



Grant Agreement No.: 101096342
Call: HORIZON-JU-SNS-2022
Topic: HORIZON-JU-SNS-2022-STREAM-B-01-04
Type of action: HORIZON-JU-RIA



Holistic, omnipresent, resilient services
for future 6G wireless and computing ecosystems

D3.1 – HORSE Platform Intelligence developed (IT-1)

Work package	WP 3
Task	Tasks 3.1, task 3.2, task 3.3, task 3.4 and task 3.5.
Due date	31/01/2024
Submission date	23/01/2024
Deliverable lead	TID
Version	1.0
Authors	Jose Manuel Manjón (TID), Juan Tamboleo (UMU), Fabrizio Granelli (CNIT), Malak Qaisi (CNIT), Eva Rodríguez (UPC), Panagiotis Gkonis (NKUA), Panagiotis Kapsalis (MAR), Vito Chiancini (MAR), Orazio Toscano (ETI) and Stefanos Venios (SUITE5)
Reviewers	Panagiotis Gkonis (NKUA) and Vito Cianchini (MAR)

Abstract	This deliverable presents the development of the modules, in terms of software descriptions and technical details, of the Platform Intelligence.
Keywords	Development, implementation, configuration, installation.

DOCUMENT REVISION HISTORY

Version	Date	Description of change	List of contributor(s)
V 0.1	22/09/2023	Table of Contents	Jose Manuel Manjón (TID)
V 0.2	03/11/2023	Introduction	Jose Manuel Manjón (TID)
	18/12/2023	Contribution to section 2.1	Jose Manuel Manjón (TID)
	19/12/2023	Contribution to section 2.1	Juan Tamboleo (UMU)
	20/12/2023	Contribution to section 4	Panagiotis Gkonis (NKUA)
	21/12/2023	Contribution to section 2.2	Fabrizio Granelli and Malak Qaisi (CNIT)
	21/12/2023	Contribution to section 5	Stefanos Venio (Suite5)
	21/12/2023	Contribution to section 6	Orazio Toscano (ETI)
	22/12/2023	Contribution to section 3	Eva Rodríguez (UPC)
	V 0.3	10/01/2023	Modifications and comments
V 0.4	17/01/2023	Modifications and comments	All involved partners
V 0.5	18/01/2023	Comments from reviewers	Panagiotis Gkonis (NKUA) and Vito Cianchini (MAR)
V 1.0	23/01/2023	Final reviewed version	All involved partners

Disclaimer

Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the other granting authorities. Neither the European Union nor the granting authority can be held responsible for them.

Copyright notice

© 2023 - 2025 HORSE Consortium

Project co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	OTHER	
Dissemination Level		
PU	Public, fully open, e.g. web	X
SEN	Sensitive, limited under the conditions of the Grant Agreement	
Classified R-UE/ EU-R	EU RESTRICTED under the Commission Decision No2015/ 444	

Classified C-UE/ EU-C	<i>EU CONFIDENTIAL under the Commission Decision No2015/ 444</i>	
Classified S-UE/ EU-S	<i>EU SECRET under the Commission Decision No2015/ 444</i>	

- * *R: Document, report (excluding the periodic and final reports)*
- DEM: Demonstrator, pilot, prototype, plan designs*
- DEC: Websites, patents filing, press & media actions, videos, etc.*
- DATA: Data sets, microdata, etc*
- DMP: Data management plan*
- ETHICS: Deliverables related to ethics issues.*
- SECURITY: Deliverables related to security issues*
- OTHER: Software, technical diagram, algorithms, models, etc.*

Executive summary

This deliverable is focused on the development of the modules that are part of the Platform Intelligence (PII). These modules were described in Deliverable 2.2, where the functionalities of each one of these modules was analysed in the context of the HORSE architecture.

These modules are: Sandboxing (SAN), where are located the Digital Twins (Prediction and Prevention and Impact Analysis) ready to test some scenarios over them; Early Modelling (EM), responsible of providing a preliminary assessment to the SAN by defining policies and rules; Distributed and Trustable AI Engine (DTE), which defines AI data collection, ensuring privacy and implementing productive measures; Policies and Data Governance (PAG), handling the data stored related to the HORSE platform by applying data policies; and Threat Detector and Mitigation Engine (DEME), developing algorithms focusing on network parameters, protocols headers and relevant data for the threat detection and mitigation.

This document goes beyond the architecture of HORSE and details the process of deployment, installation, software development or any other technical related work of the modules included in the Platform Intelligence at IT-1. Therefore, further developments are expected to arise.

These developments and the ones coming from the AI Secure and Trustable Orchestration (STO) module will serve to build a complete infrastructure ready to be integrated in the following months of the project.

For this purpose of integration at this stage of the project Task Forces have been defined along with specific use cases to make the integration process smoother and with the final goal of having a final platform release with all components integrated.

Table of contents

List of figures.....	6
List of tables	8
Abbreviations	9
1 Introduction.....	12
2 Development of the Sandboxing	13
3 Development of the Early Modeling framework	22
4 Development of the Distributed Trustable AI Engine	31
5 Development of the Policies and Data Governance	38
6 Development of the Threat Detector and Mitigation Engine.....	51
7 Conclusions	61
References	62
Appendix A: Meaningful use cases	65
Appendix B. PAG Requirements.....	70
Appendix C. KPI related to the detector performances.....	71

List of figures

Figure 1. Topology created with EVE-NG.....	16
Figure 2. The structure of the Prediction and Prevention Digital Twin (from HORSE D2.2).	19
Figure 3. A block diagram on the deployment of 5G in Comnetsemu	21
Figure 4. Early Modelling components.	22
Figure 5. Threat model.	23
Figure 6. Threat model XML schema.	27
Figure 7. ThreatActor element.	28
Figure 8. Vulnerability element.	27
Figure 9. OrganizationAsset element.	27
Figure 10. Threat element.....	28
Figure 11. CyberAttack element.	28
Figure 12. ControlAction element.	29
Figure 13. DDoS DNS amplification attack XML sample.	30
Figure 14. The internal components of the DTE.	31
Figure 15. The FLOWER concept in DTE.	33
Figure 16. MLFO Architecture.	35
Figure 17. xAI example.....	37
Figure 18. Create New Access Policy.....	39
Figure 19. Check Access Authorization	40
Figure 20. Read Policies List	40
Figure 21. List Authorization Objects	41
Figure 22. Store Anonymisation Config & Perform Anonymisation.....	43
Figure 23. Retrieve Anonymisation Config	44
Figure 24. Create New Retention Policy	45
Figure 25. Enable Retention Policy.....	46
Figure 26. Read Retention Policies List.....	46
Figure 27. MinIO Console.....	49
Figure 28. Create New Asset	49
Figure 29. Read Assets List.....	49
Figure 30. DEME sub-module in the overall architecture [28].	51
Figure 31. HORSE Threat Detector Block Diagram	52
Figure 32. ML-based chained attack detector steps.....	53
Figure 33. STEP1 regressions	54
Figure 34. DEME micro-services	56
Figure 35. Ingress data format	57
Figure 36. HORSE Real-time attack detection web.....	60
Figure 37. Reflection Amplification Attack Scheme	66
Figure 38. SMF / UPF N4	68
Figure 39. Session Deletion Attack.....	69

Figure 40. Injection data into a TCP connection 69

Figure 41. Real case of an NTP amplification DDoS attack 73

Figure 42. Weighted linear combination of concurrent regressions [47]..... 76

Figure 43. Waveforms 76

Figure 44. Standard regression detection time. 77

List of tables

Table 1. Threat meta model elements.....	24
Table 2. DEME APIs.....	33
Table 3. MLFO Comonents.....	36
Table 4. Amplification factors.....	66

Abbreviations

5GTN	5G Test Network
AI	Artificial Intelligence
AMF	Access and Mobility Management Function
API	Application Programming Interface
CDN	Content Delivery Network
DEME	Threat Detection and Mitigation Engine
DDoS	Distributed Denial of Service
DL	Deep Learning
DN	Data Network
DNS	Domain Name Server
DT	Digital Twins
DTE	Distributable AI Engine
Dx.y	Deliverable number y from Work Package x
EM	Early Modelling
EVE-NG	Emulated Virtual Environment for Network Graphing
gNB	gNodeB
HTTP	HyperText Transfer Protocol
I/O	Input/Output
IBI	Intent Based Interface
IP	Internet Protocol
ISP	Internet Service Provider
K8s	Kubernetes
KNE	Kubernetes Network Emulator
KSM	Kube State Metrics

ML	Machine Learning
MLFO	Machine Learning Function Orchestrator
NDT	Network Digital Twin
NF	Network Function
NFV	Network Function Virtualization
NIDD	Network Intrusion Detection Dataset
NTP	Network Time Protocol
NWDAF	5G Network Data Analytics Function
RAN	Radio Access Network
REST	REpresentational State Transfer architectural style
OTEL	Open TELelemetry
PAG	Policies and Data Governance
RAN	Radio Access Network
RTR	Reliability, Trust, and Resilience Provisioning
SAN	Sandboxing
SDN	Software Defined Network
SM	Smart Monitoring
SMF	Session Management Function
SVM	Support Vector Machine
URL	Uniform Resource Locator
UPF	User Plane Function
VNF	Virtualized Network Functions
VRNetLab	Virtual Routers Network Laboratory
WP	Work Package
xAI	eXplainable AI



XML	eXtensible Markup Language
------------	-----------------------------------

1 Introduction

This deliverable introduces the work done in the first iteration of the Work Package 3 components, enclosed on the HORSE Platform Intelligence. As this deliverable is an OTHER type, the information included here is mostly related with software developments of the different modules at this stage of the project.

The document is divided into five main sections, related with the five tasks of this Work Package:

- Section 2: Sandboxing, linked to task 3.1.
- Section 3: Early Modelling framework, linked to task 3.2.
- Section 4: Distributed Trustable AI Engine, linked to task 3.3.
- Section 5: Policies and Data Governance, linked to task 3.4.
- Section 6: Threat Detector and Mitigation Engine, linked to task 3.5.

Section 2 describes the developments made on the Sandbox (SAN), which includes two submodules based on Digital Twins (DT): the Prediction and Prevention DT and the Impact Analysis DT. Those Digital Twins will work together to build a suitable Sandbox to predict and test the modules and topologies of the project.

Section 3 is related to the Early Modelling (EM) component, which is the module in charge of providing the information to the Sandbox. The Early Modelling module includes two blocks: the Taxonomy, which characterizes and profiles the different components; and the Attributes, which define the strategy used to characterize the modules based on the attributes considered.

Section 4 describes the Distributable AI Engine (DTE), module that collects data from various sources and employs AI and machine learning modules to define optimal security policies while preserving privacy. Also, the DTE module takes on data management responsibilities, including pre-training data processing.

Section 5 analyses the Policies and Data Governance (PAG) module serves as the comprehensive hub for ensuring data quality, privacy, integrity, and user-friendly access. It facilitates the flow of data while upholding essential legal and ethical data management principles.

Finally, section 6 provides the development of the Threat Detection and Mitigation Engine (DEME) module, which handles the intricate analysis and processing of network data streams in highly complex and distributed network and infrastructure environments. The algorithms developed will offer an in-depth examination of network parameters, protocol headers, and the extensive data collected from network equipment, devices, and Virtual Network Functions (VNFs).

2 Development of the Sandboxing

The Sandboxing component (SAN) of the HORSE architecture will be used to perform predictions, analyse prevention strategies, and perform impact analysis of different intents and scenarios.

The availability of such a Sandboxing module will allow the HORSE architecture to perform experiments of the Digital Twin of the 6G network, continuously synchronized with the Physical infrastructure of the 6G network and its services. Those experiments are related with cybersecurity threats, like DDoS attack or API vulnerabilities. For more information, check Appendix A.

The first release of the component will focus on how to implement the Digital Twins and how to enable the HORSE architecture to interact with them. For this purpose, both Digital Twins detailed in this section will provide REST APIs and use YAML/XML files for their configuration and information exchange.

Synchronization between the Physical 6G networking architecture and the Digital Twin will be at this stage performed through a configuration file.

2.1 Impact Analysis Digital Twin

The Impact Analysis Digital Twin is responsible for emulating a real network ready to deploy and test the necessary attacks and provide some outputs that can help to take the corresponding decisions on the real network.

2.1.1 Tools

To build the Impact Analysis Digital Twin, a variety of tools will be used in the project which are detailed on the following sections.

Kubernetes

Kubernetes [1] is a powerful open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. It leverages container technology (commonly Docker) to encapsulate applications and their dependencies into isolated units. Containers enable consistent deployment and execution across different computing environments. Regarding the orchestration, Kubernetes simplifies the management of containerized applications by automating tasks such as deployment, scaling, and load balancing. It abstracts the underlying infrastructure, making it easier to deploy and scale applications consistently across various environments.

Kubernetes Network Emulator

The Impact Analysis Digital Twin is based on KNE: Kubernetes Network Emulator. KNE [2] is an Open-Source tool by OpenConfig that allows the deployment of network topologies on Kubernetes pods. For this, KNE emulates images of existing routers, as Cisco, Arista, etcetera.

VR Network Lab

VRNetLab [3] is an open-source project that provides a platform for running and testing virtual routers in a virtualized environment. VRNetLab leverages Docker containers for the deployment of virtual routers by providing a lightweight and efficient way to create, deploy, and manage virtualized applications, including networking devices.

Emulated Virtual Environment for Network Graphing

EVE-NG [4] is a network emulation software that provides a platform for designing, testing, and training network scenarios. The tool enables the emulation of complex network topologies in a virtual environment. Users can design and test network configurations without the need for physical hardware. It supports a variety of networking vendors, including Cisco, Arista, and others. This allows users to emulate devices from different manufacturers in the same topology.

Security Orchestration

BASTIÓN is a policy-based security orchestrator that allows orchestrating and enforcing security policies, considering multiple orchestration algorithms in both, proactive and reactive ways. Security orchestration policies can model different security requirements with different levels of abstraction. Thus, policy models are refined/translated during the orchestration process depending on the abstraction level received. To this aim, the orchestrator implements plugins and drivers that handle policies translation and enforcement. On the one hand, a plugin is a software component that implements the logic for translating security policies to specific technologies. For instance, a plugin could translate security policies derived from what-if scenarios into specific configurations/actions to be applied in the DT (K8s, KNE). On the other hand, a driver is a software component that implements the logic for enforcing configuration/actions across different kind of technologies. For instance, enforcing specific configurations/actions in the DT such as deploying, configuring, and executing a specific attack (e.g., DDoS from Pods) as well as enforcing specific countermeasures using concrete available technologies (e.g., Filter traffic in specific routers).

