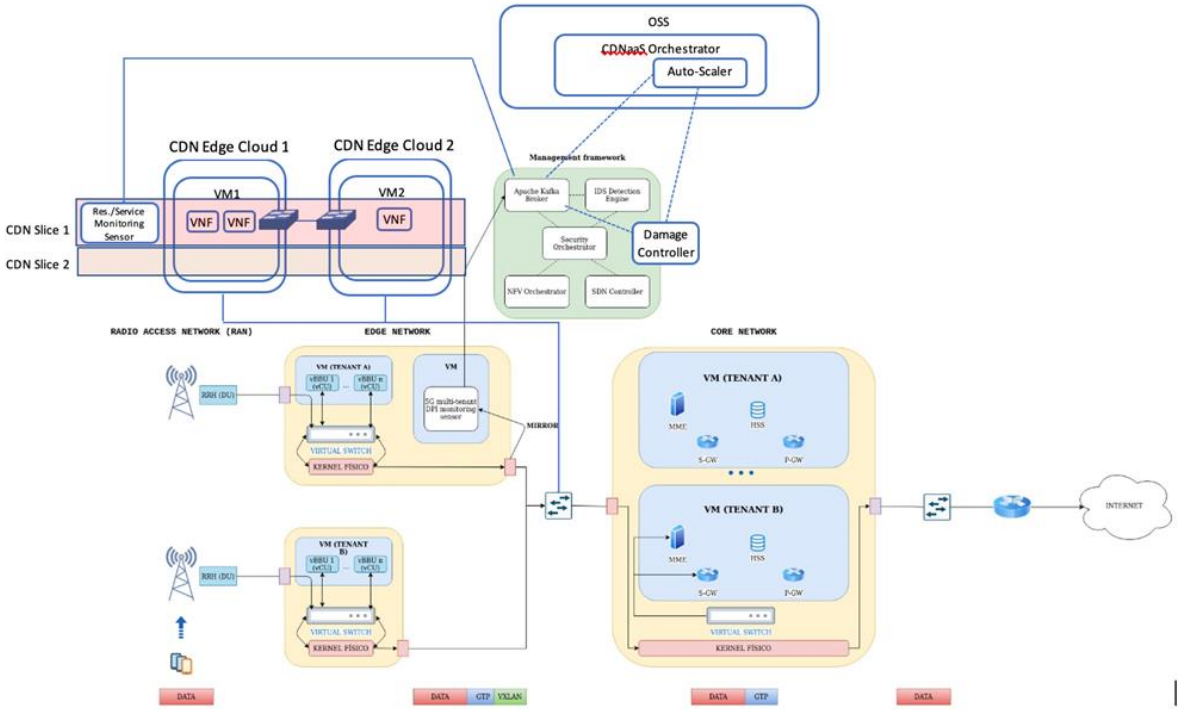

Figure 5: DDoS detection and mitigation architecture

The overall architecture of our enabled is shown in Figure 5. As can be noted, we consider a multi-tenant 5G scenario with double encapsulated traffic using Virtual Extensible LAN (VXLAN) and GPRS Tunnelling Protocol v2 (GTPv2) protocols. We also consider Content Delivery Network (CDN) slices deployed at the edge cloud infrastructure, potentially across multiple domains. The slices provide video CDN functionality as a service (CDNaaS). Each slice is composed of a set of VNF instances (e.g., streaming servers, caches, and transcoders) appropriately chained together to form a service instance. The isolation between network slices is supported using VXLAN over SDN.

Although we present a unique enabler from the design perspective, two different 5G environment settings have been deployed. On one hand, AALTO/OULU's implementation will be able to operate over a network slicing environment and, on the other hand, UMU's implementation focuses on a multi-tenant/multi-domain network environment. Furthermore, AALTO/OULU considers the detection of malicious behaviour caused by application-layer DDoS attacks by using either data collected from network layer or application layer. This results into three different implementations for the same asset.

UMU's implementation

Regarding the UMU part, the overall implementation can be divided into three main modules/layers that communicate with each other using Apache Kafka. Below, each layer is briefly described by its capabilities:

- Real-time monitoring agent: capable of dissecting multi-tenant 5G packets encapsulated with VXLAN and GTPv2 protocols.
- Real-time conversations processor: groups packets into conversations/flows. On one hand, it generates a feature-based dataset composed of 57 features per conversation (offline). On the other hand, it feeds the ML processor with real-time conversations data (online). In order to generate the dataset, we leverage the Apache JMeter tool [52] for benign traffic simulation, and the slow-http-test tool [53] for application layer DDoS attack traffic simulation.
- Real-time ML processor: based on unsupervised learning techniques, particularly Gaussian Mixture Models (GMM) (clustering) and Autoencoders (deep learning), this layer is able to use the conversations information to effectively detect potential attacks with high detection accuracy (~98%).

This architecture can be consulted in detail in Figure 6.

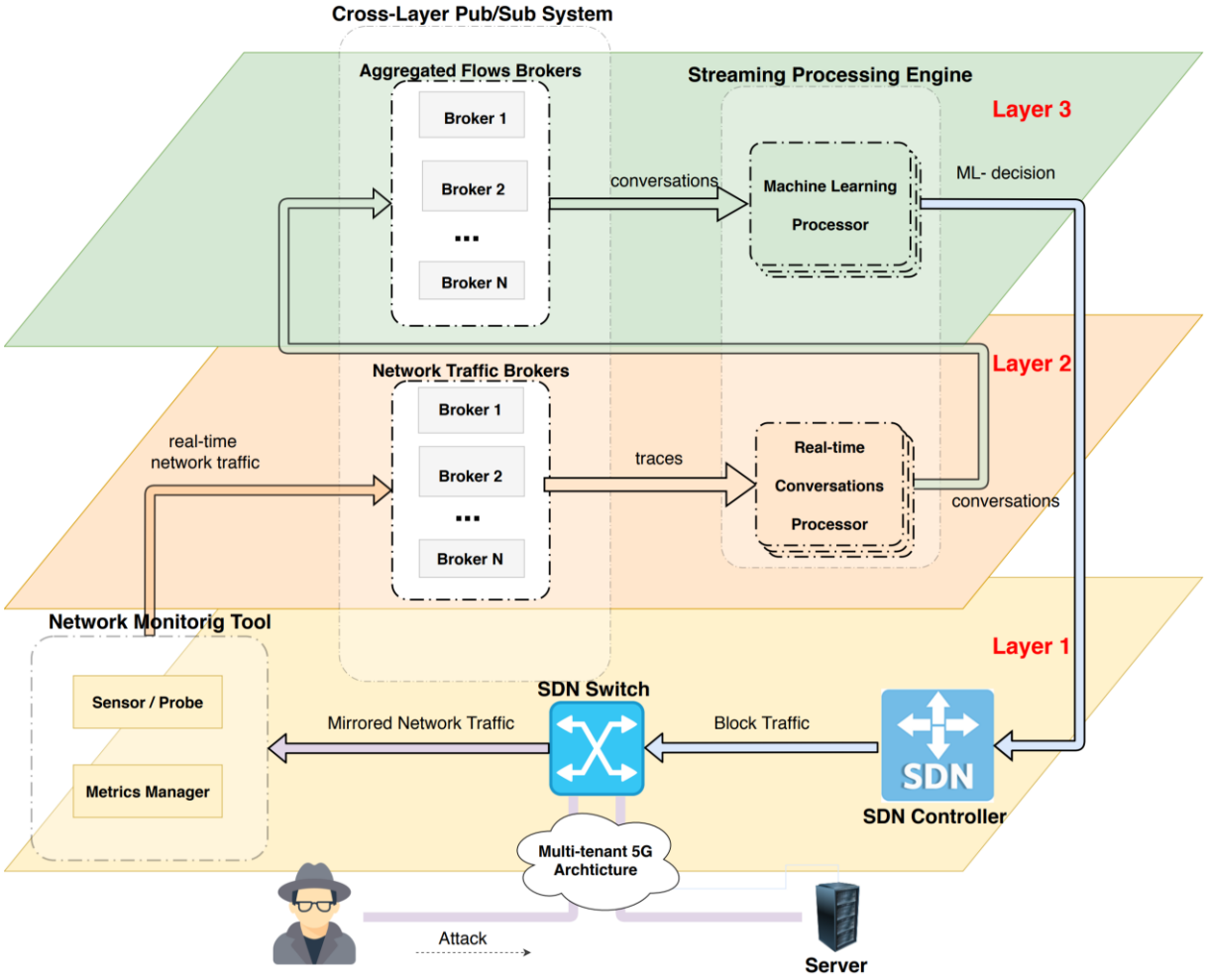

Figure 6: UMU's multi-tenant/multi-domain DDoS detection architecture

In order to enable the communication between modules, each one generates a JavaScript Object Notation (JSON) message with a specific format that will be uploaded to the Kafka broker. Below, each one of these can be seen in Figure 7, Figure 8, and Figure 9:

```
{
  "data":[
    {
      "flowId":"26147695",
      "encapsulationLayer":"0",
      "macSrc":"8:0:27:d3:fc:65",
      "macDst":"8:0:27:3c:70:b1",
      "srcIP":"10.94.94.12",
      "dstIP":"10.94.94.11",
      "l4Proto":"17",
      "srcPort":"45886",
      "dstPort":"4789",
      "l7Proto":"vxlan"
    },
    {
      "flowId":"10013632",
      "encapsulationLayer":"1",
      "encapsulationID1":"0",
      "encapsulationType1":"vxlan",
      "macSrc":"42:c:18:ff:a8:43",
      "macDst":"9a:7d:5:44:a5:4e",
      "l3Proto":"2048",
      "srcIP":"10.0.3.2",
      "dstIP":"172.0.0.1",
      "outSrcIP":"10.94.94.12",
      "outDstIP":"10.94.94.11",
      "l4Proto":"17",
      "srcPort":"2152",
      "dstPort":"2152",
      "l7Proto":"gtp-u"
    },
    {
      "flowId":"42659330",
      "encapsulationLayer":"2",
      "encapsulationID2":"200",
      "encapsulationType2":"gtp",
      "srcIP":"172.99.0.2",
      "dstIP":"172.99.0.1",
      "outSrcIP":"10.94.94.12",
      "outDstIP":"10.94.94.11",
      "l4Proto":"6",
      "srcPort":"443",
      "dstPort":"49526",
      "l7Proto":"https",
      "timestamp":"1588532009.512552",
      "ipTotalLength":"66",
      "ipTTL":"64",
      "tcpWindowSize":"350",
      "sslContentType":"chgcipher",
      "tcpFin:":"0",
      "tcpSyn:":"0",
      "tcpRst:":"0",
      "tcpPsh:":"1",
      "tcpAck:":"1",
      "tcpUrg:":"0"
    }
  ]
}
```


Figure 7: Monitoring module output (per packet)

```
{
  "data":[
    {
      "flowId":"35284409",
      "encapsulationLayer":"0",
      "macSrc":"8:0:27:3c:70:b1",
      "macDst":"8:0:27:d3:fc:65",
      "srcIP":"10.94.94.11",
      "dstIP":"10.94.94.12",
      "l4Proto":"17",
      "srcPort":"57747",
      "dstPort":"4789",
      "l7Proto":"vxlan"
    },
    {
      "flowId":"9960368",
      "encapsulationLayer":"1",
      "encapsulationID1":"0",
      "encapsulationType1":"vxlan",
      "macSrc":"9a:7d:5:44:a5:4e",
      "macDst":"42:c:18:ff:a8:43",
      "l3Proto":"2048",
      "srcIP":"10.0.2.1",
      "dstIP":"172.0.0.2",
      "outSrcIP":"10.94.94.11",
      "outDstIP":"10.94.94.12",
      "l4Proto":"17",
      "srcPort":"2152",
      "dstPort":"2152",
      "l7Proto":"gtp-u"
    },
    {
      "encapsulationLaye":"2",
      "encapsulationID2":"100",
      "encapsulationType2":"gtp",
      "uplinkIP":"172.99.0.1",
      "downlinkIP":"172.99.0.2",
      "uplinkPort":"50474",
      "downlinkPort":"443",
      "outSrcIP":"10.94.94.11",
      "outDstIP":"10.94.94.12",
      "l4Proto":"6",
      "l7Proto":"https"
    },
    {
      "features":"172.99.0.1,172.99.0.2,50474,443,2006920,2.5,1.5,290.5,80.0,357.0,52.0,116.2,
      60.0,40.0,53.333333333333336,43690.0,342.0,17681.2,43690.0,0.0,29126.666666666664,
      64.0,64.0,64.0,64.0,64.0,64.0,0.0,0.4,0.0,0.2,0.6,0.0,0.0,0.0,0.6666666666666666,
      0.3333333333333333,0.0,0.6666666666666666,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
      0.0,0.0,601.0,601.0,1.0,0.0,601.0,601.0,1.0,0.0"
    }
  ]
}
```


Figure 8: Conversations processor module output (per conversation)

```
{
   "data":[
      {
         "flowId":"35284409",
         "encapsulationLayer":"0",
         "macSrc":"8:0:27:3c:70:b1",
         "macDst":"8:0:27:d3:fc:65",
         "srcIP":"10.94.94.11",
         "dstIP":"10.94.94.12",
         "l4Proto":"17",
         "srcPort":"57747",
         "dstPort":"4789",
         "l7Proto":"vxlan"
      },
      {
         "flowId":"9960368",
         "encapsulationLayer":"1",
         "encapsulationID1":"0",
         "encapsulationType1":"vxlan",
         "macSrc":"9a:7d:5:44:a5:4e",
         "macDst":"42:c:18:ff:a8:43",
         "l3Proto":"2048",
         "srcIP":"10.0.2.1",
         "dstIP":"172.0.0.2",
         "outSrcIP":"10.94.94.11",
         "outDstIP":"10.94.94.12",
         "l4Proto":"17",
         "srcPort":"2152",
         "dstPort":"2152",
         "l7Proto":"gtp-u"
      },
      {
         "encapsulationLayer":"2",
         "encapsulationID2":"100",
         "encapsulationType2":"gtp",
         "uplinkIP":"172.99.0.1",
         "downlinkIP":"172.99.0.2",
         "uplinkPort":"50474",
         "downlinkPort":"443",
         "outSrcIP":"10.94.94.11",
         "outDstIP":"10.94.94.12",
         "l4Proto":"6",
         "l7Proto":"https"
      },
      {
         "attack":"true"
      }
   ]
}
```


Figure 9: ML processor module output (per conversation)

So as to achieve real-time communication, conversation processor and ML processor modules leverage Apache Spark Streaming library to receive and process all the incoming information continuously.

With reference to the implementation status of this part, UMU made available a first version of the previously described implementation. As future advancements, we plan to deploy the system on our own 5G testbed, implement real-time mitigation techniques based on SDN, and build a collaborative network of nodes to train the involved ML models using FL techniques, improving the overall privacy level while allowing clients to access to further data that is available in other devices.

AALTO/OULU's DDoS Detector Implementation

In this scenario, we consider the detection of application-layer DDoS attack within one slice instance. The attack targets the video streamer deployed as an NGINX web server. Figure 10 depicts the basic architectural components of the proposed solution to mitigate the application-layer DDoS attacks in a fully autonomous way. the "App-Layer DDoS Protection" component is in charge of detecting the malicious activity and issuing the security policy in case the attack is detected. It consists of four main modules communicating through REST APIs:

- Network Flow Collector which permanently collects network flows via port mirroring. To limit the impact of mirroring on the network performance, only traffic flowing from/to the monitored asset (e.g., Web server) is mirrored.
- Features Extractor analyses the collected traffic to retrieve flow's features relevant to application-layer DDoS attack detection. Once extracted, the flow features are passed to the Detector for uncovering suspicious behaviour.
- Detector relies on an DL model to detect the anomalous behaviour. If a malicious traffic pattern is identified, the Detector issues a security policy (e.g., flow dropping or steering) to the Security Orchestrator. The security policy is expressed using Medium-level Security Policy Language (MSPL) (see D3.2). Details on the proposed DL model will be provided in Section 3.1.4.3.
- Security Orchestrator which, upon receiving the security policy, converts the policy into a flow command (with the help of the Security Policy Manager) and sends it to the SDN controller. Based on the received flow command, a flow rule is pushed by the SDN controller to the corresponding virtual Switch (vSwitch) to fulfil the defined security policy.

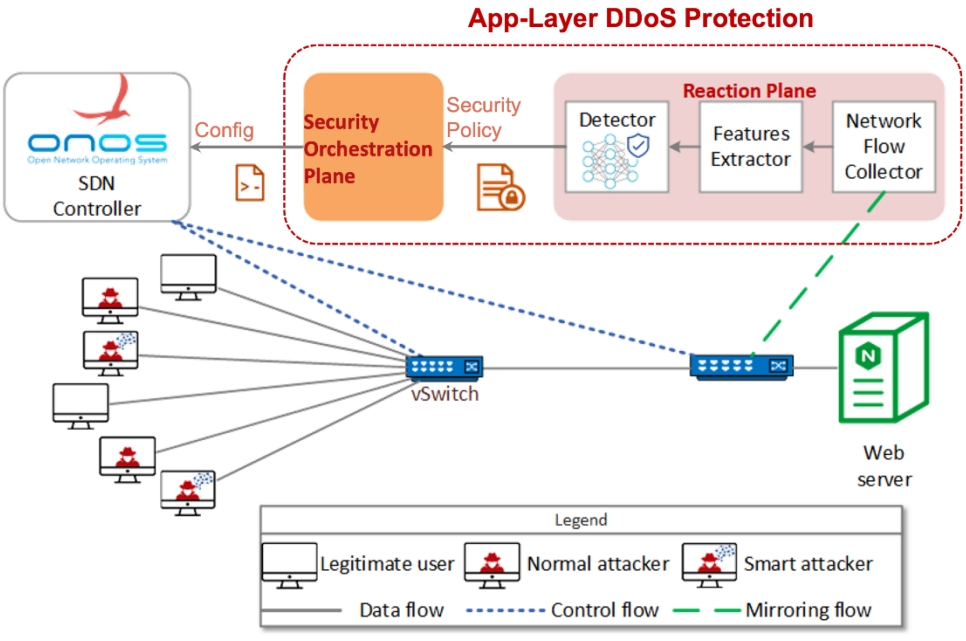

Figure 10: The App-Layer DDoS Self-Protection Framework's high- level architecture

The attacker has the capability to launch application-layer DDoS attacks, particularly HTTP-based flooding attack that aims to overwhelm the server by a voluminous number of legitimate HTTP requests. We consider both high-rate and slow-rate mode for launching the HTTP-based flooding attack. In high-rate mode, the attacker mimics a flash-crowd event by flooding the web server with a large number of legitimate HTTP requests in a short period of time. The low-rate mode, however, consists in establishing multiple HTTP connections with the web server by sending partial HTTP requests at a very slow rate.

AALTO/OULU's DDoS Mitigator Implementation

Unlike the previous scenario, we assume now that the detection of the application-layer DDoS attack may not be possible using the characteristics collected from network flows. This could be possible by generating new stealthier patterns of the DDoS traffic that has not yet been seen by the ML model integrated in the DDoS Detector. As illustrated in Figure 11, we assume also that the CDNaaS platform supports the auto-scaling functionality as a mean to deal with the workload that could be caused by DDoS attack if this last can escape detection using the DDoS Detector. However, the auto-scaling capability is a double-edged sword when the DDoS attack is underway. In fact, the auto-scaling may result in resource starvation while the shared infrastructure resources (e.g., CPU, RAM, etc.) between slices can be exhausted by the slice under attack, resulting in potential availability and performance of services provided by the co-hosted slices.

To address this problem, we propose a new solution that is able to discriminate legitimate auto-scaling requests due to flash events from malicious auto-scaling requests due to application-layer DDoS attacks. Figure 11 shows the basic architectural modules of the proposed solution. It consists of three main modules communicating through REST APIs:

- Monitoring Framework, which is in charge of collecting resource usage and performance metrics from the slices via deployed probes.
- Admission Controller Delegator, which is responsible for intercepting the auto-scaling request triggered by the auto-scaling module and delegate the scaling decision to the Damage Controller for validation.
- Damage Controller, which runs a DDoS Mitigator model that uses DL to detect whether the scaling is due to legitimate workload or rather malicious workload caused by an application-layer DDoS attack. If the workload is malicious, the scaling operation is refused. Details on the proposed DL model will be provided in the following part.