Finally, the orchestrator is composed of different modules:

- **Orchestration Service:** This module provides the entry point for orchestrating and enforcing security policies. Specifically, it implements different classes that have REST API endpoints that allow enforcing different kinds of security policies. This component is crucial in the NDT context since it makes sure that the security policies needed to analyse different what-if scenarios are applied.
- **Allocation Manager:** It implements the calculation of the orchestration and enforcement plans according to the selected allocation type and allocation algorithms. This selection is performed as the first step according to the available features. For instance, if the infrastructure is NFV-enabled, an NFV-enabled allocation type can be selected. Otherwise, conf-only allocation will be selected. Besides, each allocation type considers several allocation algorithms.
- **Policies Manager:** This module implements different managers that allow managing different policy models. This component will be very useful because it will help to digest the different types of policies that will be received. Since this module will receive information from a variety of components, including Early Modelling, Pre-processing, and Intent-based interface, managing these different inputs will facilitate the deployment of the NDT. Plugins are also part of this module.

- **NFV-MANO manager:** It implements common methods and functions that allow requests for NFV operations independently of the underlying (and available) NFV-MANOs. For instance, operations like deploy, configure, and remove.
- **Assets/Enablers Manager:** It implements a set of drivers that allow the Security Orchestrator to configure Assets/Enablers (using the results of the policy translation).
- **Data Service Manager:** It implements a data services client that allows access to data services in a common way. The specific data service driver is provided by auto-generated code using OpenAPI tools.
- **Enforcement Manager:** It implements the way that the enforcement plan is applied. Different enforcement managers can be developed by including an “enforce” method.

Telemetry

In order to make an assessment of the security policies applied to the infrastructure, gathering and analysing telemetry data is needed. Telemetry data will be composed of metrics of the network, logs from its components, and traces from the different applications. A variety of tools have been considered in order to fulfil this process:

- **Prometheus [5]** is an open-source system monitoring and alerting toolkit. It is mainly used for gathering metrics and events for different networking systems. Thanks to Prometheus, the process of data analysis and the understanding of the performance of the system is facilitated. Prometheus is also known for its scalability and flexibility since it has good integration with containers and cloud services. It will be used to store metrics and will be integrated with the Alertmanager to alert, and it will notify administrators in case of the detection of abnormal values.
- **OTEL (OpenTelemetry) [6]** is an observability framework and toolkit designed to create and manage telemetry data such as traces, metrics, and logs. OpenTelemetry is vendor- and tool-agnostic, meaning that it can be used with a broad variety of observability backends. It is the unification of two other projects that provide knowledge and expertise in this field. OpenTelemetry will help to comply with industry standards as well as make a cleaner monitoring pipeline.
- **Node Exporter [7]** is a component of the Prometheus ecosystem. It is a metric gatherer for Unix systems and is designed to get information about the hardware and the operating system. Node Exporter can gather a variety of metrics, like the use of CPU, memory, network, storage, etc. In addition, Node Exporter has great integration with Prometheus. Using Node Exporter, the assessment required for the what-if scenarios will be done by measuring the different metrics for the hardware and the operating system.
- **KSM (kube-state-metrics) [8]:** this tool helps to get information about the state of the Kubernetes cluster. This component does that using the Kubernetes API and has good integration with the rest of the components since it can send the metrics in a Prometheus format. KSM helps with the monitoring of Kubernetes clusters and gathers information regarding health and performance. The same way Node Exporter will be used for the assessment, KSM will help with information about the cluster.
- **Alertmanager [9]** is another component of the Prometheus ecosystem and takes care of managing alerts generated from Prometheus and sending notifications based on predefined rules.

2.1.2 Deployment

Once the topology is created in the EVE-NG, it is adapted to a format readable by KNE by executing a script that translate the information to one format to another. An example of a topology generated in EVE-NG is described in Figure 1.

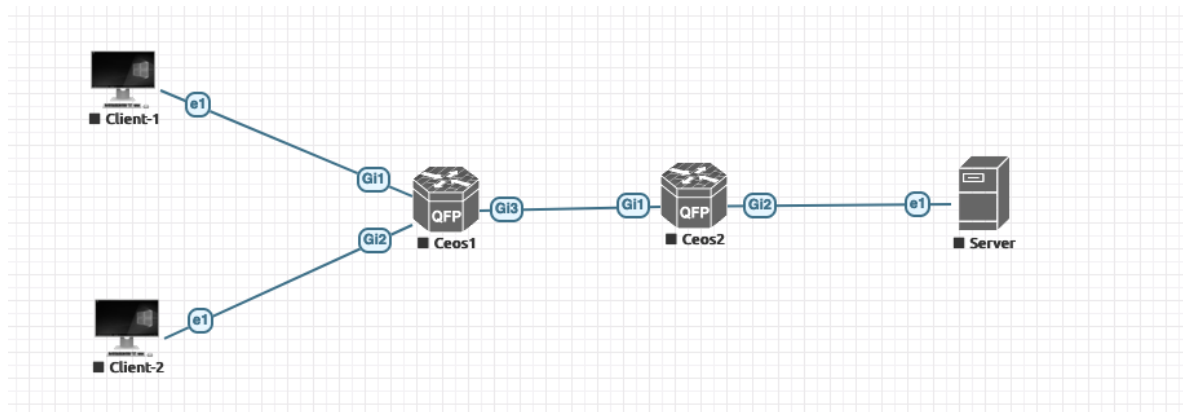


Figure 1. Topology created with EVE-NG.

And the corresponding topology descriptor transformed, that is consumed by KNE to deploy the nodes with its correspondent links:

```
name: horse-example
nodes:
- name: ceos1
  model: ceos
  os: eos
  vendor: ARISTA
  config:
    config_path: /mnt/flash
    config_file: startup-config
    file: r1-config
  interfaces:
    eth1:
      name: Ethernet1
    eth2:
      name: Ethernet2
    eth3:
      name: Ethernet3
```



```
- name: ceos2
  model: ceos
  os: eos
  vendor: ARISTA
  config:
    config_path: /mnt/flash
    config_file: startup-config
    file: r2-config
  interfaces:
    eth1:
      name: Ethernet1
    eth2:
      name: Ethernet2
- name: server
  vendor: HOST
  config:
    config_path: /home/cognet
    config_file: server-config.py
    file: /configuracion/server-config.py
    image: alpine:latest
  interfaces:
    eth1:
      name: Ethernet1
- name: client-1
  vendor: HOST
  config:
    config_path: /home/cognet
    config_file: client-1-config.py
    file: /configuracion/client-1-config.py
    image: alpine:latest
  interfaces:
    eth1:
      name: Ethernet1
```

```
- name: client-2
  vendor: HOST
  config:
    config_path: /home/cognet
    config_file: client-2-config.py
    file: /configuracion/client-2-config.py
    image: alpine:latest
  interfaces:
    eth1:
      name: Ethernet1
  links:
- a_node: ceos2
  a_int: eth1
  z_node: ceos1
  z_int: eth3
- a_node: server
  a_int: eth1
  z_node: ceos2
  z_int: eth2
- a_node: client-1
  a_int: eth1
  z_node: ceos1
  z_int: eth1
- a_node: client-2
  a_int: eth1
  z_node: ceos1
  z_int: eth2
```

With this configuration file, it is feasible to deploy the scenario over the KNE by executing some commands. Once the pods have been created, we have to set on them the network configuration with the IP addresses, routes, etc.

On a first deployment of the Impact Analysis Digital Twin, there will be 10 clients, a gNodeB a transport network made up of four routers, a 5G core and a DNS server. All these elements will be needed to emulate some attacks over it.

2.2 Prediction and Prevention Digital Twin

The Prediction and Prevention Digital Twin is built on the Comnetsemu network emulation software [10], [11]. Comnetsemu is based on the well-known mininet network emulator [12], with the integration of a docker-in-docker environment to enable the deployment of services as docker containers. In this way, it is possible to emulate a 5G SA or NSA architecture by exploiting the available open-source implementations of the 5G core and access networks.

All employed software, including Comnetsemu, is publicly available and open source.

Mininet is a well-recognized Software Defined Networking network emulator. It is characterized by a stable and realistic performance, as demonstrated in [13], as well as some limitations in extremely large emulation scenarios [14]. Comnetsemu builds up on top of such realistic network emulation to enable to deploy actual service containers, thus generating a realistic workload and enabling to build realistic scenarios for current and next-generation networks.

Prediction and Prevention Digital Twin includes the following modules:

- Digital Twin Modelling module: it is responsible for generating the DT based on the input data (traffic and topology information, orchestrated services, etc.)
- Digital Twin Engine module: it will run the DT in the Comnetsemu emulation environment.
- Digital Twin-based Prediction module: it will analyze the output of the DT Engine block using AI/ML algorithms to perform predictions and identify anomalies.
- I/O Interface module: interface with DTE / IBI for receiving requests and providing the related outcomes.

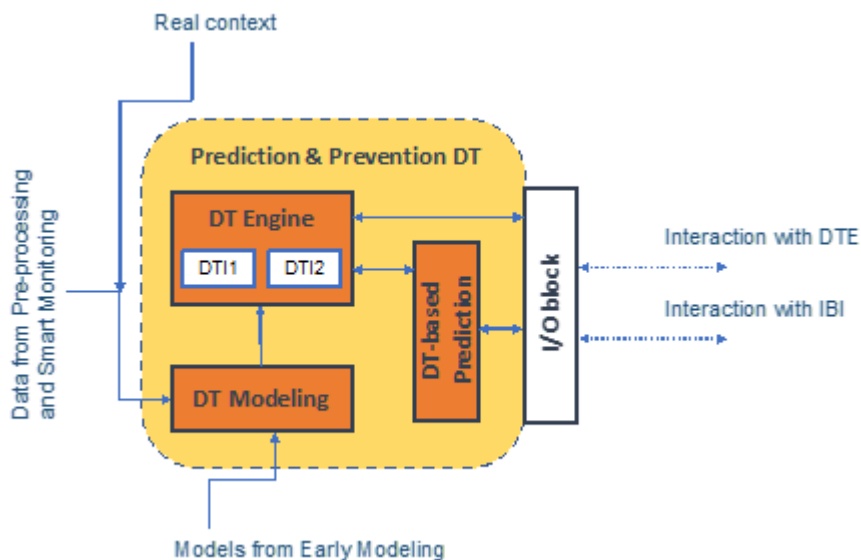


Figure 2. The structure of the Prediction and Prevention Digital Twin (from HORSE D2.2).

The Prediction and Prevention Digital Twin is available to the HORSE platform as a Virtual Machine. Deployment of the VM is performed through Vagrant.

All modules are developed in Python. The following sections describe how the internal modules are developed.

Digital Twin Modelling module

This module generates the script for replicating the Physical Twin of the 6G network into the Digital Twin.

This module receives in input via REST APIs a YAML descriptor of the network topology and the known services running in the network. The format for data collection is common for the entire HORSE sandbox, and it is the same as for the Impact Analysis Digital Twin. An example of the format of the file is as described in Section 2.1.2. Based on such information, it generates a script file to build the network and services in the Comnetsemu environment and to run the Digital Twin in the sandbox.

The following represents an example of a script for deploying a simple topology in mininet or Comnetsemu:

```
from mininet.topo import Topo

class MyFirstTopo( Topo ):
    "Simple topology example."
    def __init__( self ):
        "Create custom topo."
        # Initialize topology
        Topo.__init__( self )
        # Add hosts and switches
        h1 = self.addHost( 'h1' )
        h2 = self.addHost( 'h2' )
        h3 = self.addHost( 'h3' )
        h4 = self.addHost( 'h4' )
        leftSwitch = self.addSwitch( 's1' )
        rightSwitch = self.addSwitch( 's2' )
        # Add links
        self.addLink( h1, leftSwitch )
        self.addLink( h2, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, h3 )
        self.addLink( rightSwitch, h4 )

topos = { 'myfirsttopo': ( lambda: MyFirstTopo() ) }
```

The module receives in input via REST APIs also a YAML/XML descriptor of the attack or scenario to evaluate. This will be translated in a set of commands to deploy additional components in the Digital Twin and run/replicate network traffic.

Digital Twin Engine module

This module implements the Digital Twin. The Digital Twin is built in the Comnetsemu environment, enabling a precise emulation of an SDN network and faithful replication of services by deploying them in docker containers.

As an example, the following figure represents how a simple 5G network with Mobile Edge technology can be replicated in form of a Digital Twin in Comnetsemu.

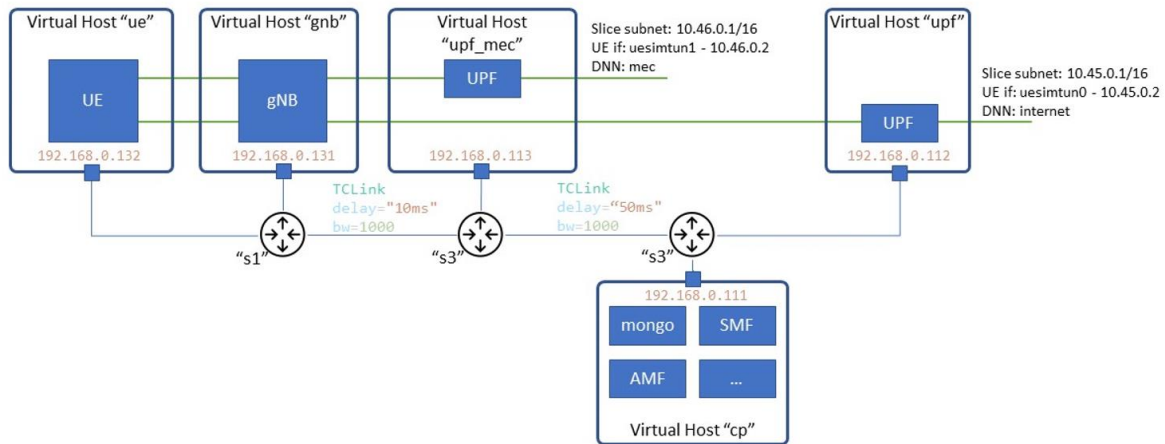


Figure 3. A block diagram on the deployment of 5G in Comnetsemu

Digital Twin-based Prediction module

This module is aimed at predicting relevant scenarios in order to signal potential treats or other performance degradations to the HORSE architecture. In the first implementation, it will be able to detect traffic peaks and potential congestion as well as some types of security attacks.

I/O Interface module

The Digital Twin offers a REST API for interaction with the other modules of the HORSE architecture, as well as for most of the interactions among its internal modules. A Swagger interface is provided to enable fast and efficient testing of the proper operation of all offered functionalities.

The REST APIs are developed using Python FastAPI framework. The exposed ports will be configurable through a proper “config.ini” file in the software distribution.

3 Development of the Early Modeling framework

The early modelling is designed to feed the Sandboxing module with all the required information to successfully perform. It consists of two main components: Taxonomy and Attributes. Taxonomy is responsible for characterizing and profiling the different threats and attacks to be considered in the 6G context within the DT. Attributes define the strategy and set of attributes to characterize the impact of the attack in the 6G components, as well as the impact of the mitigation and preventive strategies in the 6G components. Figure 1 shows the structure of the Early Modelling module.

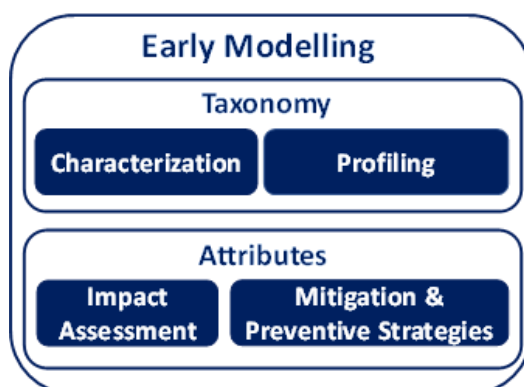


Figure 4. Early Modelling components.

3.1 Taxonomy component

The main objective of the taxonomy component is to define a model for characterizing and profiling the different threats and attacks to be considered in the 6G context within the DT.

In the threat modelling process, the first step is to identify the vulnerabilities in the system/network/application to exploit the resources and disrupt the services. Figure 5 presents the proposed meta-model consisting of all the aforementioned elements.

Attacks can be characterized by the meta-model consisting of the following elements.

- Threat actor
- Vulnerability
- Threat
- Cyber Attack
- Control action
- Information on organization Assets

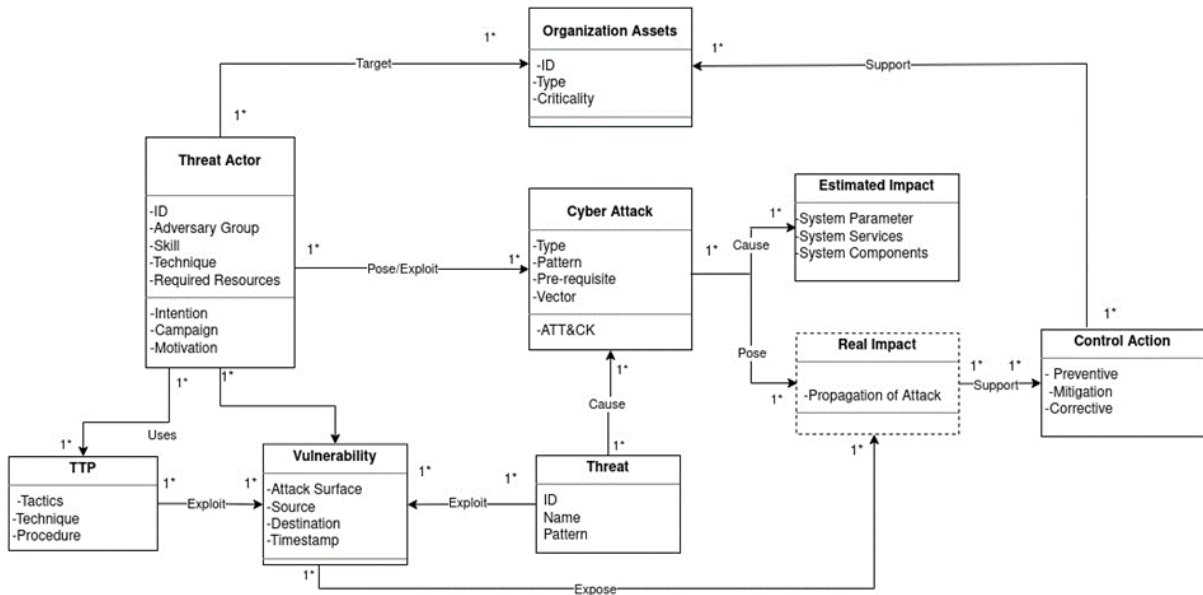


Figure 5. Threat model.

Vulnerability: Vulnerability defines the loophole in a system or vulnerable spot that can be targeted by the adversary. The meta-model makes use of an attack surface to clearly describe the vulnerability. This attack surface includes information on user equipment, network information, and services that can be targeted. While defining the vulnerability in an application/network/user process, there is a need to mention the other properties associated with vulnerabilities such as source, destination, and timestamp.

Organization Assets: Organization assets include the information of assets and other devices that threat actors want to exploit. The proposed threat modelling processes consider the criticality of each asset on a Likert scale. This would help us in defining and scaling the critical assets and devices for the organization.

Threat Actor: A threat Actor is categorized as a malicious actor who has some motivation towards using a system. Including the intention of the threat actor as he wants to steal some information or want to disturb the services or applications provided by the network. The proposed meta model defines the scope of adversary groups for threat actors by categorizing it as internal or external to the system and skill set on a Likert scale.

TTP: TTP is used by the attacker to obtain its operation by including the tactics, techniques, and procedures. TTP can be useful for gathering the cyber threat information related to attack pattern, deployed resources and exploit information.

- Tactics: it defines the goal of the attacker.
- Technique: it includes the software tools and techniques that can be used by the attacker to perform attack.
- Procedure: It includes the set of tactics and techniques to put together to make the procedure. Basically, it includes the step-by-step procedure on how to launch the attack.

Threat: Threats include harmful action facilitated by vulnerability. It can be any weakness from the ENISA threat report, use case activities, adversarial behaviour, and observable patterns.

Cyber Attack: it is a harmful action conducted by the threat actor by exploiting the vulnerability of the system. The following attributes are associated with the cyber-attack.

- **The pattern of the attack:** it can be malware, spyware, or injection.

- **Pre-requisite:** information available on vulnerability.
- **Attack vector:** include the whole mechanism to deploy the attack.

Control Action: Control action includes the course of action and countermeasures associated with an attack. This includes preventive measures, mitigation, and corrective measures.

- **Preventive measure:** policies implemented on the occurrence of a particular event (risk) probability, that are intended to prevent the appropriate actions violating the policies or link with the violation or risk associated with the third party.
- **Mitigation:** Define the mitigation action post-attack occurrence
- **Corrective:** Define the course of action and risk management related to 6G technology, designed to react to the detection of an incident to reduce or eliminate the opportunity for the unwanted event to recur

Table 1 presents describes all the elements in the meta model, as well as its type and scope in the HORSE architecture.

Table 1. Threat meta model elements.

Elements	Attributes	Description	Type	Sources from where
Organization Assets	Id (attribute)	Unique Identifier	String	SM, Pre-processing, PAG
	Type	Identifier of organization Assets	String	
	Criticality	Desired level of security for use case, based on a Likert scale, 5 levels.	Integer	
Threat Actor	ID	Unique Identifier	String	MITRE and 5G Telco network
	Adversary Group	Define the adversary group (internal/External)	String	
	Skill	The specified level of skill, based on a Likert scale, 5 levels (Knowledgeable-no-knowledge)	Integer	
	Technique	Techniques used by the threat actor (e.g. Network Scanning Technique)	String	
	Required Resources	Used of s/w to Identify which service is vulnerable and can be targeted	String	
	Motivation	Define the motivation of the threatActor using a system	String	
	Intension	What threat actor wants to steal	String	

	Campaign	Need to define the threat actor based on the capability, such as their intent, type of password used, observed patterns, behaviour, history, and motives	String	
TTP	Tactics	Define the goal of the attacker (e.g. tactics can be impact)	String	MITRE, 5G Telco network and DEME
	Technique	What tool or technique was used by the attacker (e.g. IP spoofing)	String	
	Procedure	Define step-by-step procedure on how to launch the attack	String	
Cyber Attack	Type	Define type of attack that could be launched in Horse (e.g. DNS amplification)	String	SAN, DEME
	Pattern	Define behaviour or pattern of attack such as Spyware, malware, injection	String	
	Pre-requisite	Connect or reference the Information available on vulnerability	String	
	Vector	Threat vectors define the whole Mechanism to deploy the attack	String	MITRE
	ATT&CK	Describe the ID and type available on MITRE	String	
Threat	ID	Identifier of link	String	Use-cases
	Name	Define a threat which is facilitated by vulnerability	String	
	Pattern	To identify weaknesses patterns can be (1: State of the art, 2: ENISA report, 3: Adversary Behaviour, 4: use-case threat activities, 5: specific observable pattern)	String	
Vulnerability	Attack Surface (User Equipment)	Define the User equipment (e.g. Malicious Bot (CCTV, HoLoLens headset etc.))	String	SM
	Attack Surface (Network)	Define Network (e.g. RAN, CN (core network), EN (Edge network))	String	

	Attack Surface (Services)	Define 5G/6G Services (e.g. Cloud, Roaming, third party, third-party)	String	
	Source	Define the sources of vulnerability (S/W, network, website, user process, application, configuration, or third-party vendor)	String	
	Destination	Target of vulnerability	String	
	Timestamp		date time	
Estimated Impact	Performance Parameter	Define the performance parameter.	String	SAN
	System services	Define 5G/6G services (e.g. Roaming, Multi-media, cloud , third party)	String	
	System Components	Define 5G/6G Components (e.g. RAN core convergence, Core-network, edge network, user, terminals etc.	String	
Real Impact	Attack Propagation	Define the attack propagation and cascading effect link with threat actor penetration, manipulation and severity of an attack	XML element	
Control Action	Preventive	Define policies implemented on the occurrence of a particular event (risk) probability, that are intended to prevent the appropriate actions violating the policies or link with the violation or risk associated with the third party.	XML element	PAG, RTR
	Mitigation	Define the mitigation action post attack occurrence	XML element	
	Corrective	Define the course of action and risk management related to 6G technology, designed to react to the detection of an incident to reduce or eliminate the opportunity for the unwanted event to recur.	XML element	

3.2 Threat meta model XML Schema

An XML schema has been defined for the meta-model to enable the representation of threats. This XML schema consists of the root element threatModel, which consists of a sequence of

intermediate elements threatModelElement. The threatModelElement has an ID to uniquely define each model element and a sequence consisting of all the elements in the Meta model, as shown in Figure 6.

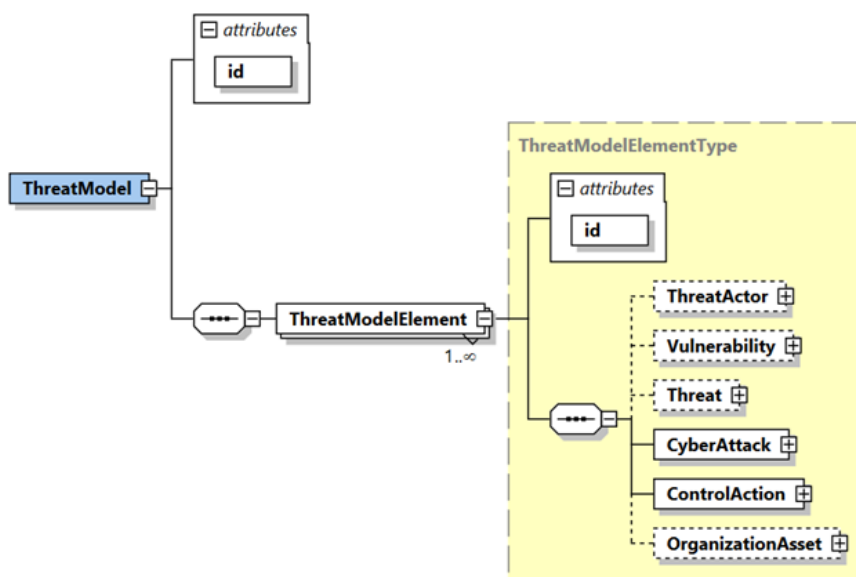


Figure 6. Threat model XML schema.

Figures 7 to 12 show the XML representation for all the elements in the threat meta model.

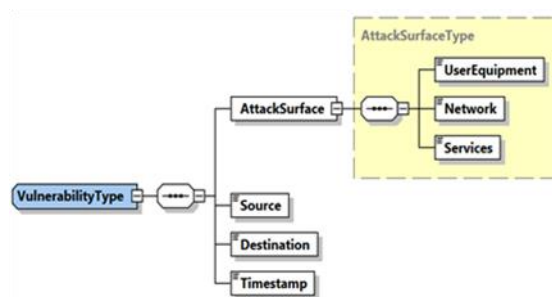


Figure 8. Vulnerability element.

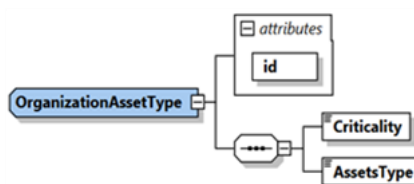


Figure 9. OrganizationAsset element.

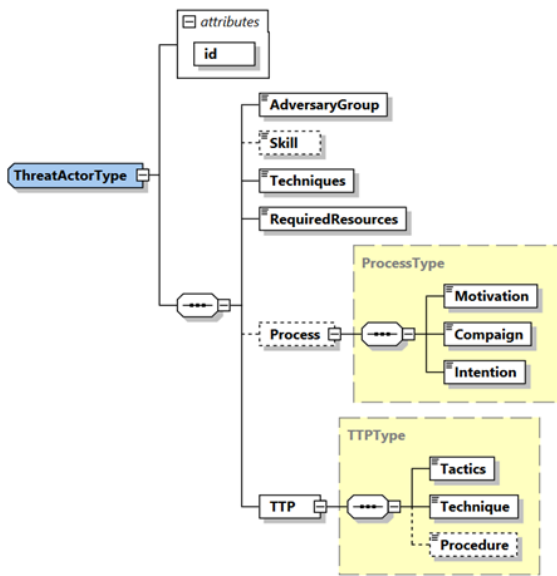


Figure 7. ThreatActor element.

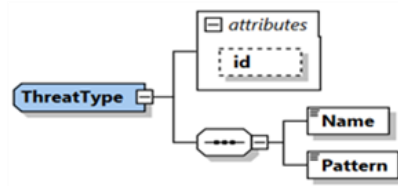


Figure 10. Threat element.

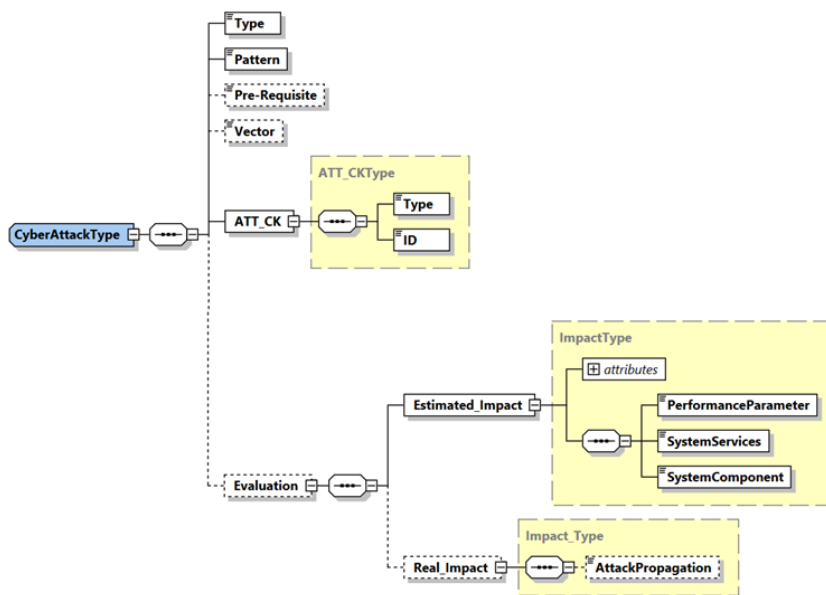


Figure 11. CyberAttack element.