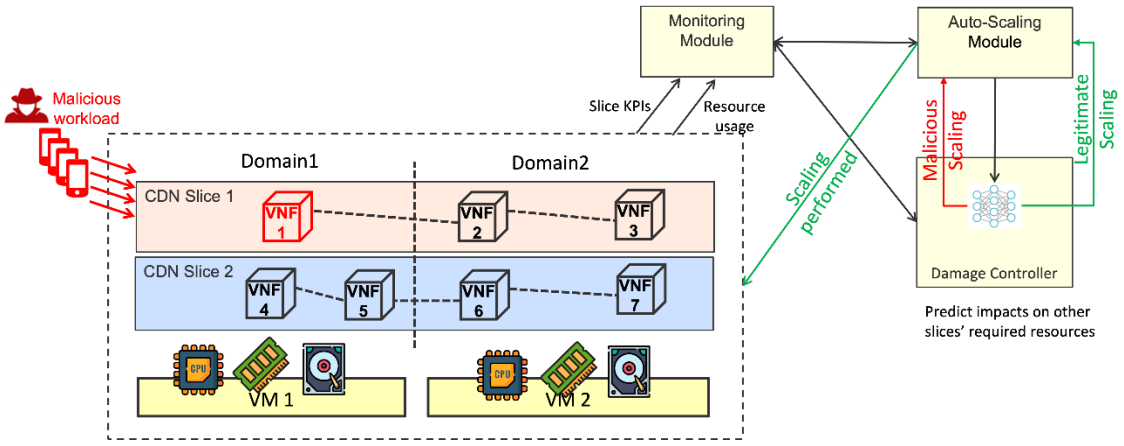

Figure 11: The application-layer DDoS Mitigator architecture

AI and ML contributions

UMU's DDoS Detection Model: With respect to the implementation proposed by UMU, our ML contributions can be found inside the proposed ML layer. In Figure 12, an overview of the proposed architecture regarding this module can be consulted. As can be noted, after an initial pre-processing of data, both presented techniques (GMM and autoencoder) are combined in order to classify conversations into attacks or normal behaviours. In a first stage, models are trained offline using the dataset previously generated by the *conversations processor*. Then, in a second stage, the system is able to operate in real-time (online) and to classify new conversations. The procedure to be followed when classifying a new instance is as follows: in first place, we evaluate the instance against the clustering algorithm (GMM). GMM-based clustering provides the probabilities of membership of each communication to the set of clusters, in such a way that those with a high value will be taken as normal

and those with a low value will be taken as abnormal. However, it is necessary to determine how those probabilities that take intermediate values (e.g., between 40% and 60%) are classified. For that reason, this clustering method requires the help of other DL techniques (in our case, an autoencoder) that allow us to break with the uncertainty of these ambiguous probabilities. Then, in those cases, we will use the autoencoder to provide the final decision for the conversation. On the other side, for evaluation purposes, we divide the original dataset into training and testing sets with a proportion of 80/20 (training/testing in %), replace infinite and invalid values with the median of the corresponding column, and normalize all values in order to bring them in the range [0 ... 1] (feature scaling).

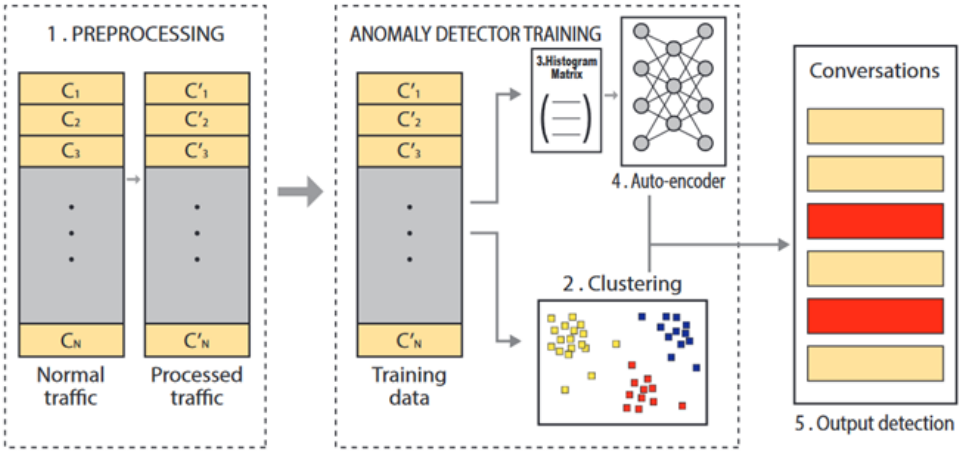

Figure 12: UMU's ML processor for DDoS detection

AALTO/OULU's DDoS Detector Model: To identify the application-layer DDoS attacks, DL is leveraged to build the detection model. The adoption of a DL model is motivated by its capacity of uncovering complex non-linear relationships between inputs and outputs, yielding higher accuracy in distinguishing application-layer DDoS flow patterns from legitimate flow patterns. Specifically, the detection model is built using an MLP algorithm. The proposed model consists of 1 input layer, 2 hidden layers with 64 neurons each, and a two-class *softmax* output layer. The model's input is the flow features received from the Extractor. The model's output is the traffic class; that is, DDoS traffic or legitimate traffic.

The MLP-based model integrated in the Detector module is trained on the recent intrusion detection dataset, CICIDS2017 [54] (where only network flows corresponding to legitimate traffic and DoS/DDoS attacks are used) augmented by normal and application-layer DDoS flows generated in our testbed [55]. The high-rate HTTP-based flooding attack is generated using Hulk tool while the low-rate HTTP-based flooding attack is launched using Slowloris tool. The DDoS attack against the video streamer has been performed using 8 LXD containers. Apache JMeter is used to simulate a legitimate client sending an HTTP request every one second. Each network flow in the dataset is defined by a feature vector containing 79 features in addition to a label identifying the flow's class (i.e., benign or malicious). The created dataset is available on [56]. 70% of the dataset's flows are used to train the model and the remaining 30% flows are used as a test set to assess the model's performance on unseen data. The model is trained for 10 epochs with a batch size of 128, Adam as an optimizer, and a learning rate of 0.001. The model is implemented using the Python's DL library Keras running on a TensorFlow backend. It achieved an accuracy of 99.65% on the test set.

AALTO/OULU's DDoS Mitigator Model: The DDoS Mitigator model is built as an anomaly detection model using the unsupervised learning technique LSTM Autoencoder on multivariate time series. The main motivation behind adopting a deep learning technique (i.e., LSTM Autoencoder) to solve the DDoS problem is its proven performance in capturing complex interactions between features of a large-amount of multi-dimensional data compared to traditional statistical-based methods (e.g., autoregressive models and exponential smoothing) [57]. Using multivariate time series allows to observe the correlation between different metrics, resulting in improved anomaly detection accuracy. Moreover, the use of unsupervised learning eliminates the need of labelled datasets which are usually

difficult to obtain in real-world setup. Figure 13 illustrates the model structure.

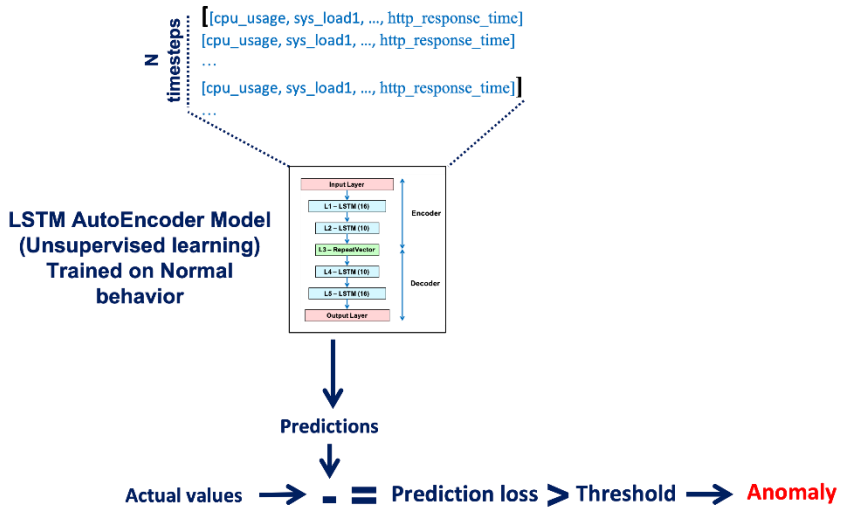

Figure 13: LSTM Autoencoder Structure

The inputs to the model are the features related to resource usage (e.g., CPU usage, system load, memory usage, I/O network traffic) and system performance (e.g., http response time).

To train the model, we generated a first dataset using two CDN slices created in our CDNaaS platform [58]. Figure 14 illustrates the testbed configuration. The VNFs of each slice are deployed as LXD containers on two Virtual Machines (VMs). 3 VMs are used to deploy the VNFs of the two slices where one VM is shared between them. Each VNF and VM has specific configuration in terms of CPU, RAM and Disk. A remote attacker performs application-layer DDoS attack against slice 1 by targeting the video streamer. The attacker launches the attack using Hulk and Slowloris tools at specific time periods. The resource usage and system performance at the VM level and VNF level for both slices are monitored using Prometheus [58].

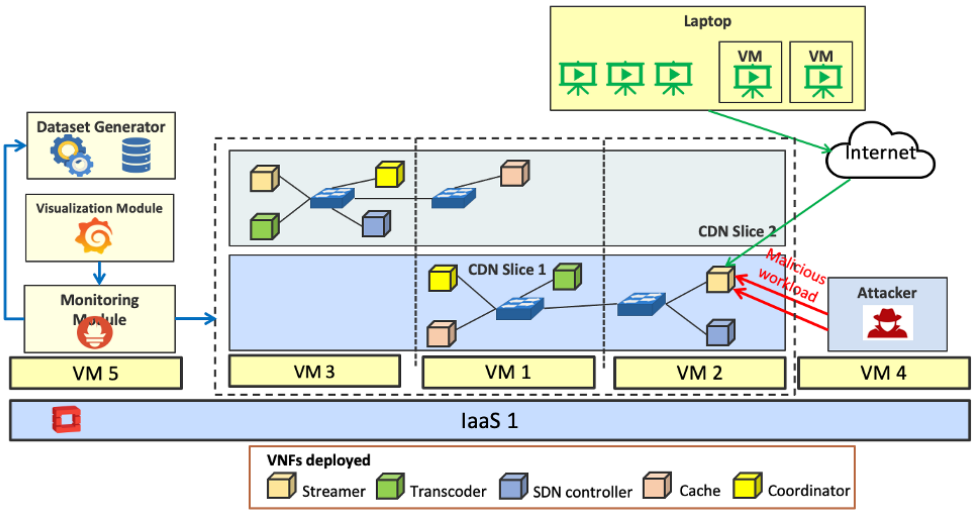

Figure 14: Testbed configuration for dataset generation

A dataset generator (see Figure 15) is developed using Python to extract the resource usage and system performance metrics as csv files. The generator extracts the features related to resource usage and system performance for all VMs and VNFs involved in slice 1 and slice 2 for a specified period defined by a “start date” and an “end date”. The “start date”, “end date”, “time step”, and the list of nodes (i.e., VMs, VNFs) from which the features should be extracted are provided to the generator in a json

file (param.json). The features to extract are formulated using PromQL queries submitted to Prometheus. The generator generates a csv file per node (i.e., VNF, VM), containing the time series of the extracted features of this node.

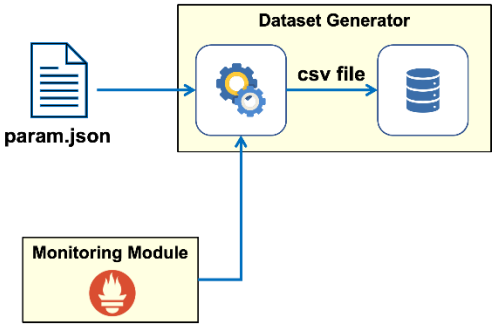

Figure 15: Dataset Generator

The model is trained on the time series for normal behaviour. The dataset contains 2761 samples, split into 2361 samples for training and 401 samples for test. The training is performed using 30 epochs and a batch size of 50. A validation ratio of 20% is used. Mean Absolute Error (MAE) is used as loss function and Rectified Linear Unit (ReLU) as activation function. To avoid overfitting, L2 regularization is used. To detect the anomalous workload, a static threshold is defined as 99% of the loss distribution. A separate model is built for each VNF and VM. Figure 16 illustrates the anomalies detected (red dots) for the video streamer under attack. As depicted in Figure 16, the preliminary results show the effectiveness of the DL-based DDoS Mitigator in detecting anomalies related to the launched Hulk and Slowloris attacks. However, we notice also that false positives are also generated.

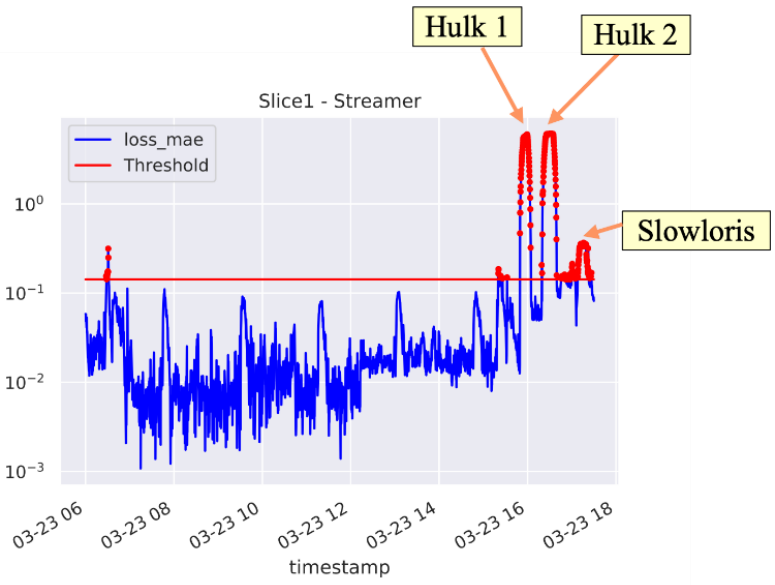

Figure 16: Anomaly Detection using DDoS Mitigator

It is worth mentioning that the implementation of the DDoS Mitigator model is still in progress. To improve its performance, we are undertaking different improvements, including:

- Creation of a new testbed based on a Kubernetes environment to deploy the CDN slices. The use of Kubernetes is motivated by its native support of the auto-scaling capability. More details on the new testbed can be found in deliverable D5.2.
- For effective detection of anomalous workload while reducing the false positives, we are increasing the size of the dataset used to train the DDoS Mitigator model. The new dataset set

will be generated using the Kubernetes testbed.

- We are developing a new DDoS Mitigator model that relies on metrics forecasting to detect the anomalous workload. To this end, we are leveraging GRU on multivariate time series. The motivation behind adopting GRUs is that GRUs are faster than LSTMs (due to their less complex structure) and can perform better than LSTMs on less training data.

3.1.5 Anti-GPS spoofing

3.1.5.1 Problem and challenges

Network slicing and management rely on the characteristics of user equipment (UE) mobility pattern and UE density. Such systems need the UE to report its location information to allocate the communication resources to a certain area. Nowadays, the Global Navigation Satellite System (GNSS), specifically GPS, is the primary location technology used by UE due to its global coverage and accuracy.

However, the unencrypted civil GPS signals are inherently vulnerable to spoofing attacks. Indeed, an attacker can use low-cost Software Defined Radio (SDR) tools, such as Universal Software Radio Peripheral (USRP), to generate fake GPS signals to fool the GPS receiver into calculating false positions. In another attack scenario, an adversary may deliberately report false GPS data to platform, which can lead to collision risks. Thus, without an accurate verification of the position claimed by a UE in an adversarial setting, the network may be deceived into allocating more resources to a mirage use case.

3.1.5.2 State of the art analysis

Several methods currently exist for the measurement of anti-GPS spoofing that are mainly focusing on GPS navigation signals analysis [59], GPS navigation message authentication [60], Inertial Navigation System (INS) based spoofing detection [61], and Mobile Positioning System (MPS) based spoofing detection [62][63]. The GPS navigation signals analysis detects the spoofed signal by estimating and comparing the Direction of Arrival (DoA) of the GPS signal, which requires multi-antennas for estimating the DoA of GPS signal or needs a secure GPS receiver to perform the cross-correlation and incurs more computational load on the GPS receiver. Compared with GPS signal analysis, GPS navigation message authentication does not need more antennas or additional receivers. The GPS Navigation Message Authentication (NMA) approach protects the civil GPS signal from attacking by embedding the cryptographic signature into the navigation messages. Nonetheless, even though NMA techniques are considered a practical and effective defence against GPS spoofing attacks, those techniques induce significant computational cost and latency due to signature verification. INS techniques detect GPS spoofing by using the position information estimated from the Inertial Measurement Unit (IMU) that consists of various on-board sensors including accelerometers, gyroscopes, magnetometers, and camera views, to cross-validate the veracity of the reported GPS position. Notwithstanding, the error accumulation of the IMU measurements is the main issue for INS, which can reduce the detection accuracy. Recently, MPS based spoofing detection has emerged as a new class of anti-GPS spoofing approaches that leverage the localization ability of mobile cellular networks to relocate the UE and discriminate the spoofed GPS positions in the base stations' coverage area. The MPS-based spoofing detection methods use the triangulation location technique, which requires at least three base stations at the same time for a desirable spoofing detection accuracy and is also sensitive to the environmental changes.

Note that the GPS spoofing detection methods discussed above either depend on expensive hardware or can be negatively affected by environment changes. Therefore, these detection methods are difficult to be used in resource constrained UEs.

3.1.5.3 Solutions descriptions and advancements

To date, there have been very few empirically published accounts of an effective GPS spoofing detection approach that accommodates resource, cost, and environmental constraints. For this purpose, we propose a new solution to devise an effective cellular-enabled UE GPS spoofing detection system, where MLP is used to analyse the statistical features of path losses between UE and Base Stations (BSs).

The proposed approach consists of three processes, namely data sampling, statistical analysis, and MLP prediction.

- **Data sampling:** There are two kinds of data that need to be collected in the processing, namely the actual path losses data and the theoretical path losses data. The actual path losses data are the observations of base stations, and the corresponding theoretical path losses data are computed according to both UE' GPS positions and BSs' GPS positions. One of the theoretical path loss models between BS and UE is defined in 3GPP document in [64]. The theoretical path loss model in [64] is adopted in our solution.

- **Statistical analysis:** A statistical analysis is then performed using moments (e.g., Mean Variance Skewness Kurtosis (MVSK)), quartile (e.g., BOX), and probability distributions difference (e.g., Wasserstein Distance (WD)) methods to extract the statistical characteristics of actual and theoretical path losses data. To obtain the metrics of path loss, we consider the number of data transmissions is *O* in a time duration *t*. The corresponding actual path loss set $L_{iv}(t)$ is expressed as:

$$\left\{L_{iv}^{1}(t), \dots, L_{iv}^{o}(t), \dots, L_{iv}^{O}(t)\right\}$$

where $L_{iv}^{o}(t)$ donates the o^{th} path loss in t between BS i and *UE* v. Similarly, the theoretical path loss set is defined as:

$\left\{\bar{L}_{iv}^{1}(t), \dots, \bar{L}_{iv}^{o}(t), \dots, \bar{L}_{iv}^{O}(t)\right\}$

where $\bar{L}_{iv}^{o}(t)$ denotes the o^{th} path loss in *t*. According to the MVSK method, we have

$L_{iv}^{M}(t) = \frac{1}{O}\sum_{o=1}^{O} L_{iv}^{o}(t)$,

$L_{iv}^{V}(t) = \frac{1}{O}\sum_{o=1}^{O}(L_{iv}^{o}(t) - L_{iv}^{M}(t))^2$,

$$L_{iv}^{S}(t) = \frac{1}{O}\sum_{o=1}^{O}\left(\frac{L_{iv}^{o}(t) - L_{iv}^{M}(t)}{\sqrt{L_{iv}^{V}(t)}}\right)^3,$$

$$L_{iv}^{K}(t) = \frac{1}{O}\sum_{o=1}^{O}\left(\frac{L_{iv}^{o}(t) - L_{iv}^{M}(t)}{\sqrt{L_{iv}^{V}(t)}}\right)^4,$$

where $L_{iv}^{M}(t), L_{iv}^{V}(t), L_{iv}^{S}(t)$ and $L_{iv}^{K}(t)$ represent, respectively, the mean, variance, skewness, and kurtosis values of actual path losses in time interval *t*. Similarly, we can obtain the theoretical ones $\bar{L}_{iv}^{M}(t), \bar{L}_{iv}^{V}(t), \bar{L}_{iv}^{S}(t)$ and $\bar{L}_{iv}^{K}(t)$. The details on BOX and WD metrics can be found at [65].

- **MLP prediction:** The statistical properties are used as inputs to the devised MLP models (See Figure 17). The input $\Delta L_{iv}^{x_m}$ denotes the m^{th} difference of path losses reported by BS i under each of the statistical methods (i.e., MVSK, BOX, WD). $m_x \in \{M, V, S, K\}$ for MVSK-based MLP (MVSK-MLP) model, $m_x \in \{Q_0, Q_1, Q_2, Q_3, Q_4\}$ for BOX-based MLP (BOX-MLP) model and $m_x = W$ for WD-based MLP (WD-MLP' model. The models' output, Prediction(*t*), is the final prediction decision, i.e., the reported GPS position is spoofed or not.

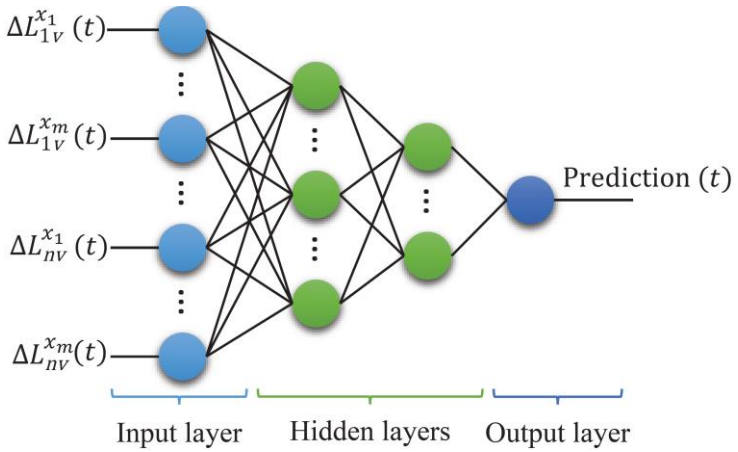

Figure 17: The structure of the MLP

The proposed anti-GPS spoofing approach can be deployed on the edge server without any additional hardware or computation load at the UE. In addition, its effectiveness is less prone to changes in environmental conditions, thanks to the stability introduced by the statistical features. By using the path losses that can be obtained from the BSs broadly and speedily, and by taking advantage of the capability of ML to deliver faster decisions, the proposed approach will empower live detection of spoofed GPS positions.

AI and ML contributions

It is well known that the performance of MLP is sensitive to hyperparameter settings [66]. Thus, to find the best configuration of the different MLP models of our study, we carry out hyperparameter tuning by varying the learning rate (LR), the number of hidden layers, and the number of neurons per hidden layer. LR is drawn from $\{0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$, the number of hidden layers is varied from one to six layers, and the number of neurons of each hidden layer is taken from $\{8, 16, 32, 64, 96, 128\}$. The hidden layers have the same number of neurons and use ReLU as activation function. The best model parameters for this task are shown in Table 3.

Scenario	Setting	MLP Algorithm		
		MVSK	BOX	WD
Three BSs	LR	0.005	0.001	0.0005
	Inputs	(12, 0)	(15, 0)	(3, 0)
	Hidden Layers	4	5	3
	Neurons	96	96	16
Two BSs	LR	0.001	0.001	0.001
	Inputs	(8, 0)	(10, 0)	(2, 0)
	Hidden Layers	2	5	2
	Neurons	32	96	64
One BS	LR	0.0001	0.001	0.0001
	Inputs	(4, 0)	(5, 0)	(1, 0)
	Hidden Layers	2	5	2
	Neurons	96	96	64

Table 3: MLP hyper-parameters

From Table 3, it is noticed that different configurations are required for MLP models to reach their best

performance under each of the three considered scenarios. The following key observations can be made: (i) the number of inputs to each MLP algorithm depends on the number of BSs involved in each scenario and the statistical method considered; (ii) the BOX-MLP algorithm needs more hidden layers and neurons compared to MVSK-MLP and WD-MLP algorithms. This can be explained by the fact that the number of hidden layers and neurons usually depends on the size of the input vector. In our case, BOX-MLP has the largest number of inputs; (iii) The WD-MLP algorithm uses at most three hidden layers and 64 neurons per hidden layer to achieve the best performance; (iv) Unlike MVSK-MLP and WD-MLP algorithms, WD-MLP algorithm requires the same MLP structure and LR to get the best performance for the three scenarios.

The models' efficiency in detecting GPS spoofing attack will depend greatly on the statistical metrics captured by the statistical methods. In fact, MVSK-MLP model requires a large amount of path loss data to ensure the accuracy of prediction. By removing outliers, the BOX-MLP model could mitigate the environment impacts on the path losses. Meanwhile, it will also lead to increased error in GPS spoofing detection since the outliers caused by attackers are also removed. The WD-MLP is used to describe the difference between actual and theoretical path losses, and thus generates only one feature value on the difference for each base station, which could result in unfitting problem.

In [65], we demonstrated the effectiveness of the proposed MLP-based anti-GPS spoofing approach in delivering accurate decisions about the authenticity of Unmanned Aerial Vehicle's (UAV) GPS positions. Thanks to the stability introduced by the statistical metrics, the prediction accuracy is greatly enhanced. Indeed, the developed MLP approach could achieve an accuracy rate that is above 93% with three base stations and can reach 80% with only one base station.

3.2 Data support for cognitive security techniques

3.2.1 Security data generation and collection

3.2.1.1 Problem and challenges

B5G network opens new opportunities to operators to apply ML to solve multiple problems, including advanced security management. To achieve these results, it is needed to invest a non-negligible quantity of effort in the data engineering process, including data sources identification, data transformation, and to evaluate conditions such as frequency and quality of the data. Network range-digital twin (NDT) appears as a potential solution for assessing solutions related to AI/ML architectures. This includes generation, collection, and transformation of data to design and test different ML models in an emulated environment before deploying in production, reducing the cost and investment. This model could fit for offline ML training, and ML inference engine delivery. Figure 18 shows the holistic process combining several enablers.

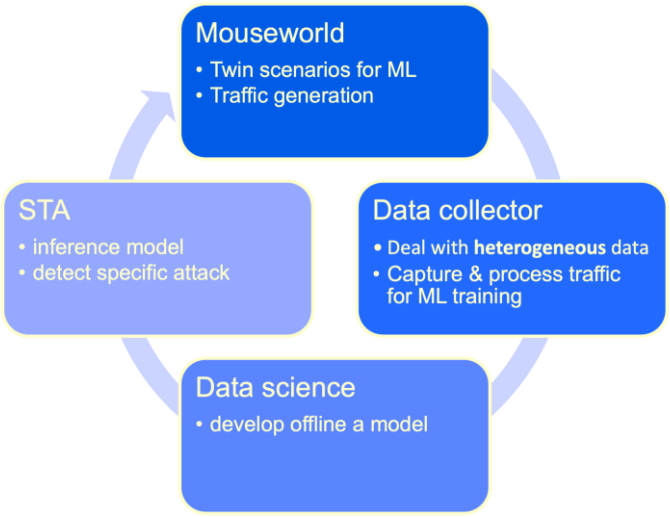

Figure 18: Data to ML cycle

Mouseworld acts as the 5G twin environment for network traffic generation and delivery. This network flows related information is collected and aggregated through a data collector. The output generated is delivered to design an ML model. The outcome is integrated into the Smart Traffic Analyzer and validated in the Mouseworld. If the model does not achieve the expectations or the traffic scenarios evolves, a redesign can be done.

3.2.1.2 State of the art analysis

ITU introduced the concept of ML sandbox in their Framework for ML in 5G (ITU-T Y.3172) for providing the capacity to simulate data pipelines to train ML models. Recently a clear effort is rising to clarify the concepts around the applicability of Digital Twin for networks. IETF has started this process [67], and one of the straightforward applications is the ML/AI.

3.2.1.3 Solutions descriptions and advancements

AI and ML contributions

Mouseworld

This lab has been conceived as a network Digital Twin infrastructure created to produce datasets related to network threats and train ML models. Complex network emulation and testing can be carried out thanks to the equipment and tools. One of the main characteristics of the lab is the ability to create/destroy several network scenarios to run different tests. The virtual scenarios are isolated from each other thanks to the NFV/SDN architecture, and therefore the traffic can be captured using VNF probes. This consistent environment allows evaluating different mitigation tools or versions under the same circumstances, making it perfect for developing and tuning machine learning network instruments.

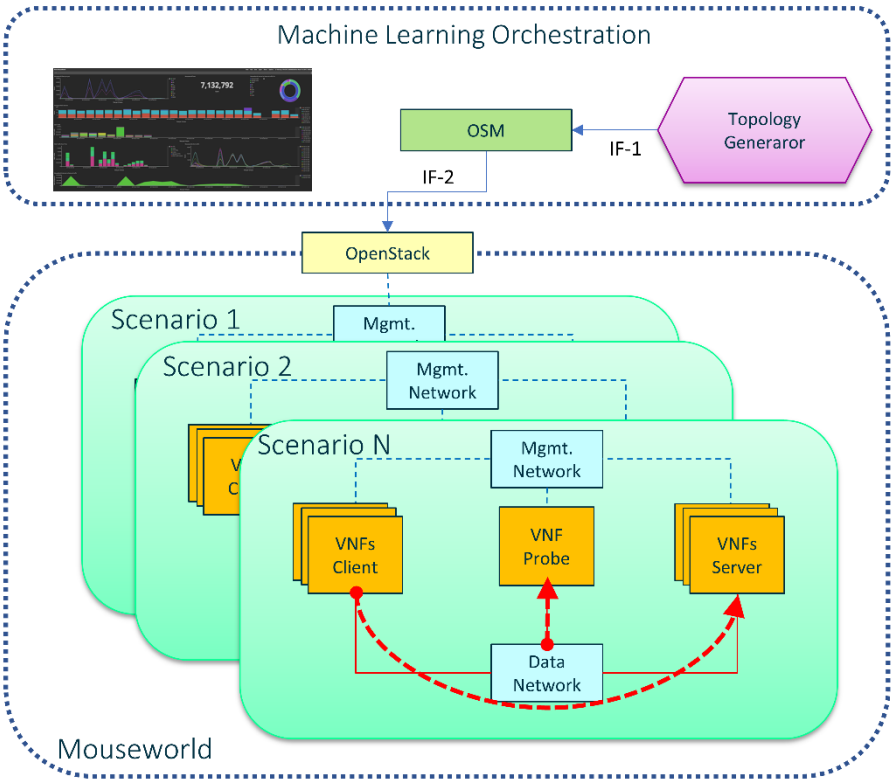

Figure 19: Mouseworld Lab components

The Mouseworld Lab mainly consists of a topology generator that creates the required virtual network elements and the interconnection between them, a Management and Orchestration (MANO) stack based on OSM and a Virtualized Infrastructure Manager (VIM) with OpenStack. Different VNFs can be deployed in each scenario, including a scheduler to coordinate the experiment execution, virtual instances of commercial traffic generator, and a probe that captures the traffic to be labelled and transformed to be used by data scientists. The lab components are shown in Figure 19.

The current activity is focused on adapting the environment to support 5G Core and access traffic emulation in specific scenarios to generate datasets. A basic scenario to generate cryptomining encrypted traffic (HTTPS) is combined with some HTTPS applications. The tools to generate the attacks and the dataset captured and used to train a binary classification model to detect cryptomining will be used in testbed as part of TC1/demo1 as a result.

Smart Traffic Analyzer (STA)

The Smart Traffic Analyzer solution introduced in D3.1 is shown in Figure 20. It can be considered as a network probe that monitors the network traffic and generates predictions based on specific ML models preloaded into their system. The final goal is to make the solution customized to specific problems just changing the ML model.

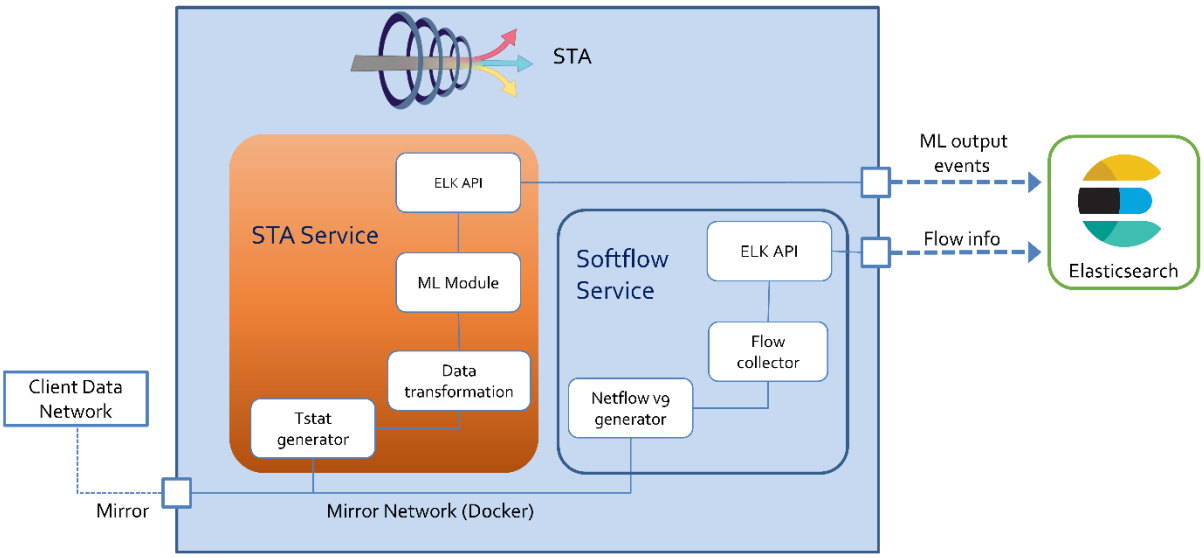

Figure 20: STA internal architecture

The current version architecture has evolved to a microservice architecture running separately in different docker containers. With the help of a docker network, the traffic is duplicated so that both services can receive and process it. Also, through an API described below, both services can send the processed information to another local docker running an ELK server or to external data analytics engines specified by configuration.

- **Softflow Service:** This service identifies the different flows from the traffic received through the interface connected to the mirrored network. This service stores the information of different flows, depending on their source and destination addresses, the transport protocol used, and the ports used by the traffic. Finalized or expired flows are delivered in NetFlow format to the internal flow collector, parsed and sent to the ELK.
- **STA Service:** The core of the STA is in charge of identifying and classifying the different types of traffic received through the mirror network by using an ML model. In this case, the service will be running a modification of tstat software [68] that extracts specific network traffic features based on the traffic flows statistics. Then using a python module, with a customized ML model trained to identify some types of traffic based on previous tstat features, it will assign a tag to each captured flow depending on the confidence level. Once the module has set a tag, the information about the flow will be sent to the ELK.

One of the STA service capabilities is the possibility to use different ML models depending on the traffic that is going to be analysed. This gives the ability to set new configurations based on the traffic to be classified. New models should be trained (see Mouseworld) before using them. Later, when a new ML model is changed, only it is necessary to add the new tstat features used in the training process and the possible tags to assign. This process will be done by configuration.

The model trained over the Mouseworld and integrated in STA uses a cryptomining binary classification, based on Random Forest to detect this malicious pattern inside a 5G SBA (Service Based Architecture) environment where everything is encrypted with TLS. The STA enabler uses only Layer 4 statistical metrics, avoiding payload inspection or decryption.

3.3 Robust AI/ML techniques for network security

3.3.1 Robust federated learning

3.3.1.1 Problem and challenges

The standard setting in ML considers centralized datasets which are tightly integrated into the system. However, in most real-world scenarios, data is usually distributed among multiple entities. More specifically, centralized data collection is challenging due to the higher communication cost for sending data, when the devices create large volumes of data, serious privacy issues coming with the sharing of sensitive data, overfitting issues with the small datasets and the biased local datasets. As a solution, federated training is proposed where each user and server collaborate to train a unified neural network model. This ML approach was formally published by Google in 2016 as Federated Learning (FL). Simply, FL is a distributed learning concept, where end devices or workers are participating for learning process. The central entity or parameter server shares the training model and aggregates the local model updates coming from workers. Workers train the shared model locally using their own data and send the trained model back to the central server. Central server aggregates the received models and shares the aggregated model with workers. The final model needs to be as good as the centralized solution (ideally), or at least better than what each party can learn on its own. Typically, FL brings the advantages in terms of improving privacy awareness, low communication overhead, and low latency. Most importantly, FL is suitable to address the distributed networking scenarios in the more complex networks.

However, FL is vulnerable to poisoning attacks by design (Figure 21). The central server can be poisoned using minimum of one adversarial worker. This will affect the learning process of the entire network. The problem is that the central server cannot guarantee that the workers provide accurate local models and have no control over the level of security at each worker. Another issue is that it is possible to encounter a single point of failure at the central server. Therefore, it is necessary to implement defence mechanisms at the central server to distinguish between poisonous and honest users. It is challenging since the central server has no validation data for verification of the model updates received by the workers.

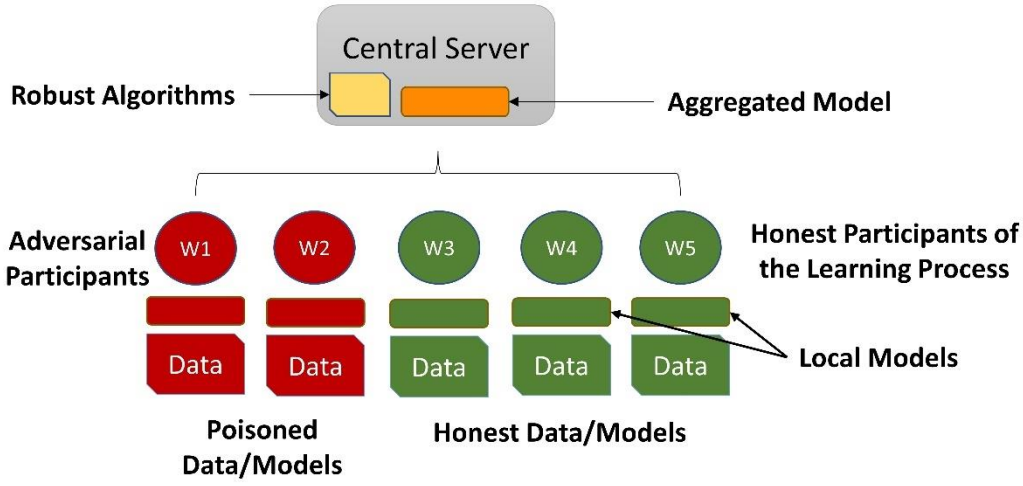

Figure 21: System model of poisoning attacks for Federated Learning

3.3.1.2 State of the art analysis

In the current state-of-the-art, some works are done to improve security in FL and to use FL as an enabler to improve security in the networks [69]. Mainly two types of attacks are identified in FL, namely data poisoning and model poisoning. In data poisoning attacks, the attackers modify the

training data in the workers by making workers' training incorrect and generating poisoned model updates. In model poisoning attacks, the attackers create poisoned model updates by manipulating benign model updates or according to pre-designed rules. The defence types are falling under three types [70].

1) Robust aggregation: The server aggregates model updates according to a special rule, instead of simply taking the average.

2) Anomaly detection: The server detects and then removes or devalues the poisoned model updates.

3) Hyper defence: A combination of robust aggregation and anomaly detection.

Currently used algorithms at the centralized server are focusing on distinguishing poisonous updates from the distributed workers. Some algorithms such as Krum [71], Bulyan [72], trimmed mean and median [73] are proposed for this purpose. All four algorithms are designed to detect adversaries when they try to deviate the learning process from a common goal. They discard the highly dissimilar model updates. Krum, Bulyan and trimmed mean algorithms require the estimation of the expected number of adversaries in the system as an input parameter. This is a major limitation of their practical application. The median algorithm can be easily compromised by increasing the number of attackers. The newly proposed algorithm called *FoolsGold* has reasonable improved performance than the above ones since it does not need the prior knowledge of adversaries (e.g., expected number of adversaries or validation dataset at the server).

3.3.1.3 Solutions descriptions and advancements

AI and ML contributions

The targeted poisoning attack on FL system is described as follows. Adversary controls *C* poisonous nodes where the system has total number of *n* nodes. We exploit the weaknesses of the existing FoolsGold design and perform a targeted model poisoning attack to circumvent its defence. For the experiments, we use a single layer fully-connected softmax for classification with MNIST digit classifier dataset [74]. FoolsGold algorithm (Figure 22) primarily depends on the Cosine Similarity (CS) of the received gradient updates. We intelligently introduce random noise to the gradient updates at poisonous nodes. The poisonous nodes train their local models with label flipped data and then add the noise before sending them to the central server. A noise vector specifically replaces a part of a given poisonous gradient update to add extra dissimilarity. Only the gradient updates corresponding to the less important features of the model is placed with noise while the vital features required for model training will not be propagated with noise. This ensures that the noise does not reduce the effectiveness of the attack. We use a subset of data at the devices using Stochastic Gradient Descent (SGD) method which reduces the training times. This is done to avoid the inconsistencies among training times due to the different amounts of data.

Algorithm 1: Attack FoolsGold with Intelligent Noise

Data: Local SGD update Δ_i of each attacker i, Attacker count C, Noise intensity I, Indexes of gradient values to be replaced
Result: Noisy gradient vectors for attackers
Attacker groups = $C \div 4$;
for *Each attacker group* **do**
  Create noise vector N_1;
  Generate N_2 such that $N_1 \perp N_2$;
  Generate $-N_1$ and $-N_2$;
  Create $I.N_1$, $I.N_2$, $-I.N_1$, $-I.N_2$;
  Replace the selected values of Δ_j of attacker j with noise values from $I.N_1$, $I.N_2$, $-I.N_1$, $-I.N_2$;
end

Figure 22: Algorithm for attack creation on the FoolsGold algorithm

We generate two orthogonal noise vectors N_1 and N_2 that comprise of x features. We also take the negatives of those vectors, $-N_1$ and $-N_2$. The cosine similarity among these four vectors are $CS(N_1, N_2) = 0$, $CS(-N_1,-N_2) = 0$, $CS(N_1,-N_1) = -1$ and $CS(N_2,-N_2) = -1$. We then multiply each of these four vectors by "noise intensity" parameter. The scalar multiplication does not affect the cosine similarity values. We then replace some of the gradient values of the four poisonous nodes of a given group by these four vectors. The value for the noise intensity should be chosen in a way that, the noise values are significantly higher than the gradient values. Then the noise values have a greater effect on the cosine similarity, even though the full gradient vectors are not orthogonal or negative to each other. The domination of noise due to a higher noise intensity keeps the cosine similarity at a lower value. This way, the four users of a given group has a minimum cosine similarity. We repeat the same procedure for the poisonous nodes in other poisonous groups. Algorithm in Figure 22 depicts the procedure for intelligent noise addition.