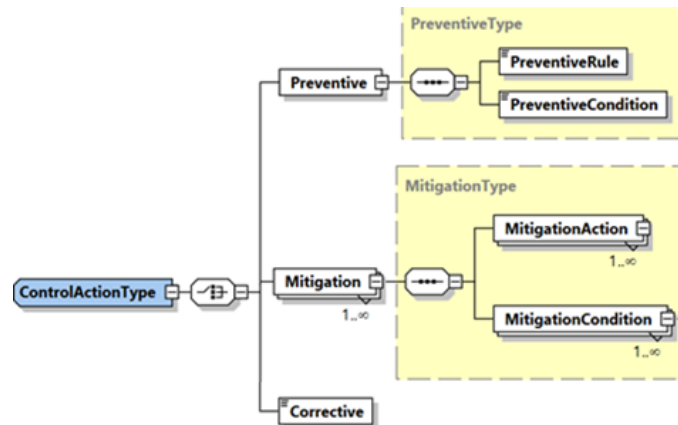


Figure 12. ControlAction element.

3.2.1 DDoS DNS amplification

To see how our model fits a specific attack, we are taking one such type of DDoS attack named as DNS amplification attack presented below. In our XML schema, the CyberAttack element contains all the attributes of the cyber-attack as mentioned above along with two child elements as “Estimated Impact of attack” and “real impact of attack”.

DNS amplification attacks can be triggered in two ways; i) DNS server and ii) NTP (network time protocol). In the example presented below, we are considering the case in which a DNS server overwhelms the resources for a single generated query. To launch this type of attack, the attacker makes use of botnets using spoofed IP addresses. As in Figure 8, we can see one attribute named “ATT & CK”.

By using the attributes of CyberAttack, we can map this information with adversarial tactics, techniques, and common knowledge. For this purpose, we only need to add the ID and type details to connect it with MITRE. We can add the MITRE information here as ID=“ T1498.002” and type “Network Denial of Service: Reflection Amplification” respectively.

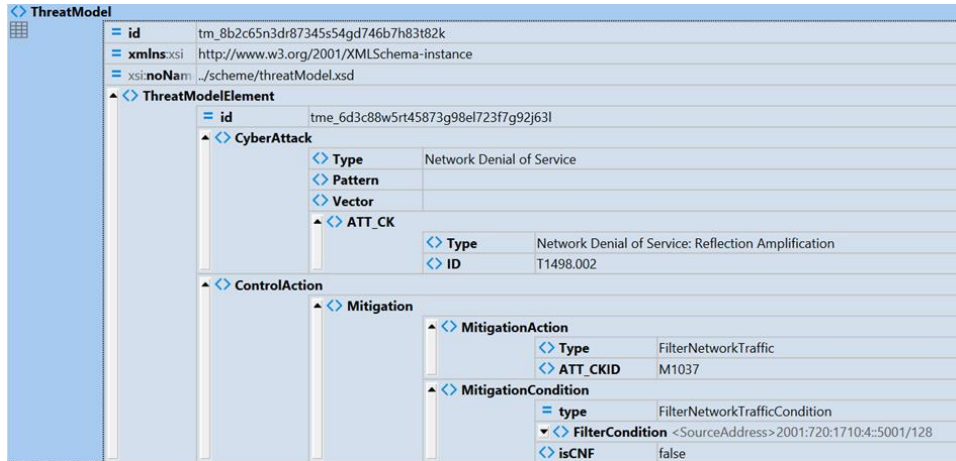
Mitigation and Preventive Strategy

To take preventive and mitigation measures, we have defined the “ControlAction” element. It includes information on proactive and preventive action when facing a threat. To take the appropriate action we need to provide the type and condition of action that could be used as a measure against the threat. In the example mentioned below (Figure 10), we are considering the DNS reflection-amplification attack on a 5G network. For the specific attack, the type and ID mentioned on MITRE is mapped with the schema element.

Mitigation against DNS reflection- amplification attack

Mitigation information can be mapped with the control action element which intercept the incoming network upstream to filter out the legitimate traffic from the attack traffic by utilizing the information of action and condition available. The defence against this attack can be offered by ISP (internet service provider), 3rd parties like CDN (content delivery network), or companies having specialization in the mitigation of DDoS attacks. Depending on the volume of the flood we can make use of “filtering by blocking the source addresses” sourcing the

attack, blocking the ports that are being targeted, and blocking the protocol being used for transport.



```

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<ThreatModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:_="http://www.w3.org/2001/XMLSchema-instance" noNamespaceSchemaLocation="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/XMLSchema-instance http://www.w3.org/2001/XMLSchema-instance" id="tm_8b2c65n3dr87345s54gd746b7h83t82k" ?>
  <ThreatModelElement id="tme_6d3c88w5rt45873g98el723f7g92j63l" ?>
    <CyberAttack ?>
      <Type>Network Denial of Service</Type>
      <Pattern></Pattern>
      <Vector></Vector>
      <ATT_CK ?>
        <Type>Network Denial of Service: Reflection Amplification</Type>
        <ID>T1498.002</ID>
      </ATT_CK>
    </CyberAttack>
    <ControlAction ?>
      <Mitigation ?>
        <MitigationAction ?>
          <Type>FilterNetworkTraffic</Type>
          <ATT_CKID>M1037</ATT_CKID>
        </MitigationAction>
        <MitigationCondition ?>
          <type>FilterNetworkTrafficCondition</type>
          <FilterCondition ?>
            <FilterCondition <SourceAddress>2001:720:1710:4::5001/128</FilterCondition>
            <isCNF>>false</isCNF>
          </FilterCondition>
        </MitigationCondition>
      </Mitigation>
    </ControlAction>
  </ThreatModelElement>
</ThreatModel>
    
```

Figure 13. DDoS DNS amplification attack XML sample.

4 Development of the Distributed Trustable AI Engine

4.1 Introduction

The distributed trustable AI Engine (DTE) module is responsible for the collection of various data from diverse sources of the HORSE infrastructure and the employment of AI/ML modules in order to define the optimum set of policies to ensure a high level of security against a wide range of attacks and impose privacy rules. DTE provides a programming interface to serve AI models and predictions to other modules, thus supporting distributed trustable AI-assisted cybersecurity tools. Moreover, DTE performs data management prior to the actual training, by employing the appropriate policies for anomaly detection (tampered data), as well as data anonymisation. In addition, DTE guarantees compliance of the proposed solutions with the policies module.

Regarding its interaction with other HORSE modules, the DTE receives inputs from the detection and mitigation engine (DEME) in the form of advices, for different types of attacks in the network. In this context, and for predefined known attacks, the information that is forwarded to the DTE includes the id of the HORSE node under attack, and the identified attack type or combination of attack types with a predefined confidence percentage. The advices from the DEME are sent using REST HTTP requests to the DTE. In the same context, DTE can also receive data in the form of policies from PAG, via REST APIs.

In the next step, mitigation measures and methodologies from well-established knowledge bases, such as the MITRE ATT&CK are exploited from both the DEME and DTE in order to build the appropriate mitigation intents. Towards, this end, the output of the DTE is send via REST API to the IBI. It should be noted at this point that the communication among all internal APIs will be based on the JSON format. Moreover, a GUI will be made available for the dynamic retraining of modules if necessary, according to the results of the ML model evaluator that will be described below.

4.2 Internal modules of the DTE

The internal parts of the DTE can be shown in the Figure below and include the NWDAF aggregator, the data processing module, the ML model training, the ML model evaluator, the ML model repository, as well as the intent creator.

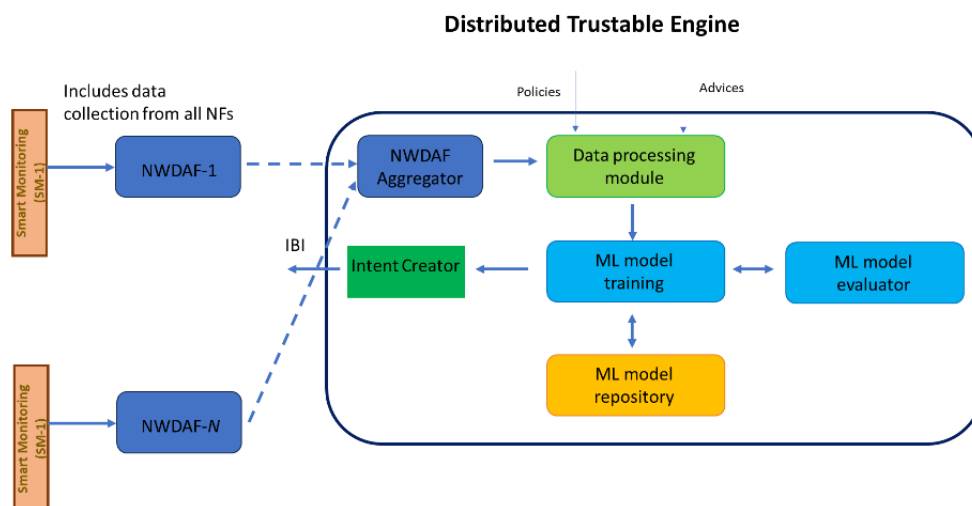


Figure 14. The internal components of the DTE.

If multiple NWDAF instances are deployed, as in the case of HORSE where different instances correspond to different network areas, an NWDAF can act as an aggregation point (i.e. Aggregator NWDAF) and collect analytics information from other NWDAFs, which may have different Serving Areas, to produce the aggregated analytics [15].

The data processing module receives direct inputs from the DEME in the form of node ID, attack type (or combination of attack types) per node, confidence interval, proposed mitigation action and transforms them to a format that can be accessible by the ML training modules.

The ML training is the main module of the DTE where various models are trained for different types of attacks. These will include supervised, unsupervised, and deep reinforcement learning approaches [16]. For this purpose, various datasets will be exploited, representing diverse attacks and network topologies. These datasets are provided i) from HORSE partners, ii) from the NKUA Open5GS and UERANSIM-based testbed, being able to replicate 5G core network attacks, as well as attacks on the 5G RAN, and iii) open datasets that have been used from relevant works on 5G attack scenarios.

At this stage, four different datasets from the literature have been analysed together with ML model training and evaluation:

- The first one is a synthetic 5G cellular network data for NWDAF [17], that is based on Open5GS and UERANSIM. In this context, a topology with a fixed number of subscribers and cells with different traffic patterns and anomalies has been considered, where the anomaly is defined as an unexpectedly high network traffic compared to the average network traffic, fading and stabilizing in time.
- The second dataset is the 5GAD-2022 5G attack detection dataset [18], that is based on Free5GC and UERANSIM. In this case, two types of intercepted network packets are included: "normal" network traffic packets and "attack" packets from attacks against a 5G Core implemented with free5GC. The captures were collected using Tshark or Wireshark on 4 network interfaces (N2, N3, N4, N6) (AMF, gNB, UPF, SMF, DN) within the 5G core. 10 attacks were implemented, mainly relying on REST API calls to different parts of the core.
- The third dataset [19] was generated on an Open5GS and UERANSIM-based testbed. Here, an SMF instance networked in parallel to the original network function acts as the attacker's entry point to the virtualised infrastructure and targets the N4 interface between the SMF and the UPF. The hijacked SMF executes the cyberattacks against the UPF. In order to obtain this data set, the network traffic data of each entity/device was captured through Tshark for each network function and radio element.
- The fourth data set, titled 5G Network Intrusion Detection Dataset (NIDD) [20] contains data in both packet-based format as well as in flow-based formats. 5G-NIDD is generated using the 5G Test Network (5GTN) in Oulu, Finland, thus providing a close resemblance to a real network scenario. 5G-NIDD presents a combination of attack traffic and benign traffic under different attack scenarios, falling into the Distributed Denial of Service (DDoS) and Port Scan/Reconnaissance categories.

For these datasets, various ML models have been evaluated by the ML model evaluator module for predefined ML metrics, such as accuracy and F1-score. It should be noted that apart from evaluating the anomaly detection performance of different ML models, e.g. support vector machine (SVM) with binary kernel for the N4 interface attacks, these attacks have already been replicated at the NKUA testbed and data are collected, as well as logs from various NFs, i.e. AMF, SMF and UPF. After the initial ML model evaluation phase is over, two distinct actions can take place: i) retraining of the ML model in case its performance is below the desired level, or ii) storage of the model in the ML repository, for retrieval in future potential attacks.

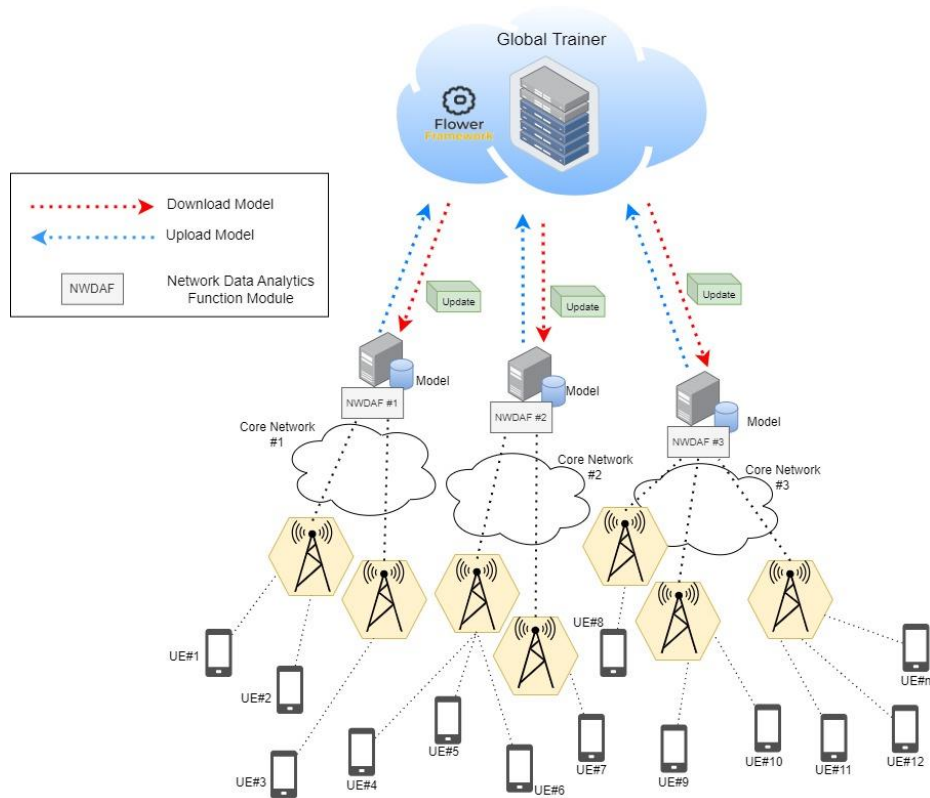


Figure 15. The FLOWER concept in DTE.

4.3 Future extensions

In the next release of the DTE, federated learning (FL) will be also applied, where each NWDAF instance will be responsible for data collection and aggregation in a distinct set of mobile nodes, as shown in the Figure above. In this case, there are multiple DTE instances per subgroup of nodes, where each one trains locally the corresponding models with the available datasets. Afterwards, the master DTE model with the NWDAF aggregator is responsible for updating the global parameters and informing the individual nodes for their updated values. For this purpose, the FLOWER concept will be applied, that can train multiple nodes in an FL fashion [21].

4.4 APIs

Table 2. DEME APIs.

Nº	API	Comments	Inputs	Outputs
1	[action,data] = getRealContext()	This API retrieves data directly from the threat detector and mitigation engine (DEME)	-	This API has two outputs: data coming directly from the DEME that can be in the form of time series and the appropriate action as it has

				been defined by the DEME
2	[action,data] = getEmulatedContext()	Same as before but now data are fed directly from the emulated context	-	Same as the previous API regarding real context.
3	policy = getpolicy()	This API retrieves the set of policies in HORSE directly from the policies and data governance (PAG)	-	The set of policies of HORSE that are fed to the DTE to ensure compliance during ML model training
4	pr_data=data_process(data)	API that is used for proper data manipulation that are received from real or emulated context This processing might include proper format transformation, missing values replacement, data normalization, etc. The output is a similar data matrix as the input one, where the processed values are stored.	Data from real or emulated context	Processed data ready for ML training
5	model_predictor = train_model(pr_data,train_method)	This API is used to train the DTE models based on the ingested data after the data_process API output. It typically accepts a dataset or data parameters required for model training. The API triggers the training process and updates the DTE models accordingly.	The processed data along with the desired training method	The output model_predictor is the trained model that can be either in the form of a function or a structure.
6	ML_KPIs = evaluate_model(model_predictor, test_data)	This API is used to evaluate the performance of the trained models. It accepts a dataset or data samples and returns the evaluation results, such as precision, recall, F1-score, mean square error (MSE) or other relevant metrics, indicating how well the models are performing. Typically, test_data can be a subset of	Test data along with model predictor	A list of KPIs in the form of a table to evaluate the accuracy of the trained model

		the pr_data that can be used for evaluation purposes.		
7	intent = generate_intent(data)	This API generates an intent to be passed to the Intent-Based Interface (IBI)		An intent in the form of an action

4.5 Machine Learning Function Orchestrator (MLFO)

The Machine Learning Function Orchestrator (MLFO) is a framework comprising various components designed to provide advanced functionalities for training machine learning models. Leveraging state-of-the-art libraries such as scikit-learn, Keras, and TensorFlow, MLFO facilitates the training and retraining of models. The orchestrated workflows trigger the execution of both Machine Learning (ML) and Deep Learning (DL) experiments, harnessing the power of cloud-native infrastructure. MLFO integrates three key components to ensure seamless functionality:

- Data Science Platform Component:** This element of MLFO is dedicated to supporting the data science process, offering a robust environment for data exploration, feature engineering, and model development. It provides a user-friendly interface and tools that streamline the data science workflow.
- Workflow Orchestrator:** At the core of MLFO, the Workflow Orchestrator coordinates and manages the entire machine learning pipeline. It oversees the training and retraining processes, ensuring efficient communication between different stages of model development. This component plays a crucial role in optimizing the orchestration of tasks to enhance overall system performance.
- Models Registry:** MLFO incorporates a Models Registry that serves as a centralized repository for storing and versioning training artifacts. This includes metrics and models generated during the training process. The Registry ensures traceability and reproducibility by maintaining a comprehensive record of model versions and associated performance metrics.

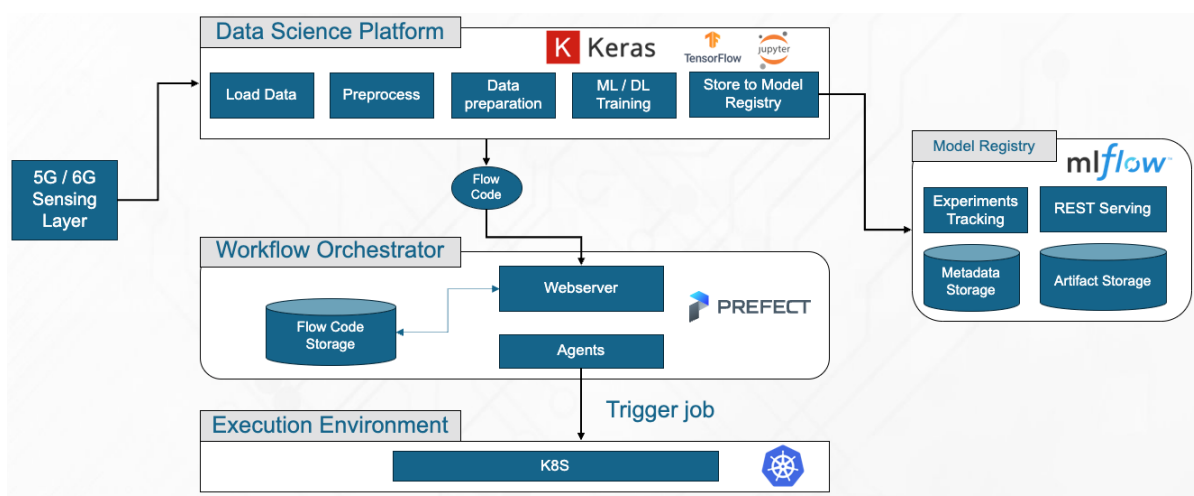


Figure 16. MLFO Architecture.

Table 3 below depicts the list of components comprising MLFO, providing their name, short description and key technologies used in their technical implementation.

Table 3. MLFO Comonents.

Component Name	Short Description	Technologies
Data Science Platform	Scientific Environment for training ML / DL experiments	scikit-learn, keras, Tensorflow, pandas, numpy, matplotlib, seaborn, SHAP, docker, K8S, Jupyter
Workflow Orchestrator	Orchestrates the execution of ML / DL experiments as K8S objects	Prefect, MinIO, Postgress, docker, K8S
Model Registry	Provide storage functionalities for training models and artifacts	MLFlow, Postgress, MinIO

4.6 Explainability of ML / DL Experiments

The Distributed Trustable Engine goes beyond model training by incorporating the crucial element of explainability for Machine Learning (ML) and Deep Learning (DL) experiments. Post-training, the Machine Learning Function Orchestrator (MLFO) utilises training artifacts to execute explainability methods, providing insights into the inner workings of models. This process is essential for assessing the performance and trustworthiness of models within the target cloud-native environment. MLFO, equipped with the installed and configured SHAP library, is poised to deliver robust explainability capabilities. Furthermore, it is designed to be adaptable, allowing seamless extension to support other eXplainable AI (xAI) libraries such as LIME, omniXAI, and ELi5. This flexibility ensures that MLFO remains at the forefront of explainability methodologies, accommodating diverse and evolving approaches to interpreting and understanding machine learning models.

In the case of the 5GNID dataset [22] employed for training machine learning (ML) and deep learning (DL) models aimed at discerning between normal and abnormal packets, with a specific focus on identifying the nature of abnormalities such as various types of cyber-attacks, the assessment of explainability plays a pivotal role in evaluating experimental outcomes. This involves elucidating how input features influence the classification output. To illustrate this concept, consider the diagram below, which delineates the impact of the "Attack Tool" feature on the ultimate classification decision output. This visual representation serves as a succinct depiction of how this particular input feature contributes to the determination of whether a packet is associated with a normal pattern or indicative of a specific type of attack. Understanding such contributions is essential for enhancing the transparency and interpretability of the model's decision-making process, ultimately fostering a more comprehensive comprehension of its performance characteristics.

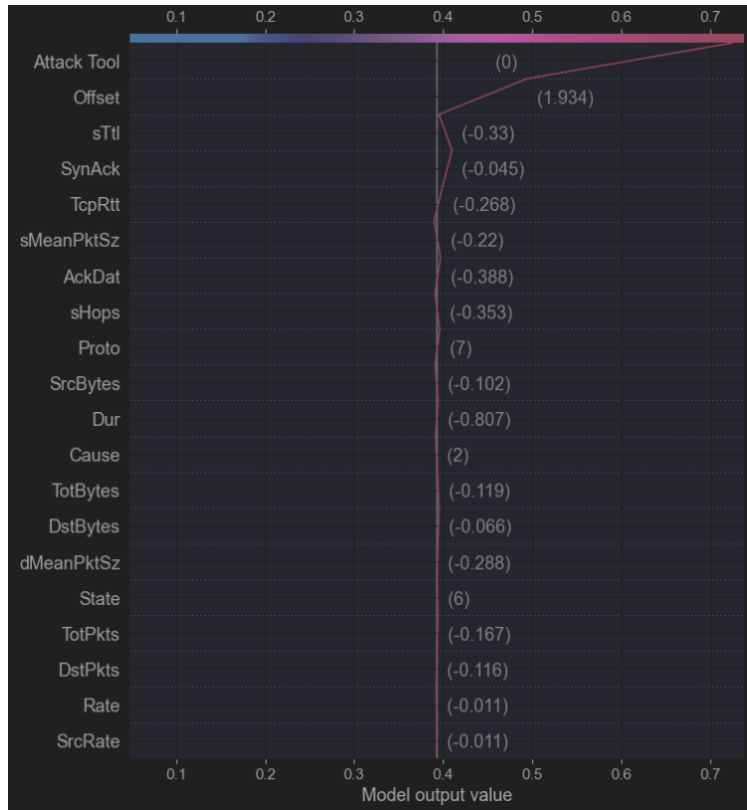


Figure 17. xAI example.

5 Development of the Policies and Data Governance

5.1 User Stories

For the description of the Policies and Data Governance component, we have opted to employ user stories. In software development and product management, a user story is an informal, natural language description of features of a software system. A user story is written from the perspective of an end user or user of a system.

In particular for the PAG, the user of the software is understood to be the “HORSE Operator”, i.e. an individual with 6G domain knowledge who is using the HORSE platform in order to set up a 6G monitoring infrastructure. The user stories related to the PAG stem from the PAG requirements (see Appendix B) and have been written around the main functionalities of access management, data anonymisation, data retention, data encryption and observability. For each user story, acceptance criteria is given, which will help the development & management teams to define a user story as done, or still in-progress.

5.1.1 Access Management

Access Management	
As a	HORSE Operator
I want to	Use a “Policies Editor” (part of the HORSE Dashboard)
So that	I manage the access policies on the collected datasets

Acceptance Criteria

- The user can define the level of the access policies on a dataset based on the requestor's attributes.
- The user can update access policies on a dataset.
- The user can remove access policies on a dataset.
- The user can combine access policies on a dataset.
- The system automatically enforces access control decisions on the collected datasets based on the associated dataset access policies.

The policies editor allows the HORSE Operator to grant/deny access to a specific dataset per user and per component. In the former case, access to a specific dataset can be granted/denied to a specific username. In the latter case, access to a specific dataset can be granted/denied to a specific component from a list of HORSE system components, e.g., the Early Modelling component, the DTE component, etc.

5.1.1.1 Implementation

For the implementation of the access management functionality, OpenFGA [23] is used as the basis for the authorisation service, complimented by custom HORSE implementation using Python programming language.

OpenFGA leverages Google’s Zanzibar, a global system for storing and evaluating access control lists. Zanzibar provides a uniform data model and configuration language for expressing a wide range of access control policies from hundreds of client services. Furthermore, it supports Python, which is the programming language used for the custom HORSE implementation. On the less positive side, OpenFGA makes it difficult to support queries of the type “Allow access to all except...” which in certain cases might increase the effort needed to express certain authorisation policies.