4 The security of AI/ML and protection techniques

As AI/ML techniques become critical in closed-loop and autonomous operation for network security, security of AI/ML also become crucial. This issue is augmented with a software security concern as the network assets including cognitive security functions themselves are becoming increasingly a system of software systems in communication networks. This evolution is to be exacerbated with the emergence of Beyond 5G or 6G networks. Therefore, we extend our security analysis towards AI/ML security and present our work in this section.

4.1 Protection against adversarial attacks

The key role that AI plays and will play in enabling fully autonomous security management capabilities is indisputable. However, its major role makes it an attractive target for attackers [66]. Indeed, AI systems, specifically ML systems, can be fooled to learn wrong models, make incorrect decisions, or leak confidential information [75]. In the INSPIRE-5Gplus project, we conducted a comprehensive investigation of the security risks stemming from the exploitation of AI vulnerabilities by adversaries. We also recommended several defence measures while advocating on which components of the ITU-T's ML5G (Machine Learning for Future Networks including 5G) unified architecture they could be enforced. More details on the main outcomes of the conducted study can be found in INSPIRE-5Gplus's D2.2 and [76].

Driven by the key role of DL in implementing some of INSPIRE-5Gplus's security enablers, as presented in the previous sections, we conducted a practical study demonstrating the vulnerability of DL models to adversarial attacks as well as the effectiveness of adversarial learning defence in building DL models that are robust to adversarial attacks [75]. To this end, we considered the MLP-based DDoS Detector Model introduced in Section 3.1.4.

The adversarial attacker is assumed to have a full-knowledge (i.e., white-box attack) on the targeted model to generate adversarial flows that will be misclassified by the model (i.e., (D)DoS flow classified as normal flow or vice versa). Three attacks are considered:

- Fast Gradient Sign Method (FGSM) [77], which generates adversarial examples by performing a one-step gradient update in the direction of the gradient's sign of the loss function relative to the input. The input is then altered by adding a perturbation that can be expressed as:

  $$\eta = \epsilon . sign(\nabla_x J(\theta, x, y))$$

  where x is a sample (i. e. , network flow), y is the label of x, $J(\theta, x, y)$ is the loss function used to generate the adversarial example and ϵ is the perturbation magnitude. Thus, the adversarial sample is generated as

  $$x^* = x + \eta$$

- Basic Iterative Method (BIM) [78], which extends FGSM by applying it iteratively with small step size and clipping the values of the adversarial example after each step such that they are within an ϵ-neighborhood of the original sample. This gives the following recursive formula:

  $$x_0^* = x,$$
  $$x_i^* = clip_{x,\epsilon}\left(x_{i-1}^* + \epsilon\ sign\left(\nabla_{x_{i-1}^*} J(\theta, x_{i-1}^*, y)\right)\right)$$

- Jacobian-based Saliency Map Attack (JSMA) [79], which creates an adversarial example by perturbing the minimal number of features of the original sample based on Saliency map, which is defined as

$$S(X,t)[i] = \begin{cases} 0, if\ \frac{\partial F_t(X)}{\partial X_i} < 0\ or\ \sum_{j \neq t} \frac{\partial F_t(X)}{\partial X_i} > 0 \\ \left(\frac{\partial F_t(X)}{\partial X_i}\right) \left| \sum_{j \neq t} \frac{\partial F_t(X)}{\partial X_i} \right|, otherwise \end{cases}$$

for a feature *i* of an input sample *X* and a neural network *F*. In fact, the adversary aims to misclassify a sample *X* such that it is assigned a target class $t \neq label(X)$. To this end, the probability of target class $F_t(X)$ must be increased while the probabilities $F_j(X)$ of all other classes $j \neq t$ decrease, until $t = \arg max_j\ F_j(X)$.

The attacks are implemented using the Cleverhans library [80] to craft adversarial examples from the test set. The adversarial training is performed based on adversarial examples generated using FGSM and BIM attacks. For the three attacks, we observed a significant drop in the original model's accuracy. It is worth mentioning that the advantage of JSMA over FGSM and BIM lies in reducing the number of perturbed features, making the generation of adversarial network flows more feasible. The results show also that the model's robustness is considerably improved when the model is trained with adversarial examples generated by the same attack. More details on the study can be found in [70].

As a PoC, we improved the application-layer DDoS self-protection framework presented in Section 3.1.4 by replacing the original DDoS Detector model with a robust DDoS Detector model. In addition to a normal attacker, we defined a *smart attacker* as an attacker who has the capability to launch an application-layer DDoS attack while evading detection by a ML-based detector. A smart attacker is supposed to be able to craft application-layer (D)DoS flow that will be misclassified as a legitimate flow by the ML-based model. It has the capability to carry out white-box FGSM attacks against the DDoS Detector model. To make the DDoS Detector model resilient to adversarial attacks, the model has been adversarially trained on adversarial flows generated by the same attack (i.e., FGSM). Through this PoC, we could demonstrate that while the original DDoS Detector model fails in detecting the adversarially-generated application-layer DDoS attacks, the use of its adversarially-trained variant allowed to prevent both ordinary and adversarially-generated attacks. More details on the performance results of the robust DDoS Detector model can be found in [55].

4.2 Elevating trust on AI by software security

To define how and where software security is useful for elevating AI/ML security, a good start is a focused technical survey categorizing and detailing the modes of attack and the classical existing mitigations. Before that, however, it is appropriate to state the discrepancies between a general-purpose data processing algorithm and AI/ML ones, in the perspective of their security. Our survey is produced at two perimeters successively. First, we consider any type of AI/ML application. Secondly, we look more precisely at the AI/ML systems implemented in 5G networks. This scope restriction is aimed at establishing what the most salient threats in AI/ML in operation in 5G networks are. We can better elaborate the attack areas laid over the data extraction and processing pipeline as well as assess if software security can bring any valuable contribution there. When this work is completed, we produce a specific technical survey on the merits of trusted execution environment to frustrate the model stealing. We then conclude by providing the results and lessons learnt from our technical investigation of Intel SGX (Software Guard Extensions) used as a shield for the model as well as other techniques applied on the data collection and inspired from this research work.

4.2.1 Specificities of AI/ML solutions (in view of their software security)

AI/ML systems are not developed and conceived by the application developer as (any) other data processing software is. The divergence relies on the core code (learning algorithm) genesis, not developed by the application developers but conversely by machine learning framework teams.

Algorithm and framework are common bricks used for several very different applications by the AI/ML (vertical) solution designers. Some depicts this situation as “AI is automatic programming” which is a factual statement as no one needs to program it. A common pre-existing software is customized and tuned automatically at the training stage resulting in the production of the model parameters. The added value or intellectual property of the AI/ML vertical application designer is in the appropriate selection of the (pre-existing) algorithm and its training with the well- or best-fitted training data. A stock phrase is that AI/ML quality depends on its training data. As a consequence, with AI/ML, the intellectual property resides precisely on the model’s parameters which consists in a data structure, not an algorithm. This entails one critical Intellectual Property Rights (IPR) issue.

The higher importance of data versus software for correct functioning of AI/ML and on the intellectual property value only fades partially the relevance of software security. In the context of an open-source AI/ML algorithm code, software confidentiality is still surprisingly important as a properly targeted attack on an AI/ML solution is far easier to spawn with the accurate knowledge of algorithm. In practice, there are many of these algorithms. Keeping the algorithm confidential is not aimed at securing the intellectual property but to conceal and hide it, thus creating opacity on its actual type to the attacker, drifting the attack from a defined strategy based on certain assumptions into guesses. On the other side, considering the model parameter security, it is at least equally important to elaborate how these data can be protected leveraging software security in appropriate schemes. In practice, securing the software which itself secures and validates the data is a valuable security boost.

When software security is discussed, a common statement is that open-source code is allegedly safer as being (security-wise) improved by a massive class of users. The rationale at stake is a lower prevalence of pending vulnerabilities to exploit than in proprietary code of lesser use. This argument is a dual sword argument as hackers will first target highly employed software to carve their attack and especially if the code can be accessed for static analysis, as is AI/ML open-source code. In short, AI/ML algorithms, despite being probably less exploitable, are attractive vulnerability-based attack targets. One residual found vulnerability jeopardizes all derived AI/ML solutions employing it.

A last element is the predominance of Python language in the AI/ML frameworks and, more generally, the preferred natural choice for any AI/ML application developer today. The security implications are discussed below.

4.2.2 Classical attack paths on AI/ML and associated mitigations

AI/ML has outweighed all ICT trends for a decade. By delivering the means to automatically generate applications for classification or inference, while classical coding would be highly complex if ever practical, the number of involved applications and industrial sectors has exploded. Because AI/ML becomes the decision-taking component of safety-critical systems, a massive research work has been produced on AI/ML security, safety, privacy and explainability.

AI/ML security is a massively discussed academics research topics in the last decade. For a comprehensive analysis, we would consider the reading of [81] which brings a clear taxonomy and it is relatively recent. An adversarial attacks survey is given here [82], while defensive techniques are detailed in [83]. Evasion attacks are detailed and categorized in [84], while Poisoning attacks are detailed by the same team in [85]. An interesting open-source tool box for Python AI/ML is given in [86]. It is worth stating that this bibliography is only aimed at delivering a general overview as new publications are continuously produced. For further information, four main classes of Adversarial ML (AML) are given in Table 6 in Appendix A.

From this initial technical investigation in T3.3, our recollection includes:

- AI/ML is driven by data analytics and statistics processing where the efficiency is related to performance, data quality and constant feature distribution laws (from training to test samples). Classical data-driven defences are equally featuring probability rules and detection scores. In this probabilistic domain, any techniques which help in dropping the attack success rate or the return on investment (e.g., Machine Learning-as-a-Service (MaaS) black box model

stealing) shall be considered.

- Defences directed to a specific attack vector could degrade the efficiency of the model elsewhere. Generic and low-level defences shall be prioritized versus focused ones.
- Some discussed defences (e.g., Ensemble, Cascade, gradient masking) are probably too complex for wildcard developers and more likely to be used by AI-as-a-Service (AIaaS) vendors experts.
- Input denoising and evasion detection techniques are interesting as being generic.
- Getting the model in clear is not a strict requirement for a successful attack but a strong driver.
- Model stealing although referred as an attack per-se while it is a first step of adversarial learning attacks (i.e., evasion and poisoning).
- The transferability principle is a main driver in black box conditions.
- Retrained models are (far more) exposed to poisoning attacks. AI/ML techniques implemented on 5G network management belong to this category.

4.2.3 Software security relevance for AI/ML solutions in networking.

With softwarization of networks, there emerges tight coupling between software security and network security regarding AI/ML schemes. Leveraging AI/ML solutions for autonomic real time management and optimization of the next generation of telecom network is a technical domain per-se. While there is a stock phrase stating that leveraging AI/ML is an unavoidable for the management of 5G networks, there is the same appreciation for the security risks this situation poses. Moreover, if AI/ML driven control is compromised, that may lead to large-scale impact due to automation and closed-loop control.

As all pre-cited techniques, there is no absolute defence and all defences shall be studied with their mitigation efficiency, possible accuracy decrease on clean data and their associated performance impact. Finally, the MTD principle applied on AI frustrates the carving of attacks targeted for fixed targets (i.e., static algorithm, static model, static training data), through dynamicity and variability. This research is certainly at its initial stage and as stated by the authors will be discussed on the heavy workflow and performance impact issues.

The technical survey made above, in a general perspective, in the context of applying AI/ML for 5G and Beyond networking, leads us the following conclusions:

- In this domain of Adversarial Machine Learning (AML), both attackers and defenders develop data-based techniques derived from data analytics and data statistics with the idea to carve attack vectors with high hit rate and equivalent detection ability on the other side. Both sides reside in some kind of uncertainty. Every technique which hardens the life of the attacker and drop the attack efficiency will be welcome and analysed in terms of “operational” security merits, overhead and setup issues.
- Before taking a closer look on the most promising software security techniques and associated best use for AI/ML pipeline security, it is worth restating the large functional scope brought by software security measures. Various complementary techniques can be applied on software for its security. From vulnerability remediation to variant singularization, execution control and run-time monitoring, the scope of possible hardening techniques is large and largely unemployed as of today.

4.2.3.1 Novel software security techniques applied to AI/ML implementations.

We introduce and discuss here techniques which have recently emerged in a general perspective, or which have not yet been applied to AI/ML domain yet (to the best of our knowledge).

Software singularization: The idea is to generate different variants for each deployment. The differentiation is created on the software binary with a differentiation ranging from simple identity secret appending (on the same untouched binary for all deployments) to the most complex scheme where each line of code is rewritten/reordered. While the former scheme provides one identification label possibly gathered at a centralized MANO, the latter employs MTD in the context of vulnerability exploitation attack.

Software remote control and monitoring: Software executable artefacts can be modified to insert monitoring probes (i.e., typically inserted at each instruction block of the software flow graph) to infer abnormal execution patterns collected and analysed either at the same execution platform or in a remote centralized workstation. Identically, the same artefacts can be modified to check at run-time their integrity. The centralized control solution would make certain that tampered variants execution stops and that only variants which show regular execution pattern keep running. A rollback to the original form of the artefacts can be initiated from a tampering detection.

Software geolocation: Software geo-location refers to a link that binds the software execution on a specific machine (or set of machines). Knowing the bound machine's location(s) brings software geo-location. The binding can be loose or strong according to the hardware anchoring type, ranging from a simple secret stored on the machine at a known path of the file system up to the storage in a trusted execution environment for that secret. When combined with variant singularization, it enforces that one specific variant executes only at a one given location which has been provisioned with the secret.

Software self-authentication (and associated self-boot): Deploying and running authenticated software is of primary importance for the networking industry and, more generally, cloud operated services. With no claim to replace the highly specified authentication scheme, a complementary traversal and in-depth self-authentication of binary files at their initiation stage can be offered. The essential benefit of the self-authentication light scheme is its total independence to the infrastructure's technical requirements which are required for classical remote authentication schemes (e.g., kernel-based report verification primitive, Trusted Platform Module (TPM) on the platforms, local agents, and remote verifier). Self-authentication (and its associated self-boot) are inserted routines inside the original binary file at protection stage. The report and the needed key to check it are also appended inside the binary. In the perspective of instance singularization, self-authentication releases the burden of delivery different measurements corresponding to each singular variant to a verifier and the associated complex workflow management. As a matter of fact, each variant carries its own report and produces the authentication verification test (on itself) by itself. For further information, Table 7 in Appendix A summarizes the security contributions brought by software security in AI/ML solutions.

As a conclusion, it comes that classical (e.g., Trusted Execution Environment (TEE), remote authentication, runtime integrity) and novel (singularization, self-authentication, geo-location, control flow run-time monitoring and execution control) techniques can significantly improve AI/ML taking part in 5G networks, independently to what data-centric AML techniques would bring. The former (classical) techniques are already employed. TEE by its protection of both model data and software offers a strong defence against model stealing. TEE concept actually turns the white box attack conditions into black box conditions. The following section makes a focus on TEE. The latter (novel techniques) are not used yet and bring new means to detect and inhibit nefarious nodes or data, specifically in distributed machine learning implementations.

4.3 TEE utilization for AI/ML security in 5G networking

Model stealing is the main gate for the attackers and, in most cases (e.g., poisoning, evasion), the first stage or a strong facilitator for the attack. TEE security contribution is unique as it brings ultimate confidentiality and integrity to both model components altogether (parameters and algorithm). TEE concepts, along with their weaknesses (e.g., performance impact, workflow consideration) and risks (e.g., side channel attacks) have been discussed in deliverables D2.1 and D2.2, respectively. In short,

whatever past successfully mounted side channel attacks on Intel SGX enclaves, they can only be successful on pre-known victim code (which could eventually become the case if an *enclaved* framework (i.e., TensorFlow inside SGX) becomes too popular. The main concern with TEE relates to software workflow (e.g., processor-dependent, software changes for Intel SGX, and required expertise to setup) as well as the performance costs. What follows details how TEE concept has been used and will probably be used to prevent model stealing attacks precisely on a general perspective.

4.3.1 State of the art

In [87], the authors first recall the security and data privacy loss risks exposed by multi-party learning models likely to take place in 5G network management (e.g., operators may not share their network operating metadata) as well as the merits of Intel SGX to mitigate these risks. Because of the expected performance losses incumbent to SGX, the authors produce some optimizations for customized binary integration of learning algorithms (K-means, CNN, SVM, Matrix factorization) and stress the requirements for data obliviousness which preserve privacy for the training and sample data, collected and generated outside SGX. In doing so, the authors map the security and privacy issues holistically, all way through the complete AI data pipeline. The incurred overhead when running the model inside SGX varies from a more than satisfactory 1% to a more impacting 91% according to the algorithm type (respectively, CNN and K-Means). In [88], the authors deliver efficient deep learning on multi-source private data, leveraging Differential Privacy (DP) on commercial TEEs. Their technology dubbed MYELIN shows similar performance (or negligible slow down) when applying DP-protected ML. To do so, their implementation goes through the compilation of a static library embedding the core minimal routines. The static library is then fully run in the TEE, which removes any costly context switch from the TEE mode to the normal execution mode. Specialized hardware accelerators (Tensor Processing Units - TPUs) are also viewed as the necessary step to take for highly demanding (fast) decision taking. That is a grey area, with no existing TEE embodiment for specialized hardware to the best of our knowledge. In addition, leveraging TEE data sealing capability looks like another path to consider for further improvements. In [89], the authors deliver a fast, verifiable and private execution of neural networks in trusted hardware, leveraging a commercial TEE. SLALOM splits the execution between a Graphics Processing Unit (GPU) and the TEE while delivering security assurance on the GPU operation correctness using Freivalds's algorithm. Outsourcing linear process from the TEE to the GPU is aimed at boosting performance, in a scheme that can be applied to any faster co-processor. Full TEE-embedded inference was the bottom line of this research, deemed as not satisfactory on the performance aspect.

In [90] , the authors recall the need for ever-growing and security-privacy sensitive training data set which calls for cloud operation, but this comes with its own off-premise security issues. The authors describe the cloud operation security threat as being training data and model stealing by a cloud operator and advocate for the perfect remediation of these risks by leveraging SGX enclave TEE. For that, they employ SCONE framework which drastically limits the efforts to integrate an application inside SGX. TensorSCONE design comprises two main components placed within SGX: TensorSCONE controller interfacing with the system for system calls (network, filesystem, user space thread) and on the other side the TensorFlow Library which enables to deploy untouched TensorFlow application. The authors describe the integration of the different TensorFlow components, namely the learning from TensorFlow and the classification from TensorFlow Lite, the feature-reduced variant designed for mobile and embedded devices. The objective of this selection is to meet the SGX's Enclave Page Cache (EPC) memory restriction (below 128 Mb for older devices, while from Intel 10th Gen the limit was raised to 1TB) while maintaining TensorFlow features, when possible, to keep the ML application framework easy and efficient. The authors discuss the performance losses compared to native TensorFlow implementations at both stages of training and classification, with a benchmark on image recognition (i.e., inception-v4). Classification throughput is degraded with a ratio 3, while training latency is inflated with a ratio 4 to 8. The authors do not foresee or discuss major gains in performance while have gone deep in their understanding the several causes of SGX induced losses (inevitable page swapping for large size sampling and training data) and instead consider approximate computing as a

possible loss saver. In [91], the authors propose a design to extend GPUs hardware with a hardware standard extension to offload kernel and security sensitive applications into the GPU. Graviton is a design to break the classical TEE performance bottleneck. Beside the detailed design explanations, the authors describe a typical implementation as Cifar-10 convolutional network. As the solution is not materialized in a chip, simulation is used to evaluate the performance overhead as being in the range from 53% to 73% according to the training phases. As AI/ML is becoming the hottest trend in ICT at the current time and GPU are well-fitted to parallelized processing of model generation, we may see emerging GPUs coping with AI/ML application security, featuring probably less workflow constraints and performance overhead than classical TEEs. Graviton is probably the initial (emulation) step in this direction.

Securing AI by means of TEEs is a nascent discipline, at its early stage, with encouraging results recently reported. From this first survey however, it appears that turnkey frameworks (TensorSCONE) are far more costly in terms of performance overhead compared to what can be achieved by more demanding customized integration. We expect that performance of AI/ML oriented SGX tools will be improved and optimized over time.