5.1.1.2 Sequence Diagrams

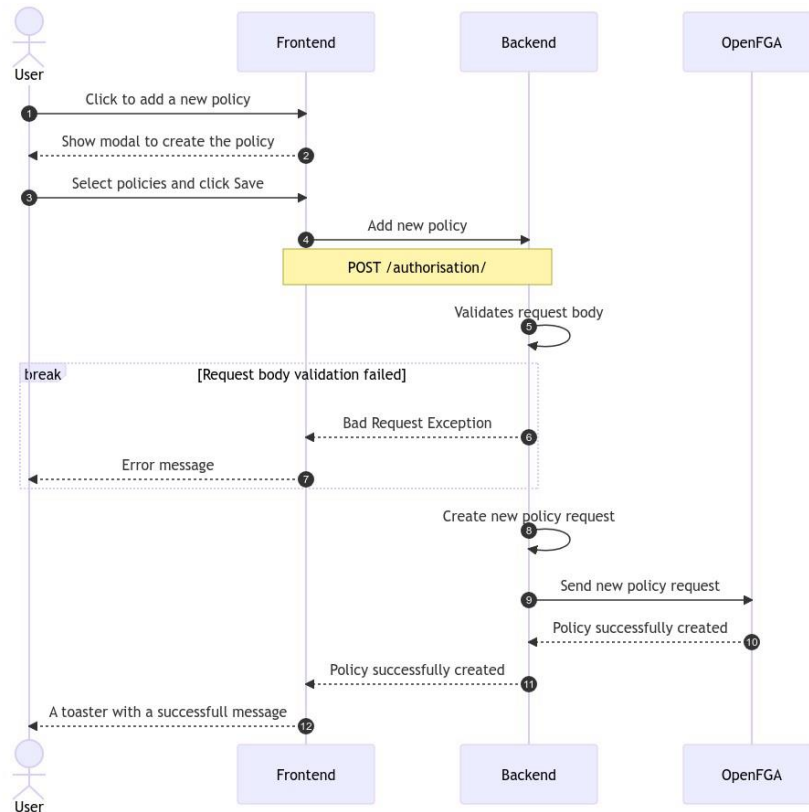


Figure 18. Create New Access Policy

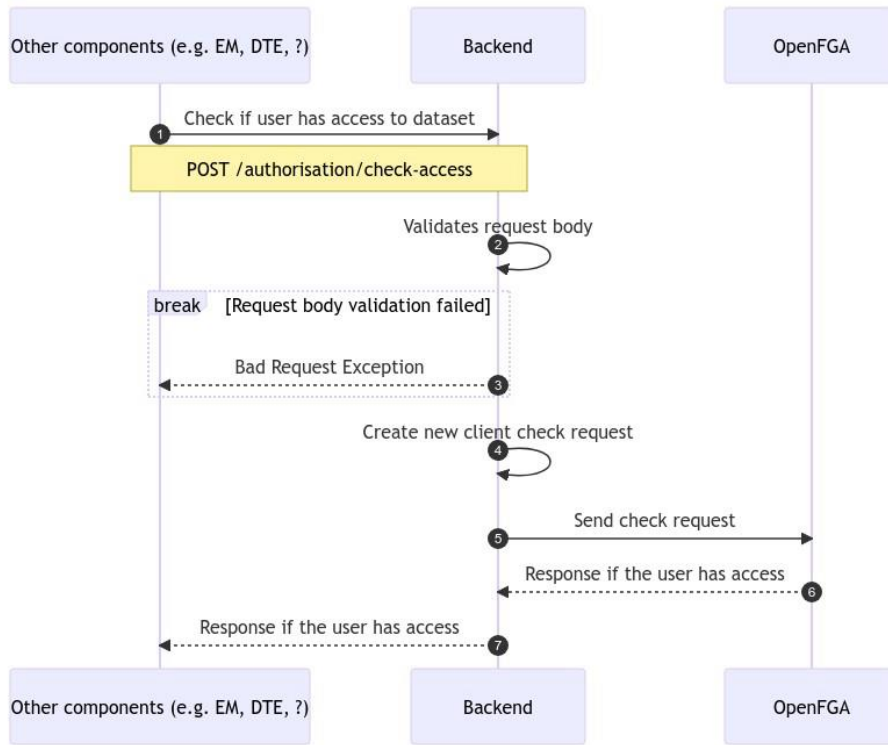


Figure 19. Check Access Authorization

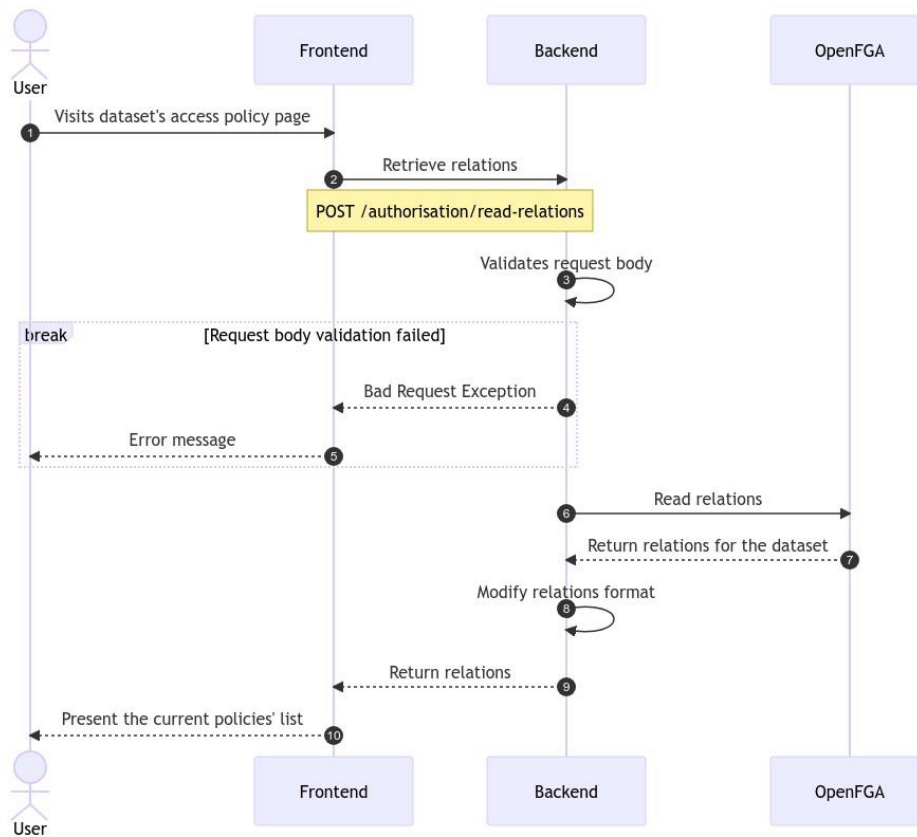


Figure 20. Read Policies List

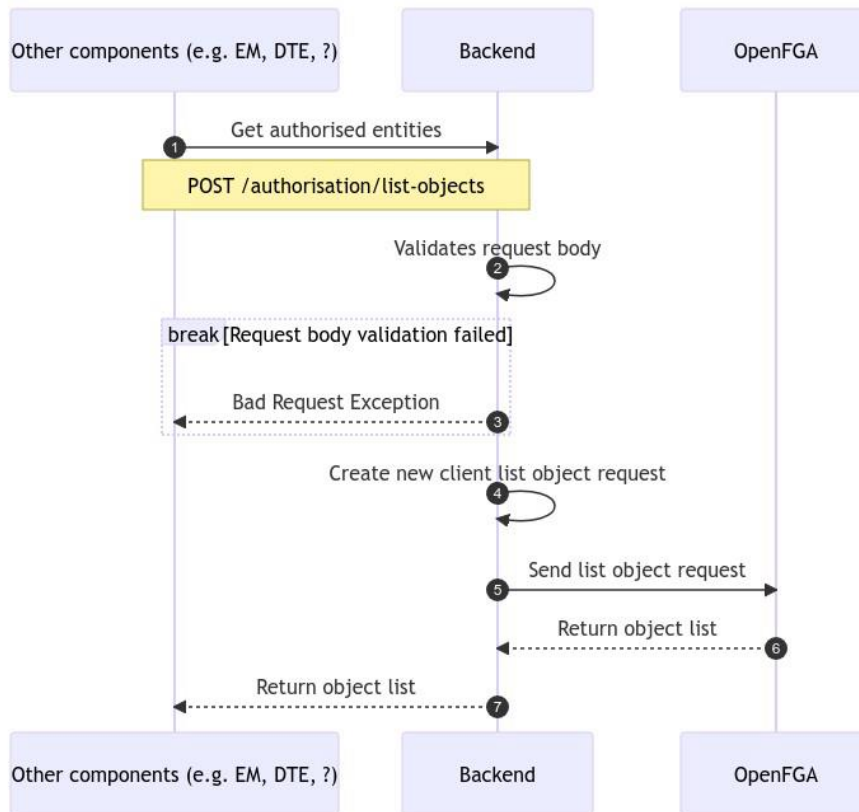


Figure 21. List Authorization Objects

5.1.2 Data Anonymisation

Data Anonymisation	
As a	HORSE Operator
I want to	Use a “Policies Editor” (part of the HORSE Dashboard)
So that	I configure data anonymisation rules on the collected datasets

Acceptance Criteria

- The user can classify the disclosure risk for the dataset’s attributes.
- The user can define data anonymisation rules based on the classification of the dataset’s attributes for disclosure risk and the anonymisation method used by the component.
- The system automatically executes data anonymisation operations on the collected datasets.

5.1.2.1 Implementation

For the implementation of data anonymisation, the PAG will use a simple anonymisation tool along with some custom implemented methods. The tool is called Smile [24] and implements text anonymisation in many languages using Faker.

Smile includes methods on anonymising network-related fields (e.g., IP addresses, MAC addresses, URLs, etc.).

Examples of possible anonymisation methods that the HORSE Operator can select on a column:

- IPv4, IPv6 Address Masking: changes an IPv4 or/and an IPv6 address into a fake one.
- Text Masking: changes a text/string by adding '*' characters into the string.
- Perturbation: adds "random noise" to a numeric field
- Date Masking: changes a date value into a fake one.

The enforcement of anonymisation (e.g., the anonymisation of certain attributes of a stored dataset) is applied in the data store(s) of the HORSE platform. For more information on the available data store(s) of the HORSE platform, see section 5.2.

5.1.2.2 Sequence Diagrams

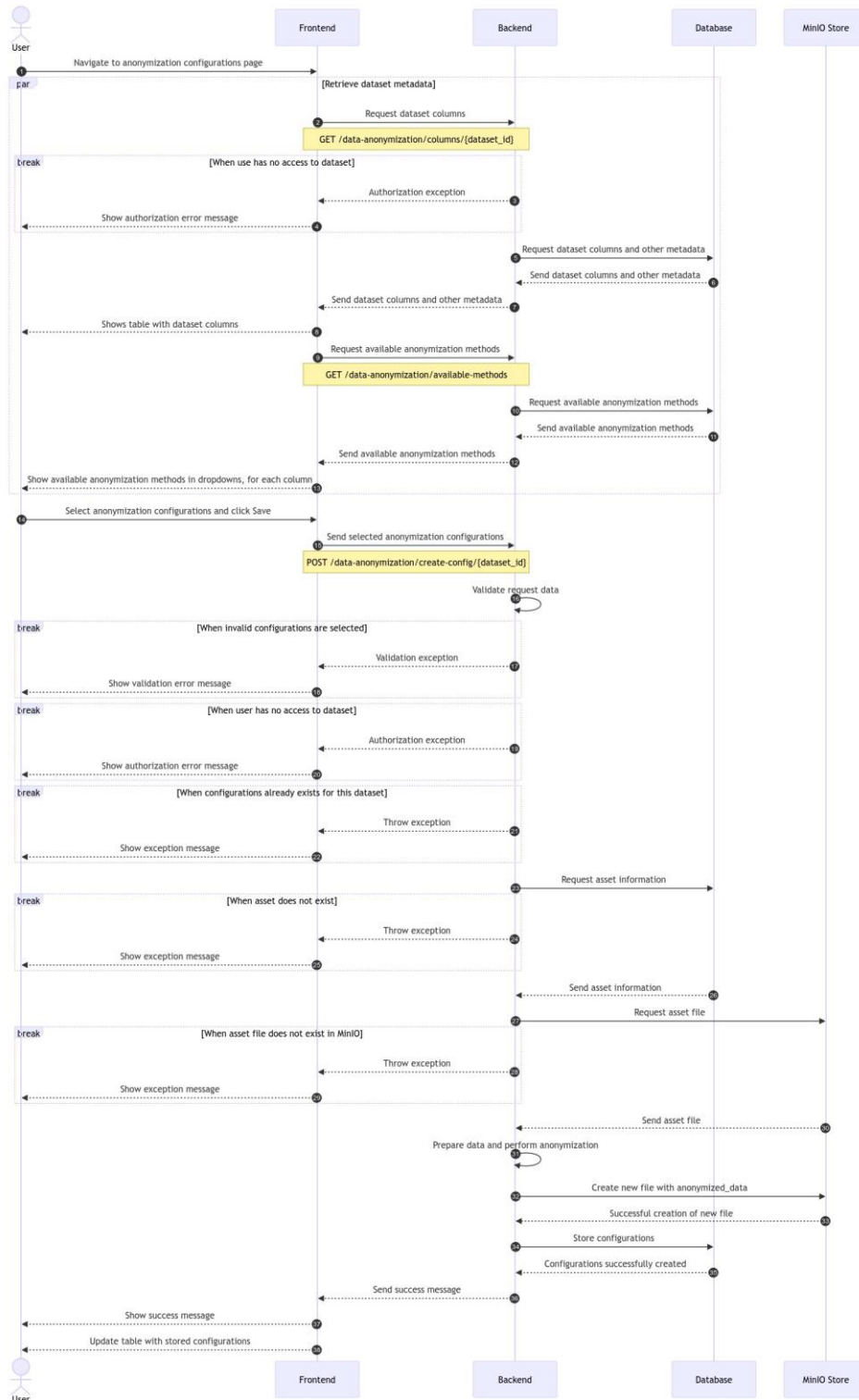


Figure 22. Store Anonymisation Config & Perform Anonymisation

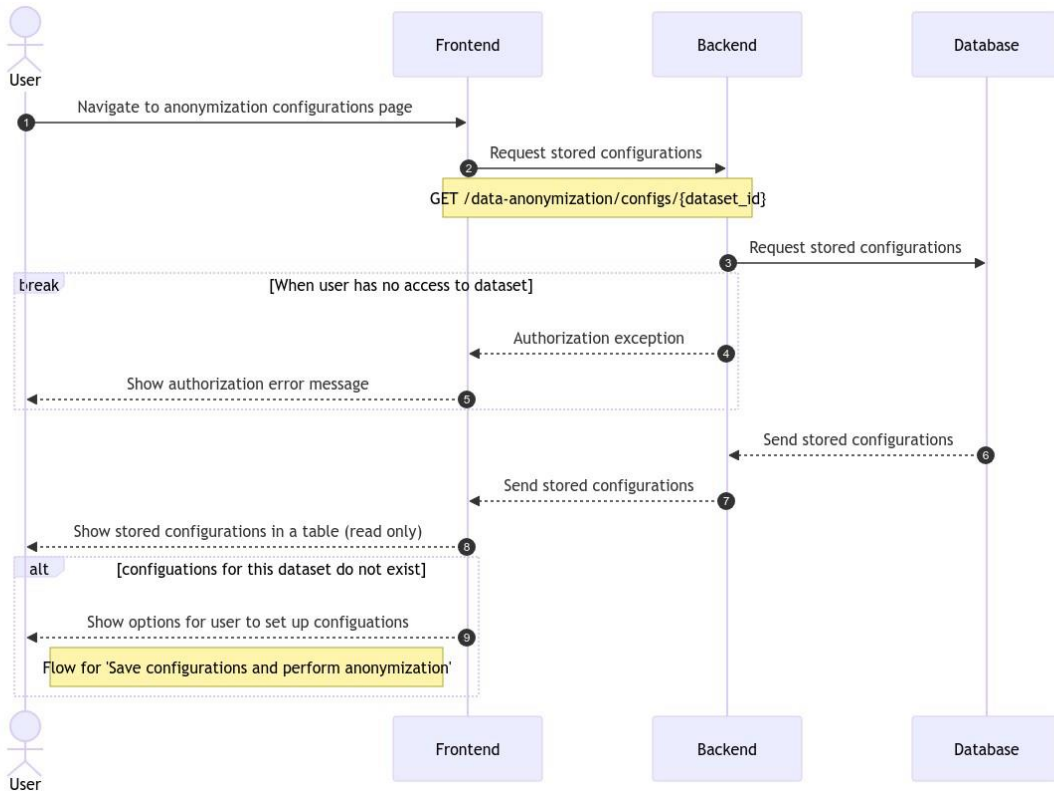


Figure 23. Retrieve Anonymisation Config

5.1.3 Data Retention

Data Retention	
As a	HORSE Operator
I want to	Use a “Policies Editor” (part of the HORSE Dashboard)
So that	I manage the data retention rules of the collected datasets

Acceptance Criteria

- The user can define retention rules for a collected dataset based on the dataset’s attributes (e.g. delete dataset 3 years after the collection timestamp, delete dataset 24h after the collection timestamp for datasets originating from RAN, etc.).
- The user can update retention rules of a dataset.
- The user can remove retention rules of a dataset.
- The user can combine retention rules of a dataset.
- The system automatically implements dataset retention rules at the correct timing for the collected datasets.

5.1.3.1 Implementation

For the implementation of data retention, a simple, lightweight Python job scheduler [25] was used. It offers:

- functionality to run in the background, in a separate thread;
- many options for setting up a scheduler which runs at specified intervals;
- functionality for exception handling.

The enforcement of the retention policy (e.g., the deletion of a dataset) is applied in the data store(s) of the HORSE platform. For more information on the available data store(s) of the HORSE platform, see section 5.2.