4.3.2 Lessons learnt in leveraging SGX

Intel SGX allows protecting in confidentiality, integrity, authentication and freshness of code and data specifically implemented for it. It allows to attest the Trusted Computing Base (TCB), to make sure the hardware is not being spoofed, and allows the provisioning and storage of secrets as a foundation to build up complex solutions. It has been designed to separate parts of a process and shelter them from the rest of the host. In this definition the Operating System (OS) is not considered a part of the TCB and, right for this reason, system calls are not supported from within the enclave, because they would execute untrusted code from the OS. Still, SGX includes instructions that let the code temporarily exit the enclave to call untrusted code so to perform, for example, I/O and then return to the enclave; this imposes a performance penalty and provides opportunities for bad implementations.

This implies that any input coming from the untrusted world, for example plain file or network reads, brings no guarantees on the validity and integrity of the data, since it might be tampered anytime by the host. To prevent this, a security mechanism must be put in place between the parties that are interconnected by enclaves (e.g., encrypt ed files or secure TLS connections with remote hosts) and the data should only be decrypted inside the enclaves. With these guidelines in mind, in the next paragraphs we are going to evaluate the challenges in designing a seamless secure data flow in a machine learning lifecycle.

Secure AI in MLOps

Maintaining a ML solution in production means taking care of multiple interconnected systems: data collection, data processing, feature engineering, data labelling, model design, model training, endpoint deployment, and endpoint monitoring (Figure 23). Each of these steps is often the result of a combination of interactions with multiple heterogeneous subsystems, for this reason putting in place a security-aware baseline becomes a challenging task if not implemented right from the design stage of such a complex architecture.

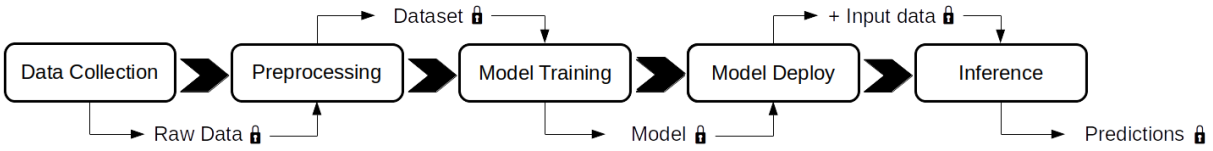

Figure 23: ML pipeline

In the following part, we evaluate the efforts and results of bringing a measurable incremental security improvement using Intel SGX. For the sake of simplification, the complete machine learning pipeline has been broken down into three main ML lifecycle phases, but clearly it can be applied to fit any different scenario. The phases taken into examination are:

1. dataset collection and processing,
2. model (re-) training
3. run-time inference.

The security criteria being evaluated are data confidentiality, data integrity and traceability of all involved parties. In particular, in the dataset collection phase we address the threats of stealing or poisoning of the training data, which could undermine respectively the intellectual property or the accuracy and reliability of the solution. Downstream, to the training phase, there will be an additional verification of the integrity of the dataset, aimed again to counter poisoning attacks of the data corpus that was transmitted to the model training infrastructure. At inference time, the goal is to protect the intellectual property of the model, against white box stealing, and to protect the integrity of the model and of the inference process itself, to guarantee the accuracy and reliability of the solution.

Hands-on experiment: Securing dataset collection and processing with Intel SGX

As described in Section 4.2, to ensure the data gathered in this phase is legitimate and not tampered, in order to ensure a seamless secure data pipeline, it may be necessary to protect additional upstream processes involved in data generation or collection.

For example, input data may come from sensors implemented with secured hardware, but in case of generative software processes, typically in RL, or in case of data coming from log files, even these additional processes may require a secure implementation with Intel SGX.

To test the efforts and the impact of implementing security with Intel SGX in a data pipeline pre-processing stage, we examined one use-case that involves the processing of a dataset made of 1,593 XML documents weighting on average 42,079 KB, for a total size of 67 MB.

The original code, written in Python, for each file, collects a subset of relevant fields and serializes them to a corresponding JSON file. In the normal use-case, the task is performed in the context of a micro web-service and the response to each request is sent to other tasks downstream. To simplify the measuring of performances, in the test environment the input and output files are read and saved on local storage.

To simulate the security context around this specific pre-processing stage, we encrypted each file of the input dataset using the SGX standard AES GCM 128. This operation increased the dataset size of just 44,604 bytes due to padding, intrinsic to the encryption algorithm.

Instead of porting the Python code to C++ to use the standard Intel SGX SDK directly, we decided to re-implement the code in Go language, using EGo, an open-source SDK that enables to develop SGX-powered confidential apps in the Go programming language [92].

Compared to the direct use of Intel SGX in C++, the ease of EGo comes with a price of slightly over 8s in order to boot the process. This overhead is only sensible at launch, and it is due to how EGo loads the main binary into the enclave. Since, as stated above, the normal execution context of the pre-processing would be within a web service, the slower start-up time should not be considered in our benchmark, having the process always on.

Porting the code from Python to Go language was straightforward, since both languages come with feature-rich standard libraries and are surrounded by prolific open-source communities. The use of EGo significantly reduced the time that would have been necessary to port the code in C++ and to use the SGX SDK directly. On the other hand, the biggest drawback was the lack of control in defining the clear-cut boundaries of what is computed inside the enclave and what in the untrusted domain. Therefore, performance is often lost in unnecessary context switches, dealt by expensive ECALL and OCALL instructions, to jump in and out of the enclave.

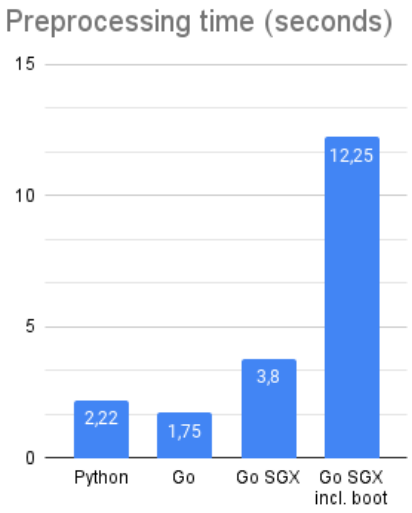

Figure 24: Comparison of dataset pre-processing times with/without SGX

Figure 24 shows that the Go implementation of the code, even if it had the additional tasks of decrypting the input files and encrypting the output after the pre-processing, performed over 20% faster than the Python counterpart. This result was expected, being Go binaries compiled into native code, instead of having the Python interpreter. The EGo SGX version of the implementation comes with a performance overhead of about of 1.58 seconds (71%) from 2.22 seconds up to 3.8. Such a performance penalty could be considered negligible in all ML use-cases where confidentiality and integrity of the dataset, and of the complete solution in general, are not an option.

In any case, the result of a 71% increase must be taken as an upper bound, because an in-depth analysis of the resulting Go binary confirmed that the generated code performs a significant amount of trusted world to untrusted world (and vice versa) context switches, causing most of the penalty. They are due to how the XML data is deserialized into the corresponding data structures and how the result is then serialized into the final JSON, specifically performing system calls to dynamically allocate memory, to read and write files.

The performances were measured on an Intel Core i7-7700HQ CPU at 2.80GHz, running on an Ubuntu 18.04.6 LTS with 16GB of RAM.

Hands-on experiment: Model training

Prior to the availability of 10th generation of Intel processors, which increased the limitation of the SGX EPC size from 128MB to 1TB, securely training deep learning models on SGX was deemed unfeasible, due to the memory-intensive nature of the task, that requires too many costly memory management operations.

In general, handling sensitive data in machine learning datasets is a hard and not yet solved problem. Several techniques are known in literature to avoid disclosure of sensitive data during the training phase of a model, most of which comes as a way to comply with various regulations such as HIPAA, PCI-DSS or GDPR. Most used techniques revolve around modifying the dataset before transmitting it to the AI/ML training compute provider. These modifications are often a combination of:

- **Data removal:** when the information is not strictly necessary for the project, it can simply be suppressed from the input data.
- **Data masking:** when the data is indeed useful, it may be possible to mask it using in a non-destructive way. Known techniques are:
    - Using a substitution cipher, replacing all occurrences of it with an encrypted or hashed counterpart.
    - Performing tokenization (also referred to as pseudonymization) which replaces

sensitive values with an unrelated dummy token and adds an additional layer of protection by storing the substitution tokens into a separate database.

- Using a dimension-reducing technique, such as Principal Component Analysis, to combine several features and then carry out ML training only on the new combined feature. While this approach is quite secure, because relies on the secrecy of dimension-reducing formula, it often comes at a cost of accuracy.

- **Data coarsening:** the idea is to lower the resolution, precision or granularity of data, like by rounding numeric data, masking bits in IP addresses or generalize geo-location data.

Encryption and dimension-reducing techniques are susceptible to frequency analysis attack, while data removal, tokenization and coarsening may significantly reduce the performance of the predictions.

The size and quality of the training dataset is usually a key factor to the success of an AI solution. In addition, for compliance with various regulations, there are often confidentiality issues related to sensitive data in the datasets. In such cases, performing the training of a neural network on the cloud can pose significant intellectual property or data confidentiality problems.

For this reason, it is reasonable to expect in the near future security-aware dedicated hardware for machine learning able to guarantee end-to-end confidentiality of the dataset and of the resulting trained model. In the meantime, several vendors of cloud services started to offer in-database model creation, training and inference local to the data warehouse. This solution avoids all the security risks involved in distributing confidential data to different teams, while preserving the access only to the specific portion of data each user inside the company would normally have.

The scope of the hands-on incremental security enhancement for the training phase, integrated with the one of data collection, described in the previous paragraph, is limited to guarantee the confidentiality of data in transit and at rest, verify the integrity and the good origin of the dataset, before performing the training. During the training, in fact, the dataset does not benefit of confidentiality, sitting in clear memory without involving SGX in any way.

After the training is performed, the outcome model is then encrypted, again using SGX, and is ready to be archived and deployed on secure machines to perform inferences.

For the experiment, we took a dataset stored as a Comma Separated Values flat file with a size of 26.2 MB. In the normal setup, this file is in clear, read by a Python script that reads it with Pandas and then Keras and TensorFlow to split it into training, validation, and test sets, to train a model.

We added an initial step of decrypting and authenticating the dataset which was encrypted with a pre-shared secret key, using the SGX standard AES GCM 128 encryption algorithm. This additional task added a cost in time of around 390ms to complete.

Similarly, we added a final step of encrypting and authenticating the model before the final remote storage. The output model weights and metadata were sized 27.8MB and the task of encrypting and tagging it takes about another 400ms.

The performances were measured on the same machine as per the previous paragraph.

Hands-on experience: Inference time

To protect the confidentiality and integrity of the model, of the input and output data and of the inference process itself, we reviewed a couple of existing alternative Intel SGX backed implementations: Scone TensorFlow Lite and MarbleRun TensorFlow demo by Edgeless Systems.

Having a Kubernetes cluster and Helm are two prerequisites to run Scone demo, while MarbleRun can be run as a standalone instance, requiring the installation of Docker, Gramine and some Python dependencies. Both technologies share the same main principles:

1. The model is encrypted and transferred into the container.
2. After an attestation process (based on a service named Palaemon for Scone and MarbleRun

for Edgeless System) the secrets to decrypt the model are provisioned to the service.

3. The input data query is securely transmitted to the TensorFlow service which emits its prediction output.

We tested the SCONE TensorFlow-Lite demo, which runs inference on the lite version of the Inception-v3 model, an image recognition model that is used to recognize and label objects in an input image.

The increase of computation time for SCONE TensorFlow Lite for inferences was nearly 50% of the original time, passing from 6.4s without SGX and reaching 9.4s when activating SGX.

At the time of writing, no performance data is available for Edgeless Systems implementation.

4.3.3 Findings and discussion

Python, Lua and Java as interpreted and strongly typed languages are massively used in AI/ML frameworks and are not immune to vulnerability-based attacks as they all support and embed a significant amount of C language code lines for their interpreters. For this reason, it is important to put in place all known security DevOps best-practices, coupling as loosely as possible the services to specific software versions so to minimize the effort required to test, validate and deploy updated services running on latest software versions.

The security of AI is a multi-factorial problem with a design space embracing the global pipeline nodes and the specific security threats or security concerns on the data. For each of these steps, one shall define the plausible security threat, existing regulations, assess the availability and usability of hardware shielding techniques and the relative constraint in terms of performance and vendor selection. In this respect, the work-flow may be a critical issue of TEE usage as being processor vendor specific and as reflected in D2.1 remains true.

Novel techniques as evoked in Section 4.2.3.1 are certainly pertaining in automated networking management, as bringing complementary and relevant security properties in any operating contexts (e.g., TEE availability, typically in distributed nodes layouts). The network extracted metadata must be qualified on their privacy or security sensitiveness so that the optimal security design can be drawn. The grand benefits of these techniques are their universal deployment as they are not restricted to specific processor features and their complementarity and orthogonal to all data-driven adversarial techniques.

Usage of Intel SGX in production to perform secure end-to-end data pipeline is becoming a viable possibility. In Section 4.3.2, we implemented a secure SGX dataset pre-processing step that, while imposing a significant performance penalty, leaves enough margin of optimization to comply with scenarios of strict computational constraints, such as high-volume real-time data pipelines.

In Section 4.3.2, we tested a solution to run secure neural network model inferences and the performance penalty is deemed compatible in scenarios where the confidentiality of the model, of the input data and the overall integrity of the inference process are important factors.

In Section 4.3.2, we also wired together the dataset collection and inference phases with the model training one, in order to authenticate and decrypt the input dataset and to encrypt and sign the trained model for later deployment. The only part of the pipeline that still requires a solution is the AI training phase: while newer Intel 10th generation processors provide amounts of EPC memory compatible with training tasks, due to performance needs, some training is only viable on dedicated GPU or TPU hardware, for which today there is no way to ensure an end-to-end confidentiality protection of the data. However, in many use-cases, such as 5G network traffic metadata analysis, there could be no real need to guarantee the confidentiality of the dataset during the model training phase, for different reasons: for example, the training could happen on local on-premise hardware, where necessary security standards are in place, or the security mitigation practices described in Section 4.3.2 can be considered sufficient.

5 Specification of the enablers' APIs

The T3.3 enablers will provide APIs to integrate with the INSPIRE-5Gplus architecture as well as among each other. The general API rationale is to integrate the entities in a simple yet capable way to realize the planned demonstrations. In the following part, API specifications are listed for T3.3 enablers.

5.1 Smart Traffic Analysis (STA)

The systems provide one main API for managing and configuring the STA. Additionally, based on configuration, a detection triggered events report will be generated with the results of the ML analytics results.

The Management STA API interfaces are detailed in Table 4.

Method	URL	Required Data Objects	Returned Data Object
GET	/sta/status/{uuid}	UUID	200, Status JSON
POST	/sta/start/{uuid}	UUID	200, Status JSON
POST	/sta/stop/{uuid}	UUID	200, Status JSON
POST	/sta/restart/{uuid}	UUID	200, Status JSON
POST	/sta/configure/{uuid}	Configuration JSON	200, Configured JSON

Table 4: Smart Traffic Analysis API

They are designed to manage (status, start, stop and restart) and configure the STA. The configuration will include the relevant parameters such as: monitoring interface, reporting interface protocol, ML model and features to use). One example of Configuration JSON is shown in Figure 25.

```
{
  "interface": "eth1",
  "kafka": {
    "enable": true,
    "servers": [
      "127.0.0.1",
      "127.0.0.1:9092"    ],
    "topic": "sta_values" },
  "kibana": {
    "enable": false },
  "model": {
    "name": "sta_crypto.pkl",
    "tstat_features": [
      1,
      12,
      13,
      43,
      12,
      32,
      54     ],
    "tags": [
      "NO Crypto",
      "Crypto"    ]
  }
}
```

Figure 25: Example of Configuration for STA

The STA reports events detections provided by the inference ML engine based on JSON. Two different mechanisms and protocols are supported: REST API and Kafka message. The common JSON format implemented (Figure 26) identify a malicious network flow, univocally by the typical five-tuple: IP addresses (origin and destination), protocol (UDP/TCP), ports origin and destination, and the confidence level assigned to the prediction tag matched.


```
{
   "timestamp"  "2021-12-23T10:05:08.435000+00:00"
   "confidence"  "0.9"
   "flow_id"  "10.0.27.1904432634.192.145.1134431640253908435"
   "ip_d"  "34.192.145.113"
   "ip_o"  "10.0.27.190"
   "port_d"  "44326"
   "port_o"  "443"
   "protocol"  "TCP"
   "tag"  "0"
   "tag_name"  "CRYPTO"
   "time_end"  "1640253909344"
   "time_start"  "1640253908435"
}
```


Figure 26: A JSON example of a STA detection

5.2 MTD control and optimization using AI/ML

OptSFC API is defined following the OpenAPI format. The API file descriptor can be opened with the Swagger editor, which interprets the REST requests and illustrates them with a graphic interface. The OptSFC API file is available in the INSPIRE-5Gplus Github repository:

https://github.com/INSPIRE-5Gplus/i5p-hla-api/blob/main/WP3_OptSFC/WP3_optSFC.yaml

The API is interfaced to the consumer, the MTD controller (MOTDEC), which is registered with an authentication key to guarantee the confidentiality and the integrity of the exchanged data and decision-making commands.

The API also identifies the security agents, which will send threat intelligence, monitoring data and security to OptSFC, for the real-time modelling of the MDP. The security agents are also created with authentication keys, and are identified by their IP address, which can be of IPv4 or IPv6 format. The fields of the consumer and the security agents can be updated with the REST queries, illustrated in Figure 27.

Figure 27: Security agent entity and REST API query

The security alerts are then sent to OptSFC by the security agents using the */attack* POST query, where the attack detected is reported based on its attack type (reconnaissance, DDoS, intrusion, eavesdropping, tampering, or infection), the protocol vector used for the attack (at the various layers of the TCP/IP stack), the resource targeted by the attack and present in the list of resources to be protected provided by MOTDEC (see Figure 28). The MTD operations chosen by the RL agent is then delivered to MOTDEC which will enforce them.

Figure 28: Attack alert and resource models

5.3 Lightweight and space-efficient vehicle authentication enhanced with misbehaviour detection

The API abstractions for misbehaviour detection enabler are listed as follows:

- "POST/configure/{uuid}": Send configuration parameters from the Security Orchestrator to deploy the V2X misbehaviour detector.
- "POST/start": Start the analytics engine (RL-based detection algorithm).
- "POST/initiate": Start the data stream (.csv format) to feed the security data collector.
- "POST/stop": Stop the misbehaving data sources.
- "POST/alert/{uuid}": Upon detection of misbehaviour, send an alert to the End-to-End (E2E) decision engine.

RL-based misbehaviour detection performs security analytics on csv-formatted V2X traces which are streamed via the security data collector. The RESTful commands, such as GET and POST, are used within this enabler's API. At inception, the POST requests invoke a chain of actions, which are starting the analytics engine and starting the data stream to feed the security data collector. Upon detection, the POST request is generated to stop the misbehaviour data source as the countermeasure. Then, an alert is sent to the E2E decision engine through the POST request. The API descriptions are currently under development according to the required integration steps to be followed in Demo1.

5.4 Security Analytics Engine and Security Agent (SA)

The APIs of the Security Analytics Engine and SA implemented by MMT, the Security Monitoring Framework and its MMT-Probes, have been described in D3.2. The Security Analytics Engine integrates an ML-based module that performs anomaly detection in encrypted traffic (presented in Section 3.1.3 of this document). The API provided by this module consists of RESTful POST and GET requests that are used both during the learning and real-time detection phases. The POST request starts the classification process based on the raw network traffic provided by the MMT-Probe. The GET request allows retrieving the results of the classification. When the classification is ready, a matrix of all extracted features is provided. If the results are not yet ready, a 404 value is returned and the request needs to be repeated later. This API can be easily changed to work in a publish-and-subscribe mode.

5.5 AALTO/OULU's DDoS Detector

The API of the AALTO/OULU DDoS Detector consists of a RESTful POST request (See Figure 29) that triggers the DL-based malicious network flows identification process. An alert report is generated containing details on each identified malicious network flow, which include its Flow_ID (in the form sourceIP-destIP-sourcePort-destPort-protocolID) and the prediction confidence level.

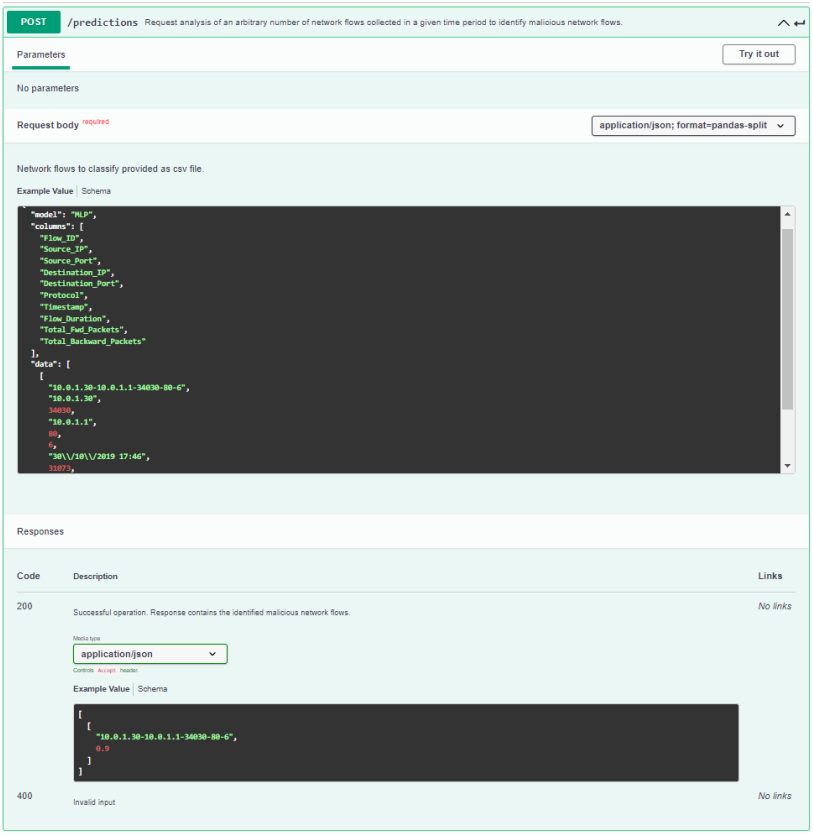

Figure 29: REST API of DDoS Detector

The format of the prediction request and the alert reports are given in Figure 30. The Open-API definition of the DDoS Detector in Yaml form is available in the INSPIRE-5Gplus Github repository (https://github.com/INSPIRE-5Gplus):

Figure 30: Model of prediction request and alert report of the DDoS Detector.

5.6 AALTO/OULU's DDoS Mitigator

The API of the AALTO/OULU DDoS Mitigator consists of a RESTful POST request (See Figure 31) that triggers the DL-based anomaly detection process. The POST request takes a multivariate time series as input and generate an anomaly predictions report containing for each timestep, the computed anomaly score and the decision on whether the values of the VNF's resources usage and performance metrics at that timestep are anomalous or not.

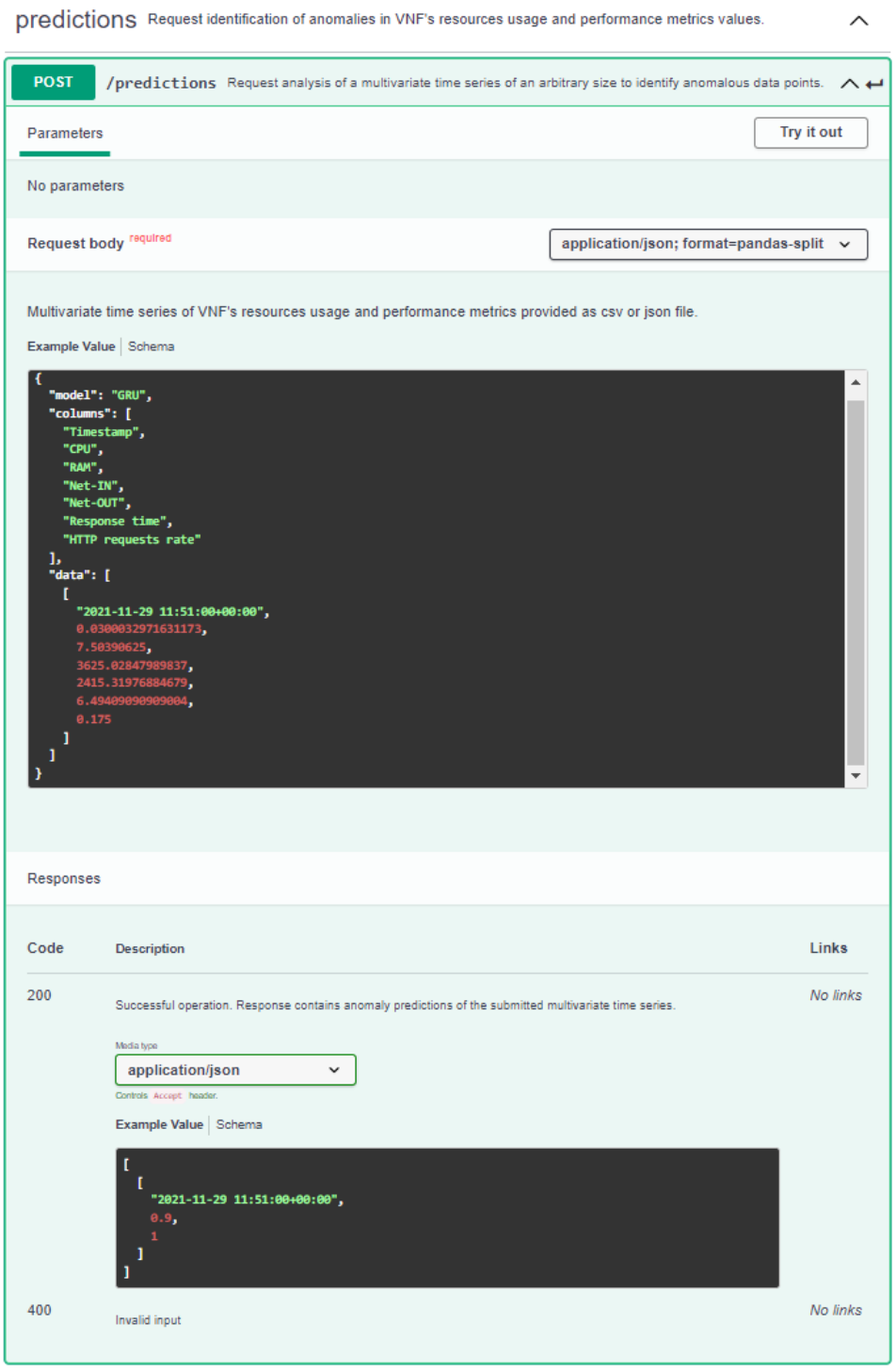

Figure 31: REST API of DDoS Mitigator

The format of the prediction request and the alert report are given in Figure 32. The Open-API definition of the DDoS Mitigator in Yaml form is available in the INSPIRE-5Gplus Github repository:

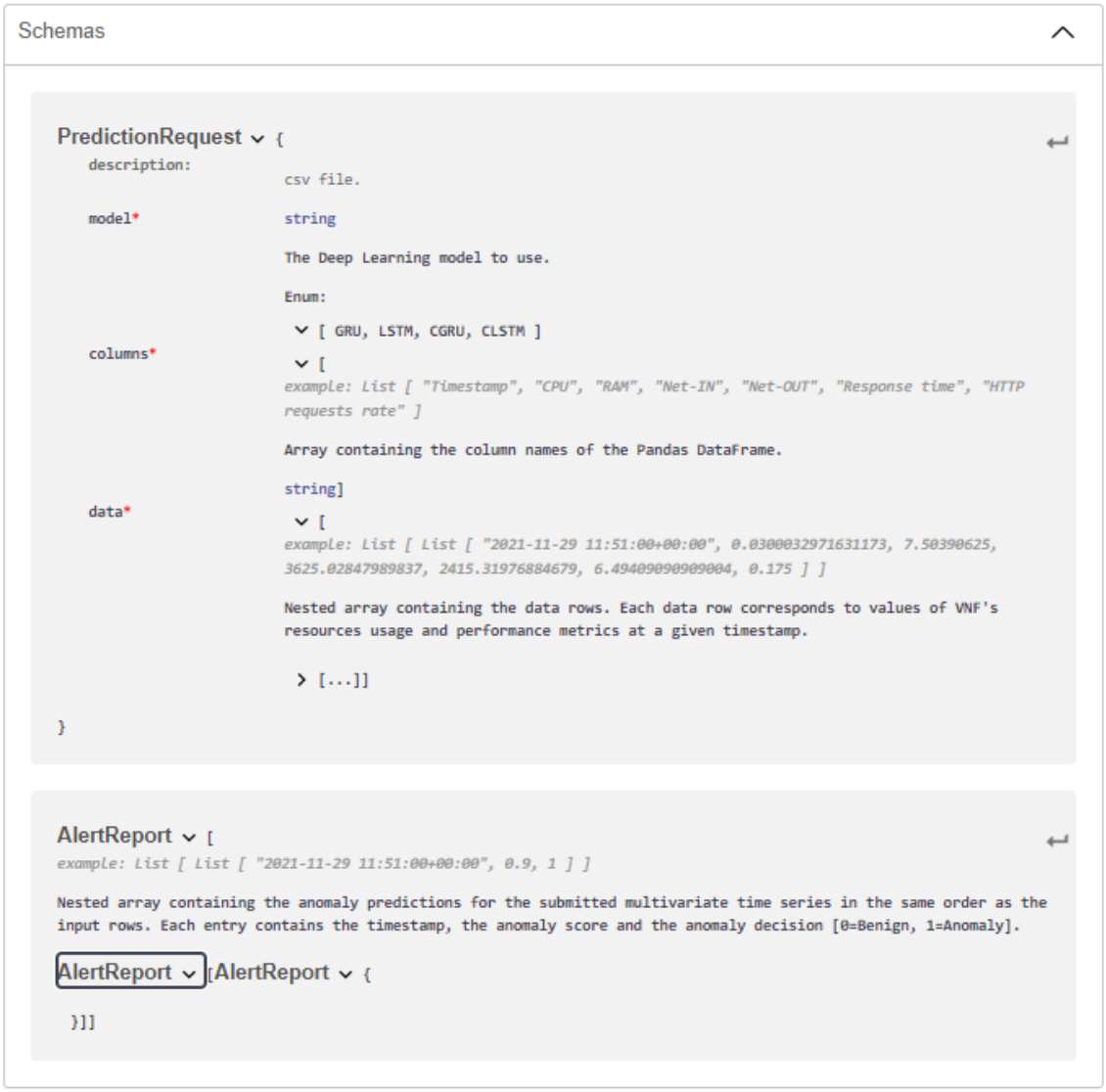

Figure 32: Model of prediction request and alert report of the DDoS Mitigator

5.7 Anti-GPS Spoofing

The API for anti-GPS spoofing is defined following the OpenAPI format. As illustrated in Figure 33, the API includes two interfaces, one for GPS verification and the other for model update. The *modelUpdate* interface is designed to update the spoofing detection model on the edge base station server, requiring the model ID, cellular ID and the update timestamp, and the response is whether the model has been updated successfully or not. The *gpsVerification* interface is used by the UAV with POST method. It takes as inputs the location information, which is an array containing GPS positions and corresponding timestamps, and delivers a report specifying if the location positions are spoofed or not.

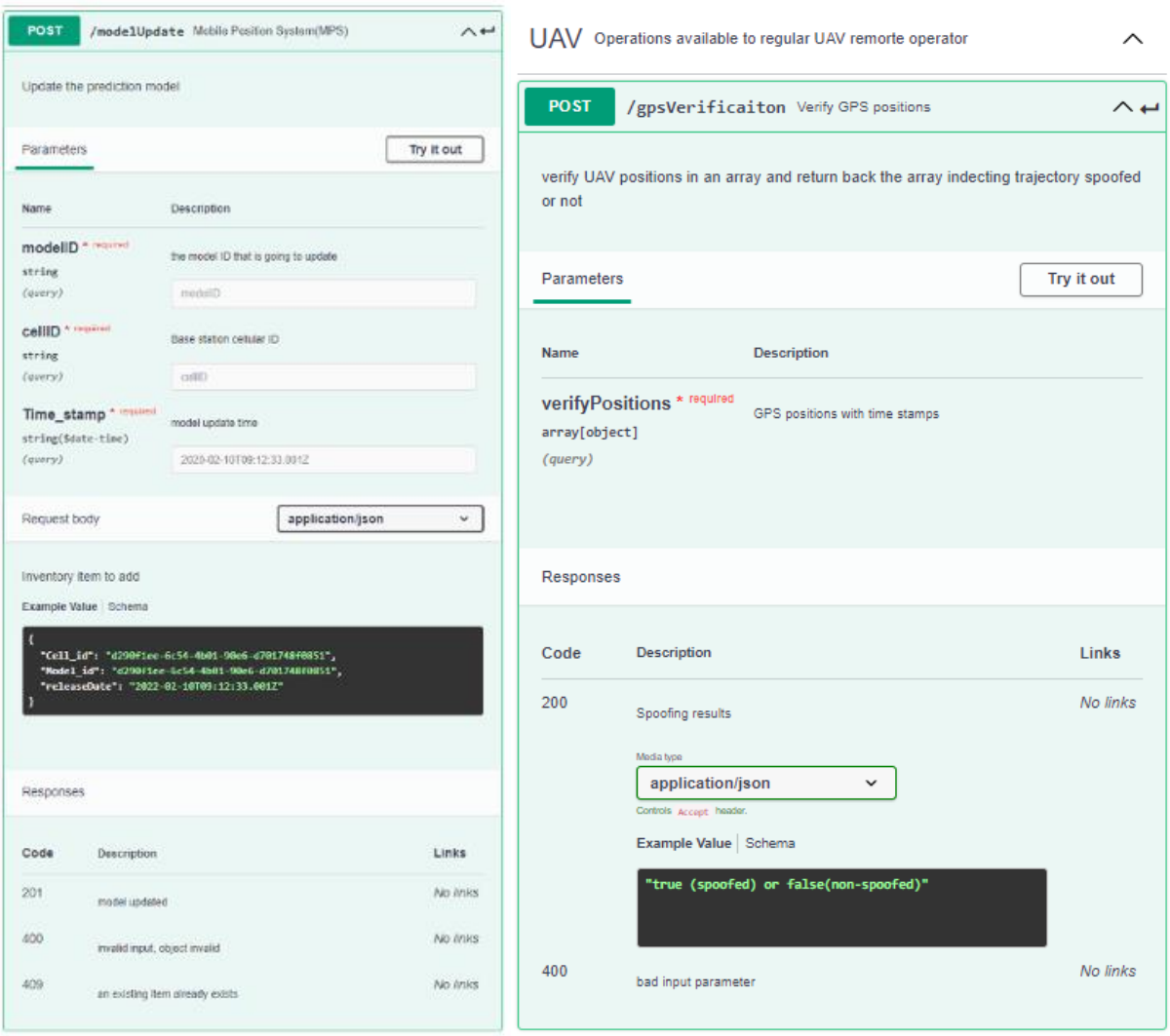

Figure 33: REST API of Anti-GPS Spoofing enabler

6 Demonstration of D3.3 enablers

6.1 Demo1 and INSPIRE-5Gplus AI/ML driven assets

Demo1 aims at showcasing the INSPIRE-5Gplus security closed-loop as well as the instantiation of the High-Level Architecture, across multiple domains and sites interconnected through the Integration Fabric. To that end, the demo showcases the enforcement request of two Security Service Level Agreements (SSLA) across different Security Management Domains (the proactive part of the loop) as well as AI/ML driven attack detections that will trigger automatic security reactions (the reactive part of the loop). The first SSLA will be focused on deploying a 5G service and multiple security features like protecting the communication between the UE access domain and the 5G service domain (e.g., through a backhaul connectivity), as well as other security requirements to avoid security issues (e.g., abnormal behaviour detection in 5G network, DDoS protection). These security requirements will be enforced by deploying different INSPIRE-5Gplus AI/ML driven enablers that will ensure the SSLA compliance.

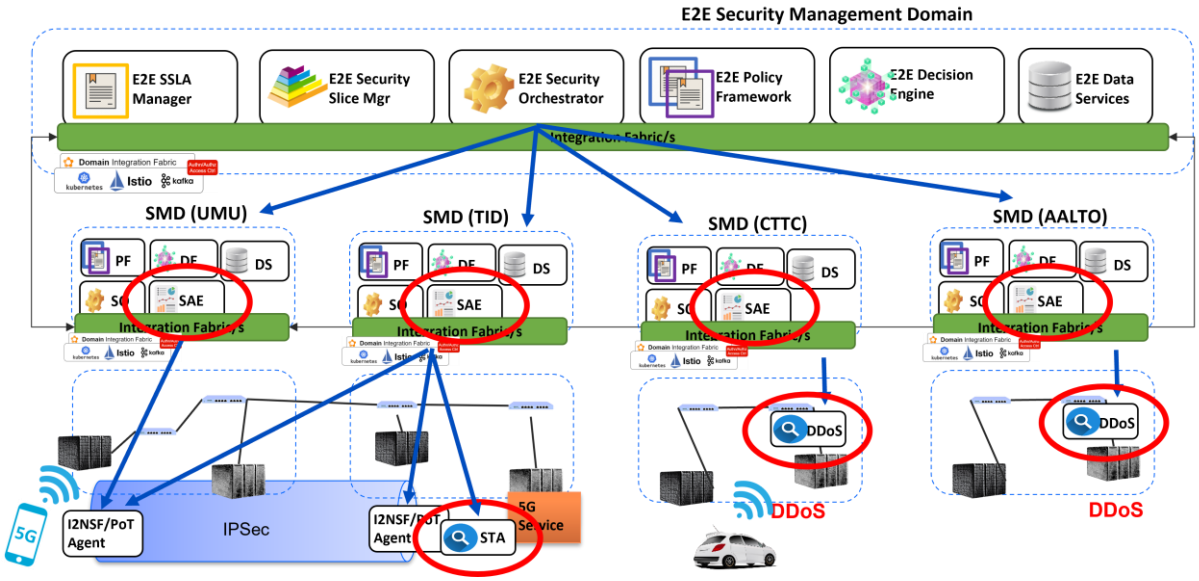

Figure 34: AI/ML driven enablers in Demo1

Figure 34 highlights AI/ML driven security enablers that will be involved in the enforcement of the first SSLA. Below, a brief description of their role in the reactive part of the close-loop is provided.

Security Analytics Engine

The Security Analytics Engine (SAE) is used in the Demo 1 to provide a communication interface between the Security Data Collectors (SDCs), such as STAs or SAs, and the Decision Engine. MI's MMT Security Monitoring Framework is a possible implementation of the Security Analytics Engine and SA. In this demo, the SAE receives alerts from the collector or agents and forwards them to the Decision Engine after analysis. It will also receive instructions from the Security Orchestrator to reinstall, redeploy, or reconfigure the data collectors.

STA

In this demo, the STA is in charge to collect different types of data in one Security Management Domain (SMD) as part of a secure slice deployed to apply analytics. The STA collects relevant network related packets in the 5G Core service. In this case the AI/ML capability provides the identification of

cryptomining activities in one software component concealed over encrypted HTTPS communications, as a Security Analytics Engine component. Once it is detected it reports the information to the domain Decision Engine, to take actions to mitigate the threat. Additionally, this information is provided in parallel to the Trust Reputation Manager to decrease the trustworthiness level of the slice and component.

Misbehaviour detector

In Demo1, a specific implementation of the enabler described in Section 3.1.2 will be performed. In particular, we aim to integrate the misbehaviour detection capabilities of our enabler towards the overarching goal of anomaly detection in Demo1 storyline. Figure 35 illustrates the core components of the enabler, i.e., Security Data Collector, Security Analytics Engine and Decision Engine, and their interactions within the HLA components. Security data collector performs the fusion of V2X network traces that are streamed from the data plane using VMs, in which VMs emulate the representation of vehicles within the RAN. These V2X traces are based on an open-source vehicular anomaly-detection dataset [23]. The incoming streaming vehicular data reports are sequentially analysed within the Security Analytics Engine based on the mobility patterns parameters such as position, velocity, and acceleration, to instruct an RL algorithm for the detection of misbehaviour patterns. The issued security policy is expressed using MSPL. Upon detection of misbehaviour, the detection framework in the decision engine provides the verdict to Security Orchestrator to apply the pre-determined security policy, i.e., misbehaving data source to be isolated, dropped, or blocked.