5.1.3.2 Sequence Diagrams

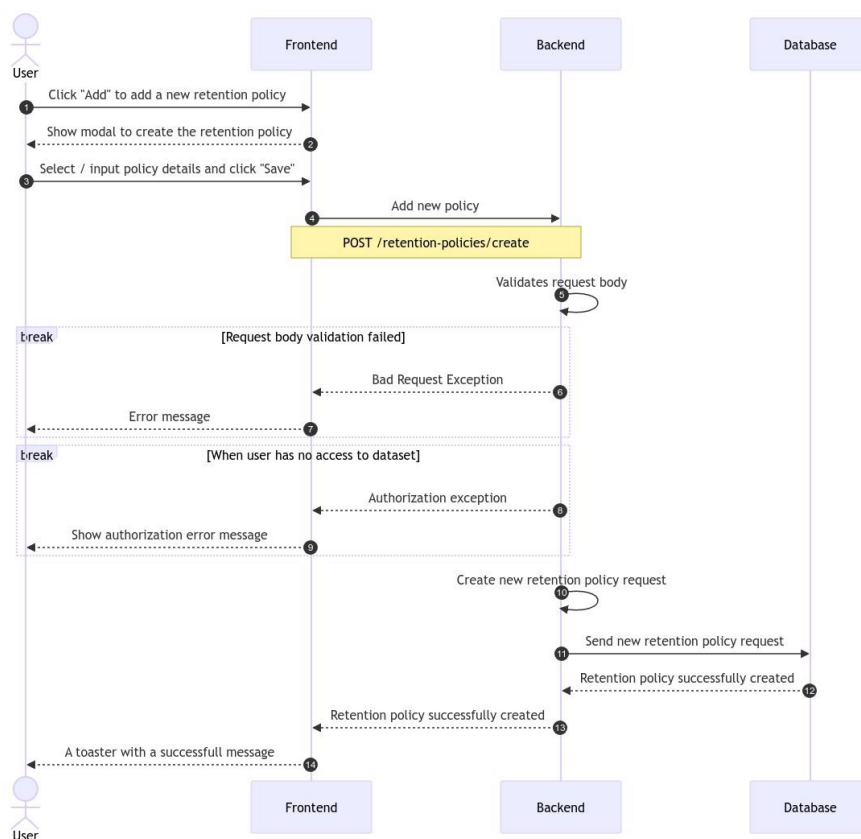


Figure 24. Create New Retention Policy

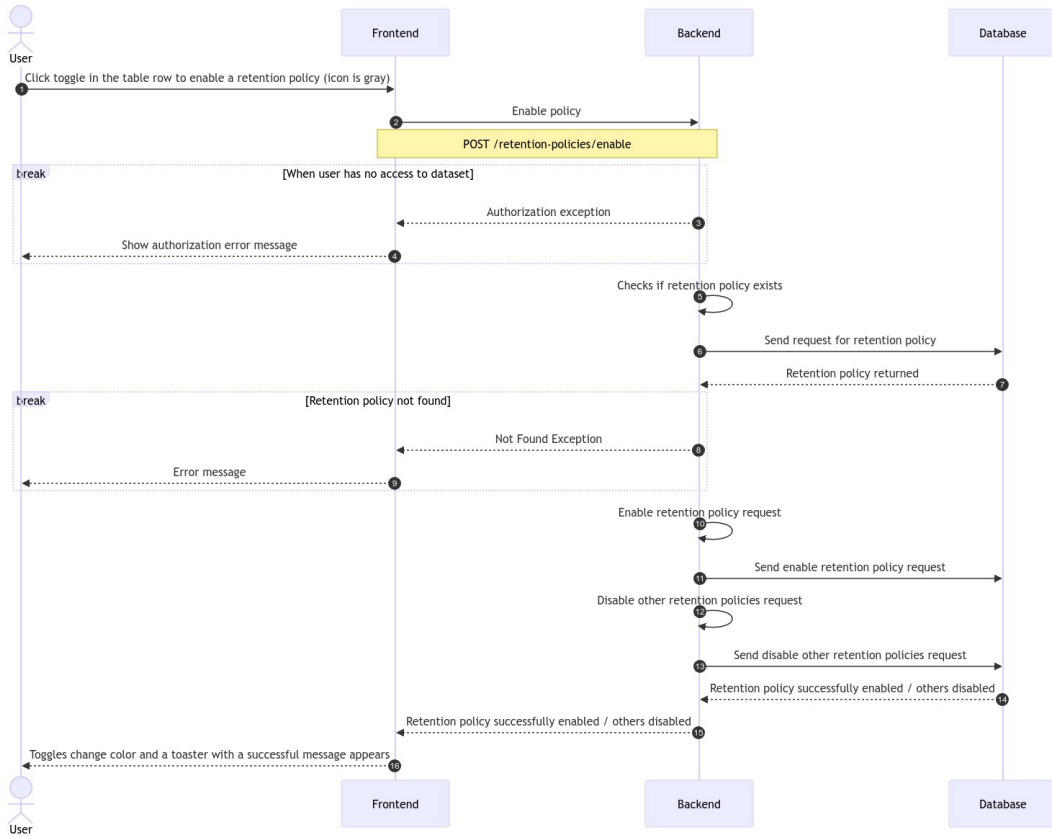


Figure 25. Enable Retention Policy

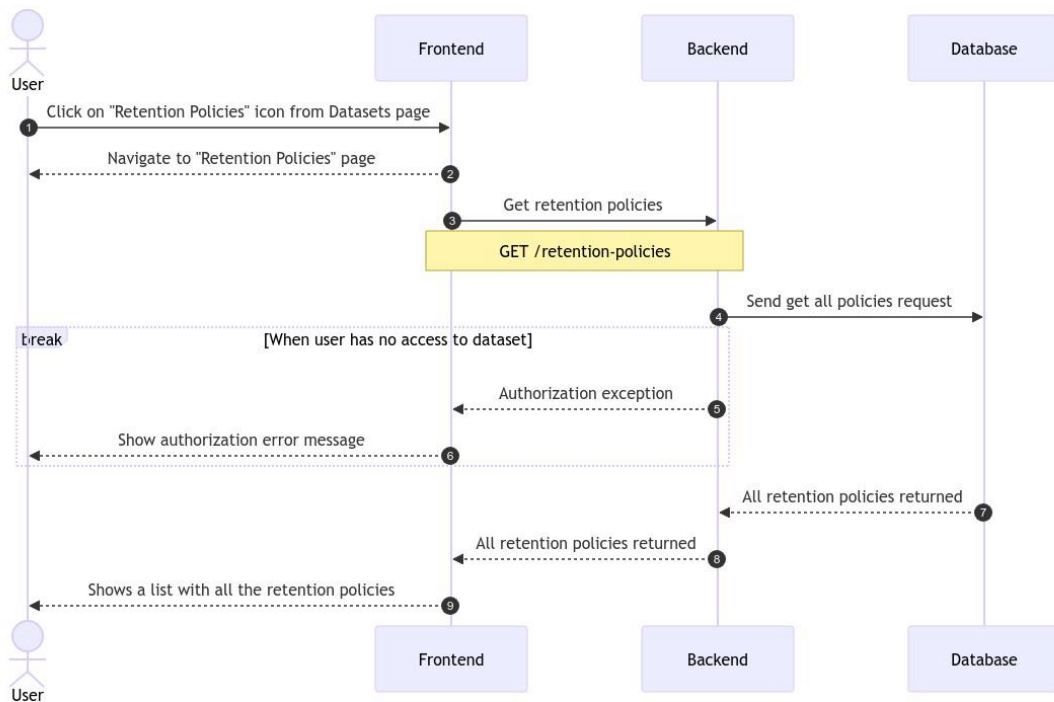


Figure 26. Read Retention Policies List

5.1.4 Data Encryption

Data Encryption	
As a	HORSE Operator
I want to	Use a “Policies Editor” (part of the HORSE Dashboard)
So that	I configure encryption on the collected datasets

This user story shall be further elaborated in deliverable D3.2 HORSE Platform Intelligence developed (IT-2).

5.1.5 Observability

Observability	
As a	HORSE Operator
I want to	Use a UI
So that	I monitor the execution status of the dataset collection jobs

This user story shall be further elaborated in deliverable D3.2 HORSE Platform Intelligence developed (IT-2).

5.2 Datasets

The following datasets have been used during the development of the initial release of the PAG component and have been imported in the Demo Data Store:

- **sample_data.json**

This file contains sample data regarding VNFs that belong to a simulated 5G Network. In order to capture the bytes transmitted, Kiali and Prometheus were reused on top of the containerized network Istio Service Mesh.

The captured metric is **istio_request_bytes_sum** (the total number of bytes, added together across all requests).

The data model which accommodates the captured metric is an array of values that capture the sum of bytes every 15 seconds beginning with metadata in each value.

The values list contains all the requests performed in the 5G network.

- **5G-NIDD**

This dataset presents 5G-NIDD [26], a fully labelled dataset built on a functional 5G test network that can be used by those who develop and test AI/ML solutions. 5G-NIDD contains data extracted from a 5G testbed. The testbed is attached to 5G Test Network in University of Oulu, Finland. The data are extracted from two base stations, each having an attacker node, several benign 5G users. The attacker nodes attack the server deployed in 5GTN MEC environment. The attack scenarios include DoS attacks and port scans. Under DoS attacks, the dataset contains ICMP Flood, UDP Flood, SYN Flood, HTTP Flood, and Slowrate DoS. Under port scans, the dataset contains SYN Scan, TCP Connect Scan, and UDP Scan.

5.2.1 HORSE Data Store

At M12 of the project, as per the architecture described in deliverable D2.2, the main data store for the data assets in the HORSE project resides inside the pre-processing module.

The design of the PAG component has taken into account the possibility that during the evolution of the project, multiple distributed data stores might become available. The PAG component has implemented a structure in which datasets and their data stores are connected, which enables the PAG to extend its implementation and support multiple data stores, unknown at M12, in case this need arises in the future.

5.2.2 Demo Data Store

A demo data store has been implemented provisionally for the development of the PAG initial release. This demo data store is based on MinIO and holds the demo datasets available at M12 of the project.

The demo data store will not be maintained beyond the development of initial release of the PAG component.

MinIO [27] is a high-performance, S3 compatible object store. It is built for large scale AI/ML, data lake and database workloads. It is software-defined and runs on any cloud or on-premises infrastructure.

MinIO is built to deliver exceptional speed and efficiency, allowing organizations to effortlessly manage and scale their object storage needs. Its flexibility, ease of deployment, and robust features make MinIO a compelling choice for businesses seeking a reliable and cost-effective solution for distributed object storage.

Below is a screenshot from MinIO Console, the web-based graphical user interface that MinIO provides. MinIO Console is a useful tool for interacting with a MinIO Server and manage various tasks like Identity and Access Management, Metrics and Log Monitoring, or Server Configuration.

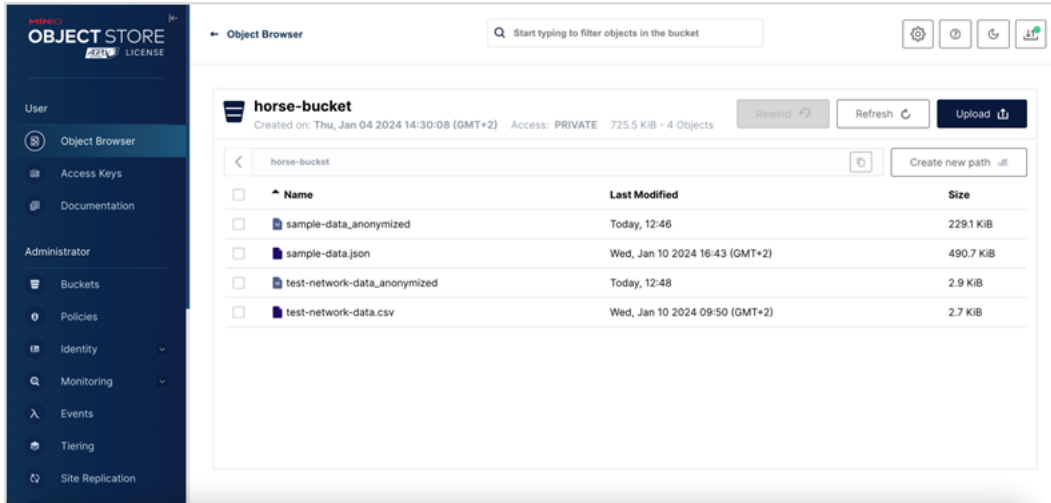


Figure 27. MinIO Console

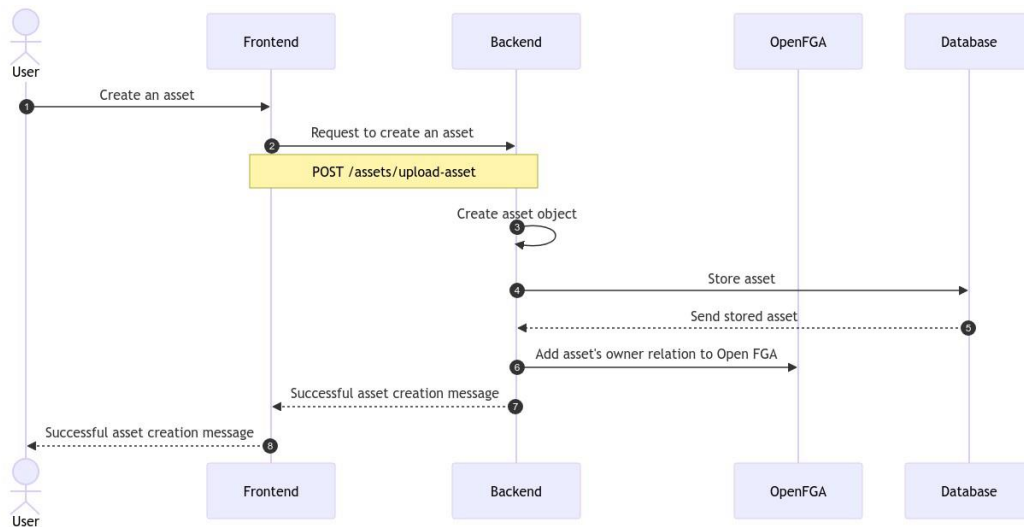


Figure 28. Create New Asset

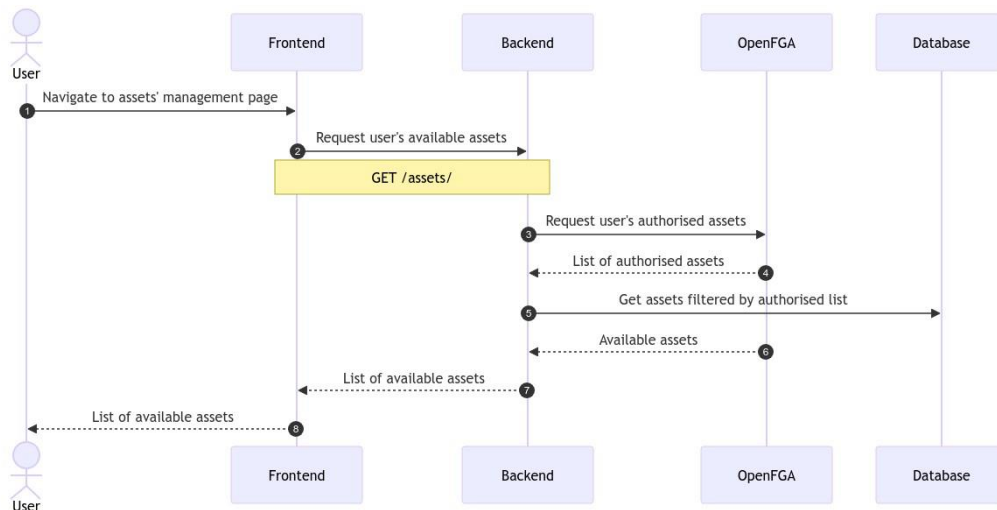


Figure 29. Read Assets List

5.2.3 Metadata

The PAG Storage does not store data assets per se; it stores the metadata of the data assets, while the data assets themselves are stored in the available data store(s) of the HORSE platform (see 5.2.1 and 5.2.2 above). For each dataset, the following metadata are imported into the PAG Storage:

- name
- description
- file type (e.g., json, csv)
- the path of the dataset inside the data space
- columns (types and names in a json format columns)

5.3 Development Roadmap

The initial release of the PAG component comes on M13 of the project, as a standalone component ready to be integrated. It includes the functionality for access management, data anonymisation and data retention, as described with the help of user stories in section 5.2 above. Additionally, it includes (i) the PAG storage for storing the metadata and the policies, and (ii) the Demo Data Store for storing the datasets.