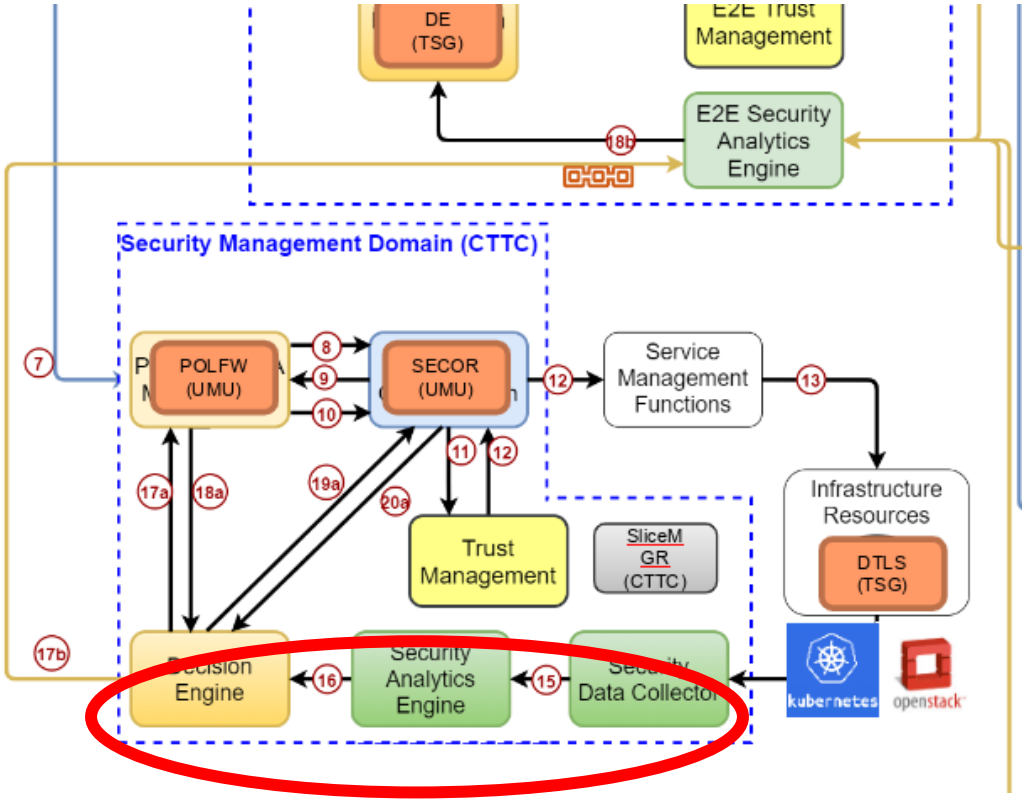

Figure 35: Involved HLA components for misbehaviour detector in Demo1

AALTO DDoS detector

In Demo1, the AALTO's DDoS Detector enabler will be integrated in a closed loop that allows fully automated detection and mitigation of external application-layer DDoS attacks within a slice instance deployed in an SDN environment and offering video streaming service. As illustrated in Figure 36, the

DDoS Detector enabler involves (i) a Security Agent and Security Data Collector for collecting network traffic; (ii) the Security Analytics Engine for extracting network flow features and analysing the extracted features using the DL model to detect specious network pattern; and (iii) the Decision Engine to issue the security policy (e.g., flow dropping) to be enforced by the Security Orchestrator. Note that the issued security policy is expressed using MSPL. In this Demo, we use the SDN controller ONOS to enforce the issued security policy after being translated by the Security Orchestrator into flow command. The attackers will be simulated using different containers launching HTTP DDoS attacks against a video streamer deployed as an NGINX web server.

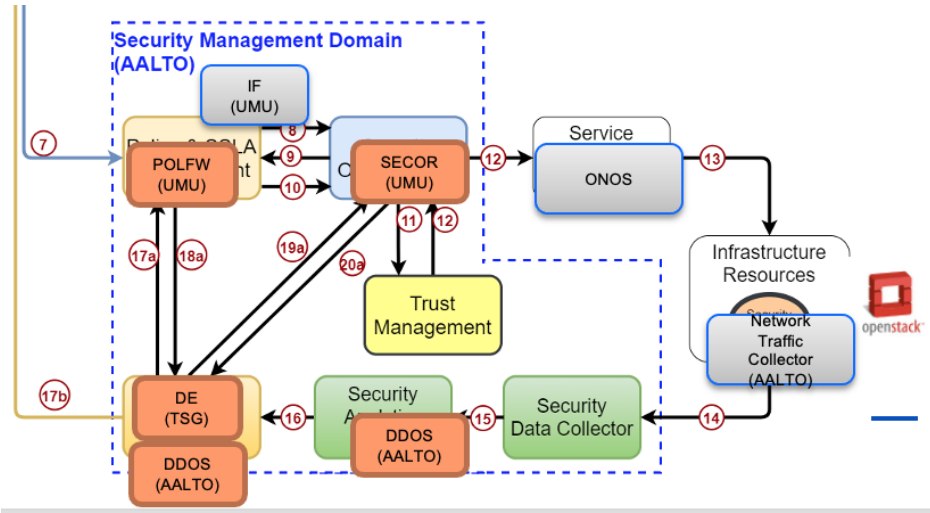

Figure 36: Involved HLA components for DDoS Detector in Demo1

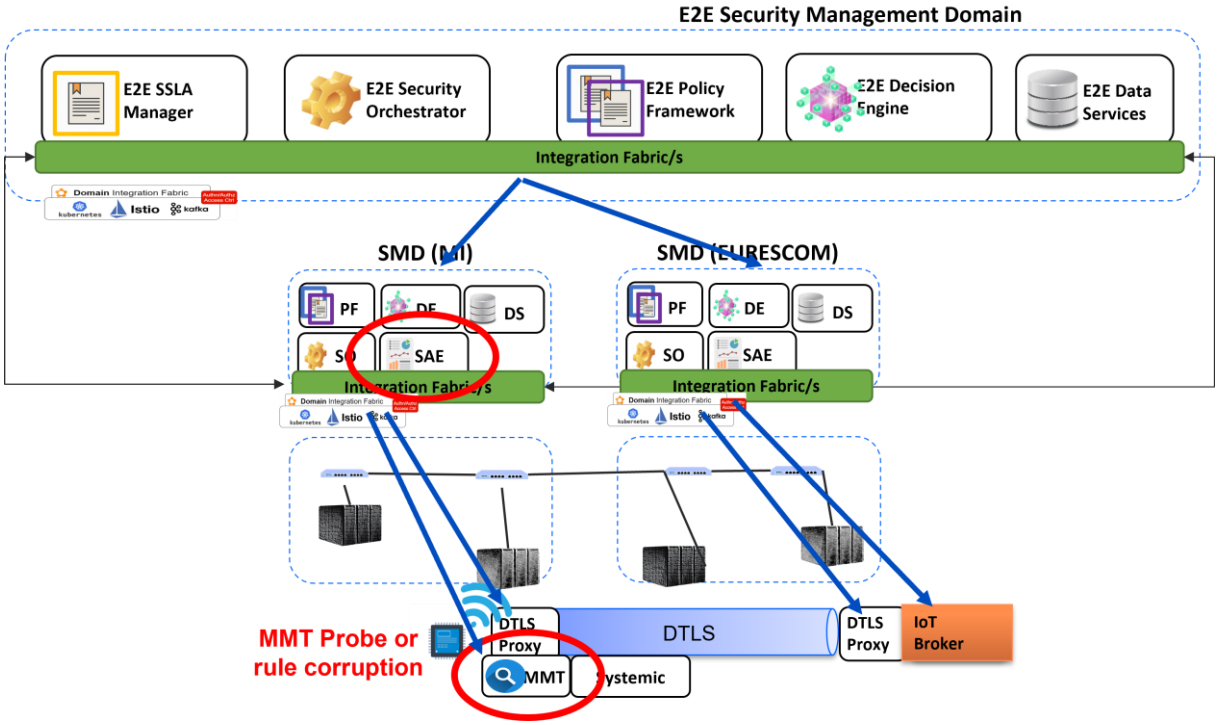

Figure 37: AI/ML driven enablers in Demo1 (SSLA2)

Regarding the second SSLA, it focuses on securing sensor traffic between IoT sensors in remote sites and a global IoT supervision centre - interconnected via a dedicated 5G network slice - by requesting channel protection capabilities between IoT devices and the supervision centre's IoT broker. Figure 37

highlights AI/ML driven security enablers that will be involved in the enforcement of the second SSLA. Following, a brief description of their role in the reactive part of the closed-loop is provided.

MI Monitoring Tool

The MI's Security Monitoring Framework (MMT [93]) is used in the Demo1 to collect different information concerning IoT network traffic and raise alerts when the traffic violates the IoT channel protection SSLA. It is an implementation of the Security Analytics Engine (SAE) and the Security Agents (indicated as MMT in Figure 37). This figure illustrates the disposition of the Security Analytics Engine and its MMT-Probes with respect to the other enablers involved in the Demo1 IoT channel protection. The probe captures and analyses the DTLS cypher suite used in the communications between the IoT devices and the broker. It then issues alerts when the cypher suite corresponds to a lower security level than required, e.g., when attackers might downgrade the cypher suite to a weaker one or even to plain traffic.

The probe is deployed together with Tages' Systemic solution providing a TEE in order to protect it against illegal tampering and modification attacks. Systemic will raise alerts when such internal attacks occur. The alerts will be sent to the Security Analytics Engine, which is in charge of forwarding them to the Decision Engine (DE). The latter may require that the Security Orchestrator (SO) triggers a predefined security policy, such as blocking the malicious IoT device's traffic, redeploying MMT-Probes to a safe location to avoid internal attacks, or reinstalling/redeploying an MMT-Probe to a pristine version as part of the mitigation of internal tampering or modification attacks.

6.2 Demo3 and INSPIRE-5Gplus AI/ML driven assets

Demo3 aims at showcasing an automated closed-loop security application based on MTD mechanism for the proactive and reactive security of network slices and VNFs.

MTD operations are based on the reconfiguration and strategic placement of virtual resources, reducing the attacker's knowledge of the network, and obscuring its attack surface.

Demo3 requires state-of-the-art machine learning in order to automate the security process in a complex and dynamic environment such as 5G. ML is applied in multiple stages of the closed loop: 1**) monitoring and detecting anomalies** and **2) optimizing the decision-making system.**

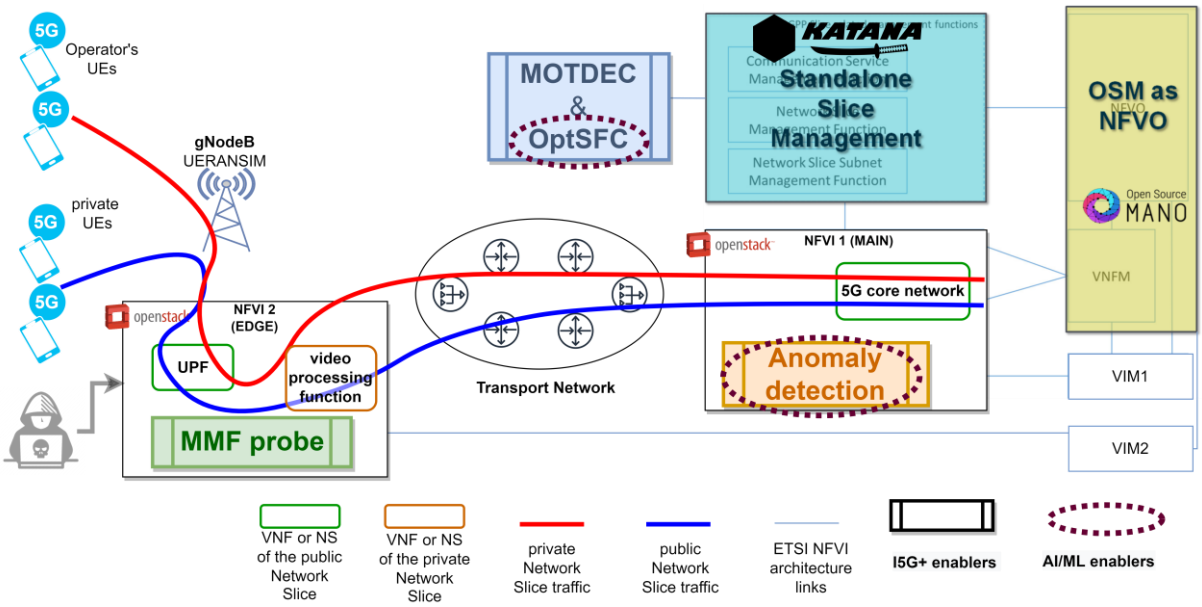

Figure 38: AI/ML enablers in Demo3

These AI/ML requirements are covered by different INSPIRE-5Gplus enablers described in the previous sections. They are in the Demo3 storyline depicted in Figure 38, namely:

Anomaly Detection Framework - ADF (NCSRD)

Demo3 integrates the anomaly detection system described previously in Section 3.1.3. In fact, an important aspect of Demo3 is the joint analysis of heterogeneous data from points of interest within the 5G infrastructure for integrated monitoring. Security Agents will act as distributed probes that will be deployed on-the-fly and adapted to changing requirements and topology. These probes will extract data from packets, flows, system and applications logs that will be subsequently used by the **Anomaly Detection Framework (**ADF) and the MTD mechanism. The ADF will focus on detecting and classifying anomalies associated with security incidents and will inform the MTD enabler for their subsequent mitigation and resolution for protecting the deployed slices.

MTD strategy optimization - OptSFC (ZHAW)

OptSFC is the enabler that learns and optimizes the MTD strategy using RL, as described in Section 3.1.1. OptSFC bases its decision-making on the near-real-time monitoring data collected by the network slice manager, the NFV orchestrator, and the network metrics measured by the MMT-Probes on the edge nodes. Such data is used to model the MDP, formally representing the network run-time state (as described in Section 3.1.1). Moreover, the security alerts of the Anomaly detection system are received to decide on the reactive MTD operation aiming to mitigate the incident.

The MDP modelled by OptSFC is independent of the RL agent used for the training phase and the decision-making. Hence, multiple RL algorithms can be benchmarked to explore which one has the best performances. The RL agent is trained in an off-line 5G testbed before its deployment in production. However, continuous training can also be established during production, adapting the RL model to the dynamics of the network.

MMT-Probe

Here, the MMT-Probe [94] fulfils the role of the Security Agents that will monitor the network traffic and extract the features required by the ML algorithms during the learning and operation phases. Currently, 59 features are provided related to packet and session characteristics (e.g., number and types of packets, protocols and encryption used, duration of sessions, sizes of packets and sessions, times and frequency, and more).

More details for this demonstration will be provided as part of INSPIRE-5Gplus WP5 outcomes.

7 The INSPIRE-5Gplus cognitive security landscape and D3.3 enablers

In this section, we describe the overall picture of D3.3 cognitive enablers in the INSPIRE-5Gplus project and how they are characterized. This description aims to provide an overview on how they utilize AI/ML to provide their functions in terms of particular AI/ML techniques, datasets and mitigated threats. To this end, the following table (Table 5) shows how the enablers rely on AI/ML for their security functions. As shown in that depiction, INSPIRE-5Gplus heavily utilizes AI/ML models, trained using open-source datasets and generated datasets (labelled in Table 5 as "GD") from various scenarios and testbeds developed in the project. Moreover, the Mouseworld enabler plays an important role and provides dataset generation support for cognitive enablers, thanks to its network digital twins infrastructure (Section 3.2.1).

The key ML technologies used in INSPIRE-5Gplus cognitive security functions are RL (Q-learning, A2C, PPO, and DQN), DL (MLP, CNN, SAE, and AE), FL, and useful combinations of these.

The OptSFC enabler learns MTD strategies based on the network state observed, benchmarking various RL algorithms and DL variations (Section 3.1.1). On the other hand, the V2X misbehaviour detector uses RL to identify vehicle misbehaviour by modelling the problem into a sequential decision-making process using MDP (Section 3.1.2).

The Security Analytics Framework (SAF) and the Advanced Encrypted Traffic Analysis (AETA) (respectively, in Section 3.1.3.1 and Section 3.1.3.2) both develop a ML solution combining SAEs and CNNs for advanced fingerprinting and behavioural-based detection of network traffic related to attacks or anomalies, even when encrypted at the networking layer (IPSec) or the application layer (TLS).

The DDoS detector of UMU combines a more conventional GMM algorithm with DL Autoencoders to detect DDoS attacks on multi-domain and multi-tenant NFV environments. At the same time, AALTO/OULU proposes a solution based on MLP and LSTM-AE for DDoS detection in cloud-native networks with auto-scaling capabilities (Section 3.1.4).

The Anti-GPS spoofing enabler uses MLP combined with classical statistical methods such as MVSK, BOX, and WD to predict whether a reported GPS position is legitimate or the result of a spoofing attack (Section 3.1.5). The robust federated learning enabler aims to secure FL deployment against model poisoning performed when learning nodes are partially under the attacker's control (Section 3.3.1). Finally, the TEE leveraged for AI/ML enabler extensively studies and showcases protection of ML models during both training and inference phases using Intel SGX TEE (Section 4.3).

Enabler	Partner	AI/ML relevance	Additional labels	Used dataset(s)
OptSFC	ZHAW	MLP, CNN, RL		GD
V2X misbehaviour detector	CTTC	RL		VeReMi
SAF	NCSRD	CNN, SAE	CTI	NSL-KDD
Advanced encrypted traffic analysis	MI	CNN, SAE	CTI	GD
DDoS detector and DDoS mitigator	UMU	GMM, AE		GD
DDoS detector and DDoS mitigator	AALTO/OULU	MLP, LSTM-AE		CICIDS2017, GD
Anti-GPS spoofing	AALTO/OULU	MLP		GD

Enabler	Partner	AI/ML relevance	Additional labels	Used dataset(s)
Mouseworld	TID		CTI	GD
Robust federated learning	OULU	FL		MNIST
TEE utilization for AI/ML security	TAGES	DL		GD

Table 5: INSPIRE-5Gplus security enablers in D3.3 and their AI/ML aspects

8 Conclusions

As presented in the previous overview section, each D3.3 enabler addresses central challenges relevant to the security of 5G and B5G networks in different perspectives and for various security scenarios.

Services supported by the same infrastructure have requirements and network related behaviours. Added to the fact that communication is encrypted, most of the legacy monitoring tools become insufficient to counter emerging large-scale and zero-day attacks. Moreover, the cloud- native nature of the underlying infrastructure also increases the attack surface of such telecommunication networks, demanding for proactive and resource efficient cognitive security solutions.

This work shows that AI/ML techniques improve the identification and detection of network traffic related to malicious activities, such as Botnet traffic, DDoS attacks, V2X misbehaviour, and GPS spoofing. It optimizes security mechanisms such as MTD, which requires proactive and efficient decision making. In the same security context, ML models should be protected against attacks affecting the training or the inference of the model. This is particularly true when dealing with distributed ML systems as with FL, highly sought after in 5G/B5G networks. The study and work done on improving FL security, as well as the novelty of TEE protection of ML models give promising results in terms of feasibility and performance, highlighting the direction of future research and work.

The enablers presented in this deliverable and achieved by the INSPIRE-5Gplus T3.3, contributes to the main objective of the project, identified as realizing automated security management and orchestration aligned with the ZSM closed-loop specification, covering the complete 5G/B5G infrastructure. The leverage of protected SotA AI/ML is crucial to obtain such automation property in a diverse multi-tenant and multi-domain environment.

In the upcoming INSPIRE-5Gplus WP5 activities, these enablers will be further integrated into actual demonstrations for empirical performance measurements with additional test cases. To this end, any identified further development (e.g., missing API endpoints, performance improvements or missing functionality) will be carried out based on the functional, integration and performance tests.

References

[1] INSPIRE-5Gplus D2.2: Initial Report on Security Use Cases, Enablers and Mechanisms for Liability-aware Trustable Smart 5G Security - https://www.inspire-5gplus.eu/wp-content/uploads/2021/05/i5-d2.2_initial-report-on-security-use-cases-enablers-and-mechanisms-for_v0.14.pdf

[2] Zero-touch network and Service Management (ZSM) V1.1.1 (2021-06); https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/003/01.01.01_60/gs_ZSM003v010101p.pdf

[3] P. Porambage, G. Gür, D. P. Moya Osorio, M. Livanage and M. Ylianttila, "6G Security Challenges and Potential Solutions," 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), 2021, pp. 622-627.

[4] P. Porambage, G. Gür, D. P. M. Osorio, M. Liyanage, A. Gurtov and M. Ylianttila, "The Roadmap to 6G Security and Privacy," IEEE Open Journal of the Communications Society, vol. 2, pp. 1094-1122, 2021, doi: 10.1109/OJCOMS.2021.3078081.

[5] D. P. Bertsekas, Reinforcement Learning and Optimal Control, Athena Scientific, July 2019.

[6] Donald E. Kirk, Optimal Control Theory: An Introduction. Prentice-Hall. p. 55. ISBN 0-13-638098-0, 1970.

[7] Osband, I., Blundell, C., Pritzel, A., and Van Roy, B., "Deep exploration via bootstrapped DQN," Advances in neural information processing systems, 29, 4026-4034, 2016.

[8] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. "Dueling network architectures for deep reinforcement learning," International Conference on Machine Learning (pp. 1995-2003). PMLR, 2016.

[9] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K., "Asynchronous methods for deep reinforcement learning," International Conference on Machine Learning (pp. 1928-1937). PMLR, 2016.

[10] Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S., "Noisy networks for exploration," arXiv preprint arXiv:1706.10295, 2017.

[11] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Azar, M.G., and Silver, D., "Rainbow: Combining improvements in deep reinforcement learning," Thirty-second AAAI Conference on Artificial Intelligence, 2018.

[12] X. Chai, et al., "DQ-MOTAG: Deep Reinforcement Learning-based Moving Target Defense Against DDoS Attacks," 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC), Hong Kong, Hong Kong, 2020 pp. 375-379.

[13] Seunghyun Yoon et al., "Moving target defense for in-vehicle software-defined networking: IP shuffling in network slicing with multiagent deep reinforcement learning," in Proc. SPIE 11413, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II, 114131U (Apr 21 2020).

[14] CVSS (Common Vulnerability Scoring System), https://www.first.org/cvss/

[15] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," Knowledge-Based Systems, vol. 214, p. 106685, 2021.

[16] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez, "Explainable reinforcement learning via reward decomposition," IJCAI/ECAI Workshop on Explainable Artificial Intelligence, 2019.

[17] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," Journal of Machine Learning Research, vol.

22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[18] H. Ye et al., "Machine learning for vehicular networks: Recent advances and application examples," IEEE Vehicular Technology Magazine, vol. 13, no. 2, pp. 94–101, 2018.

[19] C. Kalalas and J. Alonso-Zarate, "Lightweight and Space-efficient Vehicle Authentication based on Cuckoo Filter," 2020 IEEE 3rd 5G World Forum (5GWF), 2020, pp. 139–144.

[20] R. W. van der Heijden, S. Dietzel, T. Leinmuller, and F. Kargl, "Survey on misbehavior detection in cooperative intelligent transportation systems," IEEE Communications Surveys Tutorials, vol. 21, no. 1, pp. 779–811, 2019.

[21] P. Sharma and H. Liu, "A machine-learning-based data-centric misbehavior detection model for internet of vehicles," IEEE Internet of Things Journal, vol. 8, no. 6, pp. 4991–4999, 2021.

[22] S. Gyawali and Y. Qian, "Misbehavior detection using machine learning in vehicular communication networks," 2019 IEEE International Conference on Communications (IEEE ICC), 2019, pp. 1–6.

[23] J. Kamel, M. Wolf, R. W. van der Hei, A. Kaiser, P. Urien, and F. Kargl, "VeReMi extension: A dataset for comparable evaluation of misbehavior detection in VANETs," 2020 IEEE International Conference on Communications (IEEE ICC), 2020, pp. 1–6.

[24] R. Sedar, C. Kalalas, F. Vazquez-Gallego, and J. Alonso-Zarate, "Reinforcement learning-based misbehaviour detection in V2X scenarios," in Proc. of IEEE International Mediterranean Conference on Communications and Networking 2021 (IEEE MeditCom 2021), September 2021.

[25] R. Sutton et al., Reinforcement learning: An introduction. MIT press, 2018.

[26] C. Watkins et al., "Q-learning," Machine Learning, vol. 8, no. 3-4, pp. 279–292, 1992.

[27] VeReMi dataset: https://github.com/josephkamel/VeReMi-Dataset

[28] A. H. Mirza and S. Cosan, "Computer network intrusion detection using sequential LSTM Neural Networks autoencoders," 2018 26th Signal Processing and Communications Applications Conference (SIU), 2018, pp. 1-4.

[29] Canadian Institute for Cybersecurity datasets, http://www.iscx.ca/datasets/

[30] R. U. Khan, X. Zhang, M. Alazab and R. Kumar, "An Improved Convolutional Neural Network Model for Intrusion Detection in Networks," 2019 Cybersecurity and Cyberforensics Conference (CCC), 2019, pp. 74-77.

[31] KDD dataset, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[32] W. Lin, H. Lin, P. Wang, B. Wu and J. Tsai, "Using convolutional neural networks to network intrusion detection for cyber threats," 2018 IEEE International Conference on Applied System Invention (ICASI), 2018, pp. 1107-1110.

[33] S. A. Althubiti, E. M. Jones and K. Roy, "LSTM for Anomaly-Based Network Intrusion Detection," 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), 2018, pp. 1-3.

[34] CIDDS – Coburg Intrusion Detection Datasets, https://www.hs-coburg.de/forschung/forschungsprojekte-oeffentlich/informationstechnologie/cidds-coburg-intrusion-detection-data-sets.html

[35] Gümüşbaş, D., Yıldırım, T., Genovese, A., Scotti, F. "A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems," *IEEE Systems Journal, 2020.*

[36] Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A., "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[37] Andresini, G., Appice, A., Di Mauro, N., Loglisci, C., and Malerba, D., "Multi-channel deep feature

learning for intrusion detection," *IEEE Access*, *8*, 53346-53359.

[38] Andresini, G., Appice, A., Di Mauro, N., Loglisci, C., and Malerba, D., "Exploiting the auto-encoder residual error for intrusion detection," 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (pp. 281-290).

[39] Encrypted Traffic Analysis. November 2019. ENISA. URL: https://www.enisa.europa.eu/publications/encrypted-traffic-analysis

[40] Wang, Pan, et al., "A survey of techniques for mobile service encrypted traffic classification using deep learning," IEEE Access 7 (2019): 54024-54033

[41] Velan, Petr, et al., "A survey of methods for encrypted traffic classification and analysis," International Journal of Network Management 25.5 (2015): 355-374

[42] Kim, Jihyun, et al., "Long short term memory recurrent neural network classifier for intrusion detection," 2016 International Conference on Platform Technology and Service (PlatCon). IEEE, 2016.

[43] Lam, Jordan, and Robert Abbas. "Machine learning based anomaly detection for 5G networks," arXiv preprint arXiv:2003.03474 (2020)

[44] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS Flooding Attack Detection using NOX/OpenFlow," IEEE Local Computer Network Conference, Oct. 2010, pp. 408 – 415.

[45] T. Kohonen, "The self-organizing map," Proceedings of the IEEE, vol. 78, no. 9, pp. 1464 – 1480, 1990.

[46] T. Tang, S. A. R. Zaidi, D. McLernon, L. Mhamdi, and M. Ghogho, "Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks," in Proceedings of the 4th IEEE Conf. on Network Softwarization and Workshops (NetSoft), June 2018, pp. 1 – 5.

[47] S. S. Mohammed, R. Hussain, O. Senko, B. Bimaganbetov, J. Y. Lee, F. Hussain, C. A. Kerrache, E. Barka, and M. Z. A. Bhuiyan, "A New Machine Learning-based Collaborative DDoS Mitigation Mechanism in Software-Defined Network," in Proceedings of WiMob, Oct. 2018.

[48] K. Hong, Y. Kim, H. Choi, and J. Park, "SDN-assisted slow http DDoS attack defense method," IEEE Communications Letters, vol. 22, no. 4, Apr. 2018.

[49] M. Siracusano, S. Shiaeles, and B. V. Ghita, "Detection of LDDoS attacks based on TCP connection parameters," CoRR, vol. abs/1904.01508, 2019.

[50] Ana Serrano Mamolar, Zeeshan Pervez, Jose M. Alcaraz Calero, and Asad Masood Khattak, "Towards the transversal detection of DDoS network attacks in 5G multi-tenant overlay networks," Computers & Security, 79:132 – 147, 2018. ISSN 0167-4048.

[51] Z. Kotulski, T. Nowak, M. Sepczuk, M. Tunia, R. Artych, K. Bocianiak, et al., "On end-to-end approach for slice isolation in 5G networks. Fundamental challenges," in Proceedings of Federated Conf. Comput. Sci. Inf. Syst., pp. 783-792, Sep. 2017.

[52] Apache JMeter tool: https://jmeter.apache.org/

[53] Slowhttptest tool: https://github.com/shekyan/slowhttptest

[54] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in Proceedings of the 4th Int. Conf. on Inform. Syst. Security and Privacy (ICISSP), Jan. 2018.

[55] C. Benzaid, M. Boukhalfa, and T. Taleb, "Robust Self-Protection Against Application-Layer (D)DoS Attacks in SDN Environment," in Proceedings of IEEE WCNC 2020, Seoul, Korea, Apr. 2020.

[56] MOSA!C, "Application-Layer DDoS Dataset," https://www.kaggle.com/wardac/applicationlayer-ddos-dataset, 2019.

[57] R.-F. Fouladi, O. Ermiş, and E. Anarim, "A Novel Approach for Distributed Denial of Service

Defense using Continuous Wavelet Transform and Convolutional Neural Network for Software-Defined Network," Computers & Security, Vol. 112, Jan. 2022.

[58] T. Taleb, P. A. Frangoudis, I. Benkacem, and A. Ksentini, "CDN Slicing over a Multi-Domain Edge Cloud," IEEE/ACM Trans. Mobile Computing., Vol. 19, No. 9, Sep. 2020, pp. 2010 – 2027.

[59] M. L. Psiaki, B. W. O'Hanlon, J. A. Bhatti, D. P. Shepard, and T. E. Humphreys, "GPS spoofing detection via dual-receiver correlation of military signals," IEEE Transactions on Aerospace and Electronic Systems, vol. 49, no. 4, pp. 2250–2267, 2013.

[60] Z. Wu, R. Liu, and H. Cao, "ECDSA-Based Message Authentication Scheme for BeiDou-II Navigation Satellite System," IEEE Transactions on Aerospace and Electronic Systems, vol. 55, no. 4, pp. 1666 – 1682, Aug. 2019.

[61] K.-C. Kwon and D.-S. Shim, "Performance analysis of direct GPS spoofing detection method with ahrs/accelerometer," Sensors, vol. 20, no. 4, p. 954, 2020.

[62] G. Oligeri, S. Sciancalepore, O. A. Ibrahim, and R. Di Pietro, "Drive me not: GPS spoofing detection via cellular network: (architectures, models, and experiments)," in Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks, 2019, pp. 12–22.

[63] Y. Dang, C. Benzaid, Y. Shen, and T. Taleb, "GPS Spoofing Detector with Adaptive Trustable Residence Area for Cellular based-UAVs," in Proceedings of IEEE Globecom, Dec. 2020.

[64] 3GPP, "Enhanced LTE support for aerial vehicles," 3rd Generation Partnership Project (3GPP), Technical report (TR) 36.777, 01 2018. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?

[65] Y. Dang, C. Benzaid, B. Yang, and T. Taleb, "Deep Learning for GPS Spoofing Detection in Cellular-Enabled UAV Systems," in 2021 International Conference on Networking and Network Applications (NaNA) 2021, Nov. 2021.

[66] C. Benzaid and T. Taleb, "AI-driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions," IEEE Network Magazine, vol. 34, no. 2, pp. 186 – 194, March/April2020.

[67] Zhou, Cheng, et al., "Digital Twin Network: Concepts and Reference Architecture," Internet-Draft Draft-Zhou-Nmrg-Digitaltwin-Network-Concepts-04. Internet Engineering Task Force. Work in Progress. 2021. Available online: https://datatracker. ietf. org/doc/html/draft-zhou-nmrg-digitaltwin-networkconcepts-04 (accessed on 1 September 2021), 2021.

[68] TCP STatistic and Analysis Tool tstat software: tstat.polito.it

[69] Mothukuri, V., Parizi, R.M., Pouriyeh, S., Huang, Y., Dehghantanha, A. and Srivastava, G., "A survey on security and privacy of federated learning," Future Generation Computer Systems, 115, pp.619-640, 2021.

[70] Tan, J., Liang, Y.C., Luong, N.C. and Niyato, D., "Toward smart security enhancement of federated learning networks," IEEE Network, 35(1), pp.340-347, 2020.

[71] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine Learning with Adversaries: Byzantine tolerant Gradient Descent," in Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 118–128.

[72] R. Guerraoui, S. Rouaultet al., "The Hidden Vulnerability of Distributed Learning in Byzantium," International Conference on Machine Learning. PMLR, 2018, pp. 3521–3530.

[73] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust Distributed Learning: Towards Optimal Statistical Rates," International Conference on Machine Learning. PMLR, 2018, pp. 5650–5659.

[74] MNIST database, http://yann.lecun.com/exdb/mnist/

[75] C. Benzaid and T. Taleb, "ZSM Security: Threat Surface and Best Practices," IEEE Network

Magazine, Vol. 34, No. 3, Jun. 2020, pp. 124 - 133.

[76] C. Benzaid and T. Taleb, “AI for Beyond 5G Networks: A Cyber-Security Defense or Offense Enabler?”, IEEE Network Magazine, Vol. 34, No. 6, Nov. 2020, pp. 140 - 147.

[77] Goodfellow, I.J., Shlens, J., and Szegedy, C., “Explaining and harnessing adversarial examples,” International Conference on Learning Representations, 2015.

[78] Kurakin, A., Goodfellow, I.J., and Bengio, S., “Adversarial examples in the physical world,” International Conference on Learning Representations, 2017.

[79] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., and Swami, A., “The limitations of deep learning in adversarial settings,” in Proceedings of the 1st IEEE European Symposium on Security and Privacy, pp. 372– 387, 2016b.

[80] Cleverhans, https://github.com/tensorflow/cleverhans

[81] Heinrich et al., “You fool me once, shame on you. You fool me twice, shame on me. A Taxonomy of Attack and Defense Patterns for AI Security,” In Proceedings of the 28th European Conference on Information Systems (ECIS), An Online AIS Conference, June 15-17, 2020.

[82] Biggio et al., “Wild patterns: Ten years after the rise of adversarial machine learning,” Pattern Recognition. 2018;84. doi:10.1016/j.patcog.2018.07.023

[83] Goodfellow et al., “Making machine learning robust against adversarial inputs,” Communications of the ACM. 2018;61(7). doi:10.1145/3134599

[84] Biggio et al., “Evasion attacks against machine learning at test time,” Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol 8190 LNAI, 2017.

[85] Biggio et al., “Poisoning attacks against support vector machines,” In Proceedings of the 29th International Conference on Machine Learning, ICML 2012. Vol 2, 2012.

[86] Adversarial Robustness Toolbox: https://github.com/Trusted-AI/adversarial-robustness-toolbox

[87] Ohrimeko et al., “Oblivious Multi-Party Machine Learning on Trusted Processors,” 25th USENIX Security Symposium (USENIX Security 16). Published online 2016.

[88] Hynes et al., “Efficient Deep Learning on Multi-Source Private Data,” Berkeley University. 2017

[89] Tramèr et al., “Slalom: Fast, verifiable and private execution of neural networks in trusted hardware,” 7th International Conference on Learning Representations (ICLR 2019), 2019.

[90] Roland Kunkel, Do Le Quoc, Franz Gregor, Sergei Arnautov, Pramod Bhatotia, and Christof Fetzer, “TensorSCONE: A Secure TensorFlow Framework using Intel SGX”, https://arxiv.org/abs/1902.04413

[91] Volos et al., “Graviton: Trusted Execution Environments on GPUs,” 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), 2018.

[92] EGo SDK, https://github.com/edgelesssys/ego/

[93] MMT, https://github.com/Montimage

[94] MMT-Probe, https://github.com/Montimage/mmt-probe

Appendix A Software security and AI/ML implementations

A.1 Adversarial machine learning

This table provides a detailed overview of adversarial machine learning modes, objectives and mitigation techniques. In the following table, we refer to white box conditions to designate the situation where the adversary gets to unaltered training data, feature values of each sample, the algorithm and finally the parameters of the trained model. Conversely, black box conditions refer to the situation where the adversary has no access or knowledge over each of these elements.

Attack Mode name	Principle, Objective, and targeted AI/ML types.	Realization, implementation	Mitigation and limits
Model stealing in white box conditions	*Collect the model elements in cleartext on the execution memory or cold storage.* *Get the model parameters (and associated used algorithm).* *Preferred AI/ML targets: The model is easier to access in Federated or any other distributed AI schemes, where the model is present in locations controlled by the adversary.*	*Memory introspection, File access violation*	*Confidential computing as offered by TEE (e.g., Intel's SGX), memory segregation techniques (from VM or middleware, model obfuscation, Strict file access control management.)* *The mitigation limits are the incurred performance costs and the technical implications (e.g., SGX enclave leverage conditions.)* *One can view all mitigations as turning white box conditions into black box conditions.*
Model stealing in black box conditions	*Brute force probing to re-construct an equivalent similar model.* *Preferred AI/ML targets: MaaS used by licensed users.*	*With no possible access to the model, the adversary probes the model to infer the missing data (model parameters).* *A different path is to divert the training data (during the training stage which can be facilitated by regular retraining schemes) for*	*Detection of brute force probing (identification of the request sender and request pattern monitoring).* *Over-complexify the model to decrease the return on investment of the model re-construction.* *Protection of the training data transfer software and*

Attack Mode name	Principle, Objective, and targeted AI/ML types.	Realization, implementation	Mitigation and limits
		elaborating a "similar" model, leveraging the transferability principle.	*transport.*
Membership Inference (MI)	*Principle: In all ML, training data show the highest score compared to unknown sample data.* *Preferred AI/ML targets:* *Overfitted MaaS operated mode. Inclination to Security or Privacy-sensitive training data.*	*Create a Membership inference network interfaced with the attacked network and capable to classify if a probing data is part of the training data of the target model (based on classification scores).* *Conditions: Prior knowledge of the algorithm in use. Overfitted networks are easier to MI attack.*	*Avoid model overfitting by Dropout (fitted for deep neural networks only), consisting in edge removal-cleansing.* *Avoid model overfitting by model Stacking (fitted for any other ML types), based on Ensemble learning (multiple and hierarchical ML)*
Training data poisoning	*Contamination of the training data to mis-lead the classifier-predictor.* *Preferred AI/ML targets: On-the-fly retrained models are exposed.*	*A corpus of publications and associated research based on statistics and data analytics, spans over different types of poisoning (e.g., targeted Clean label, poisoning GAN, label flipping and regression attacks) which all elaborate ad hoc poisoning samples to degrade classifier accuracy.* *Elaborated poisoning attacks can be referred as Backdoor attacks where the attackers carve "triggers" which can be sticked on any clean sample for its poisoning. More advanced research brew easier to operate triggerless backdoors,*	*Poison detection is probabilistic, not a certain process. Each type of poisoning attacks has its own attack-bound defence (i.e., Deep-kNN, Certified Defense, label sanitization and linear regression iterative residual extraction. Recent works develop generic and attack-type agnostic defences.*

Attack Mode name	Principle, Objective, and targeted AI/ML types.	Realization, implementation	Mitigation and limits
		exploiting neurons dropped out (statistically) to prevent overfitting.	
Evasion	*Construct test samples which mislead the classifier-predictor.* *Preferred AI/ML targets: Evasion attacks are not specific and apply to any AI/ML. They are however greatly eased and more efficient with a model that exposes inner working elements. However, pure black box attacks on MaaS are also possible.*	*Another corpus of publications details most-used evasion attacks. Evasion attacks are specific and trimmed to classifier type (e.g., linear, convex-inducing, SVM).* *Evasion attacks efficiency is related to the degree of knowledge of the attacker on the feature space, the model (learning classifier and parameters), the decision function or the label outputs on adversary samples. Hence, gradient-based or surrogate-type attacks require the model gradients (white box or by reconstruction), the confidence score attacks which estimate the model gradients by analysing the scores and the less accurate hard label attacks (e.g., boundary attack) which only needs the label outputs.*	*Empirical defences, inspired by experience and carving successful attack vectors. Adversarial training retrains the attacked model with the successful attack vectors with the correct label. This arm race is endless and leads to efficiency degradation.* *Ensemble, Cascade, Robust Optimization and Spectral normal methods have been developed with respect to the quality of the adversarial sample sets, the maintenance of the classifier efficiency and performance.* *Gradient masking and gradient obfuscation are not an Adversarial Training defence per se and frustrates the generation of stealthy vectors by concealing the model gradients.* *Input cleansing-denoising are statistics-based methods removing adversary-generated noise. Sometime fed downhill, detection methods infer adversarial vectors by analysing the classification before and after denoising. Else, detection methods are autonomous and use raw statistics from various sampling probes over pipeline. Both input denoising and detection methods can be implemented seamlessly on pre-running models.* *Null Class is a specific defence for outliers (samples located out of the normal data*

Attack Mode name	Principle, Objective, and targeted AI/ML types.	Realization, implementation	Mitigation and limits
			distribution bounds) to short cut the classification into the null class thus avoiding mis-prediction.

Table 6: Adversarial machine learning

A.2 Software security techniques for protecting AI/ML software

This table summarizes the security contributions brought by software security in AI/ML solutions.

Attack description	Software security mitigation. Implied techniques.
Evasion attack through model page tampering	Encryption of the model and decryption and processing in token provisioned trusted platforms. Placement of the model in confidential computing environment (Trusted Execution Environment)
Evasion attack by model analysis for attack vector	Encryption of the model and decryption and processing in token provisioned trusted platforms. Placement of the model in confidential computing environment (Trusted Execution Environment)
Evasion attack by model inference code tampering	Inference code integrity verification. Software Integrity enforcement (Trusted Execution Environment)
Poisoning by malicious source-node (training sample) feed.	Source-node authentication by identification label singularization and transmission.
Poisoning by interception and modification of training sample in transit.	Content authentication check prior processing of incoming training data. On both side (emitter and receiver), the ad hoc primitive-routine insertion.
Poisoning by modification of training data at generation and at receipt (model location) by memory introspection	Placement of the exposed software in confidential computing environment (Trusted Execution Environment).
Poisoning by modification of training data at generation and at receipt (model location) by means of software modification (noise generation routine)	Software run-time integrity verification. Establishment of a chain of trust between interacting components using these integrity reports.
Tampering of distributed model algorithms or training data generation modules in distributed or federated layouts	Singularization for identification, authentication and integrity remote verification. Centralized execution monitoring and control of each deployed process.

Table 7: Software security in AI/ML solutions

[end of document]