The functionality for data encryption and observability shall be further elaborated in deliverable D3.2 HORSE Platform Intelligence developed (IT-2) and will be developed as part of the final release of the PAG component on M30. Additionally, the final release shall include the connection with the HORSE Data Store and any updates or enhancements to the functionalities already delivered on M13.

PAG initial release (M13)	PAG final release (M30)
Access Management	Data Encryption
Data Anonymisation	Observability
Data Retention	Connection with HORSE Data Store
Demo Data Store	Enhancements/Updates of functionalities delivered on M13
PAG Storage	

6 Development of the Threat Detector and Mitigation Engine

6.1 Threat Detector and Mitigation Engine Features

According to the main concepts described in Section 3.2 of the HORSE Architectural Deliverable D2.2 [28], here recalled for the sake of simplicity, the HORSE platform encompasses three main layers:

- the Intent-based Interface (IBI), that aims to simplify the network configuration and operation by receiving high-level intents from the network manager or software agents.
- the AI Secure and Trustable Orchestration (STO) module, that enables reliable network operation by assuring correct orchestration of the network resources and execution of policies proposed by the IBI layer.
- the Platform Intelligence (PIL) module, that adds intelligence and autonomy to the network management, including sub-modules that can predict the behaviour of the network before reconfiguring the network and sub-modules capable of detecting and reacting to network security threats, i.e., the Real Context -DEME component Figure 30.

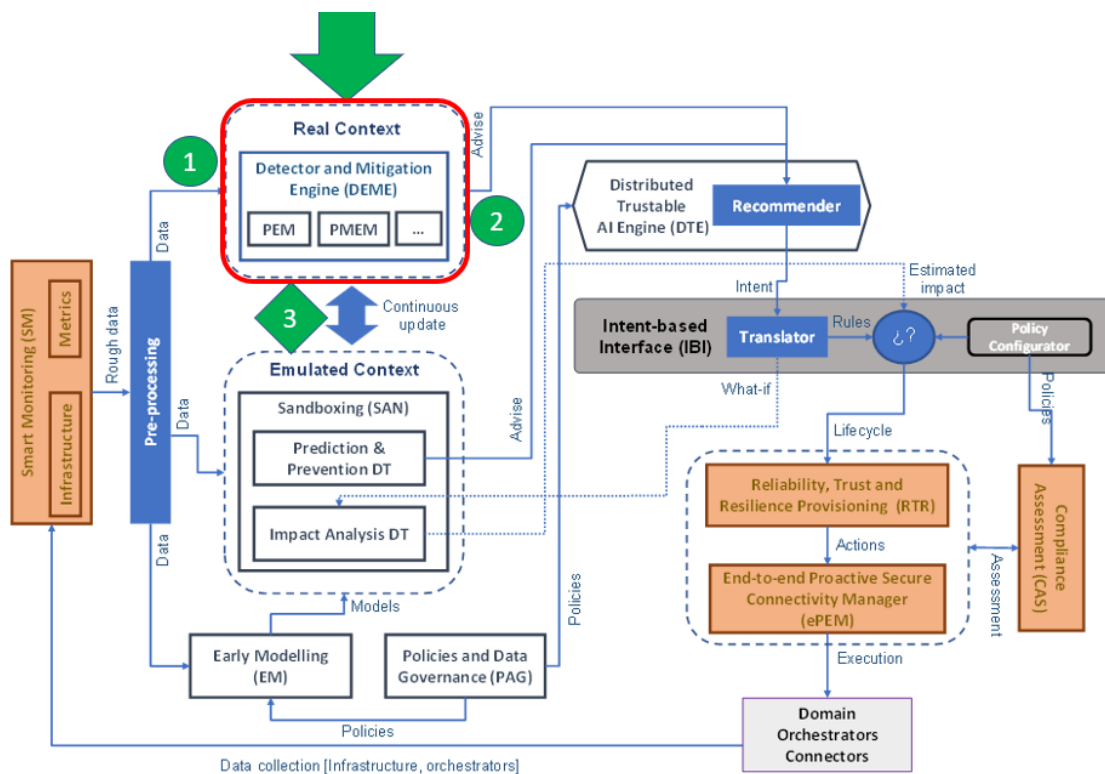


Figure 30. DEME sub-module in the overall architecture [28].

The Detector and Mitigation Engine (DEME) works in the “real context” providing threat detection in the real infrastructure. It focuses on threat detection and high-level mitigation advise with a special attention to the most dangerous attack cases, able to impact, and often paralyze, whole portions of the network for a long amount of time [28].

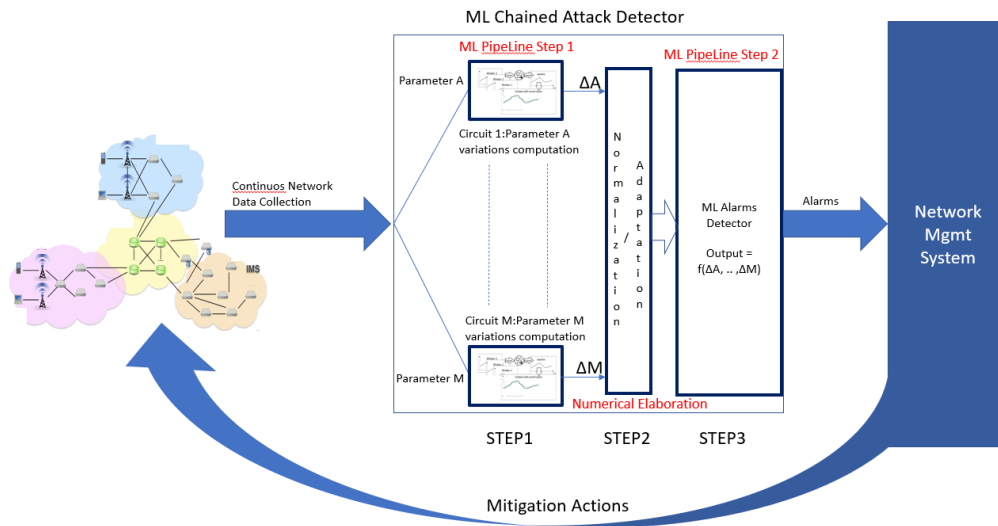


Figure 31. HORSE Threat Detector Block Diagram

The conceived solution applies a ML pipeline or chain architecture (that are currently seldom applied and mainly on the image processing arena [41]) to improve the SotA.

The overall implemented solution, exemplified in Figure 31, consists of a double processing step, easily extendible with further steps according to the evolving future cybercrime landscape.

The first ML step 1 is dedicated to the baseline elaboration and real-time parameters variations computation and therefore allows to avoid provisioning complex thresholds sets.

The last step, ML step 2, is able to analyse simultaneously all the parameters variation in order:

- Provide Early Detection on any known alarm identifiable by the M monitored parameters, much faster than using SotA specific detectors.
- Provide Early Detection on any unknown alarm, 0-day, identifiable by the M monitored parameters.
- Provide a relative attack probability or detection confidence.
- Automatically adapt the probability numerical evaluations based on the network evolution.

More in detail, the scheme that has been conceived, implemented and tested, is constituted by a sequence of three specialized steps, Figure 32, whose composition is particularly suited to the functional requirements outlined.

- 1) STEP1: Use of predictive analytics (Supervised) ML tools (i.e., regressions) trained by the historical network data. This first step must provide N parallel blocks, one for each monitored parameter, and must be able to provide in output the N real-time comparisons of the current network data (that are periodically collected) with the predicted baselines.
- 2) STEP2: Use of a numerical elaboration to normalize and adapt the N comparison outputs in order to provide the following ML step a homogeneous set of data to work

- upon. Each comparison, indicated with Δ in Figure 32, will have to be normalized and adapted.
- 3) STEP3: Use of a (Supervised) ML tool trained on the variations of the N variables provided by the previous steps to promptly detect any related attack with a 360 degrees perspective (no siloization).

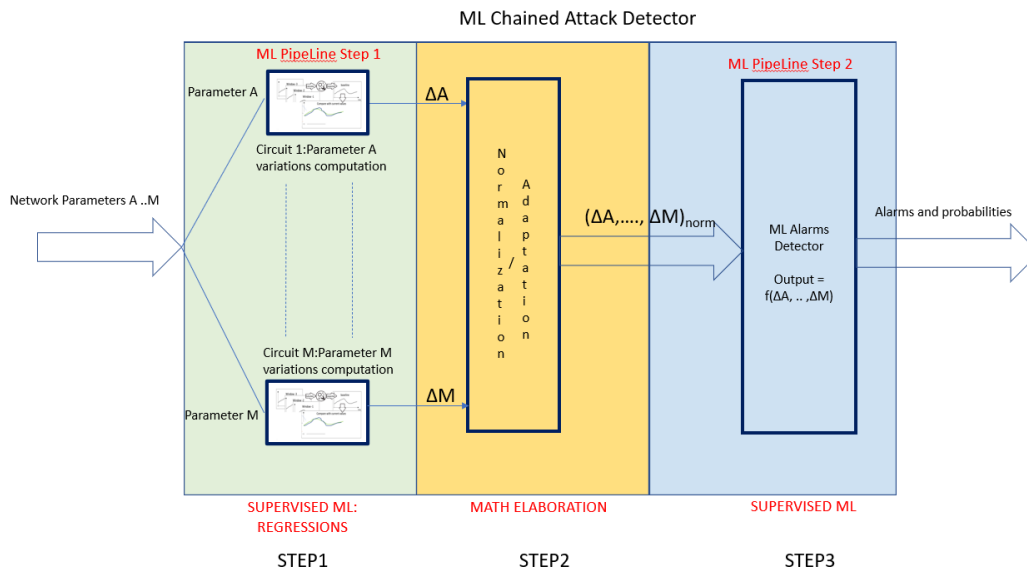


Figure 32. ML-based chained attack detector steps

6.1.1 Step by step detailed operation description

The first stage, STEP1, foresees the application of predictive Supervised Machine Learning Techniques.

In this step many techniques have been tried and among them stands out the first implementation of a new regression algorithm (Ericsson patent) specifically conceived for threat detection sped-up (detailed tests results will be available and presented in the second phase of the HORSE project, in the next version of this deliverable, D3.2, and in the WP5 context).

Each regression is trained with a proper set of historical values and is able to retrieve a trained model, i.e., a kind of predicted trend or baseline that is automatically updated during the following network life.

It is just the case to mention that different concurrent and multi-threading technologies and frameworks have been investigated, applied and tested to achieve the best performance and scalability results.

After the training phase, each ML regression starts comparing the current values extracted from the network with the expected (predicted) trends.

It is important to highlight the fact that no threshold scheme is applied on the comparison outputs: the computed differences are only normalized and adapted, in the STEP2 stage, in order to provide the last ML block with a homogeneous and consistent data space.

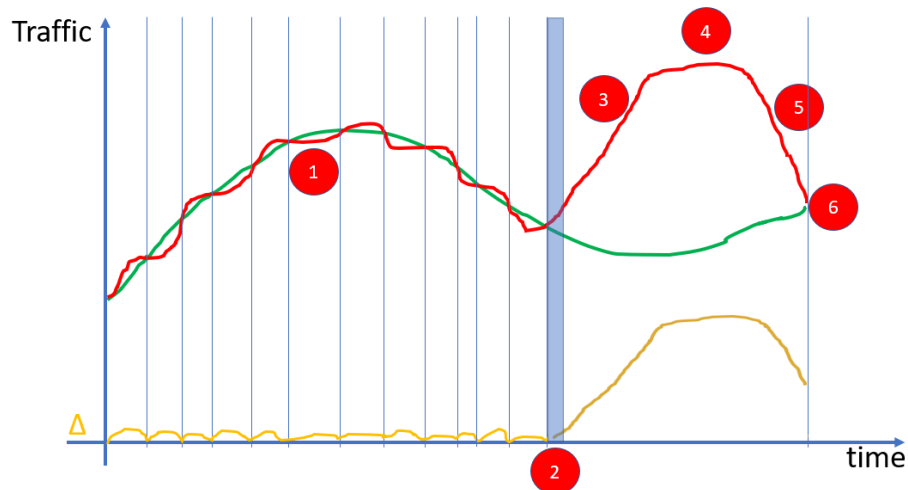


Figure 33. STEP1 regressions

Figure 33 shows in green the predicted trend and in red the current values that are progressively collected and compared with the green prediction in search of any possible anomaly.

1

During normal states, the normal network fluctuations cause the red curve to ripple around the green predictions, therefore the Δ comparison will result in relatively small rippling values.

2

At the beginning of a real attack, highlighted by the light blue area, traffic is starting to grow but its difference with the expected value (Δ comparison) has not yet reached very high values. Therefore, the last ML stage, STEP3, during the training phases will observe small rippling values relative to both normal states and relative to the initial stages of the attacks. It will therefore estimate a normal or attack probability based on the historical attack frequency (the occurrence rate). In other words, if an attack occurs repeatedly the ML learning STEP 3 will tend to modify the confidence levels associated to small rippling values and vice versa.

3

During the attack the monitored parameter rumps up in a very short time and the detector progressively increase the associated detection confidence triggering the network management mitigation actions.

4

After detections but before that the mitigation actions are activated and become effective the monitored parameter shows a saturation effect.

5

After the mitigation actions have become effective the monitored parameter rapidly decreases reaching again the normal state.

6

Normal state achieved after the effect of the mitigation actions.

In the bottom part of Figure 33, in orange we can see the absolute and normalized value of the comparison, indicated as Δ .

To implement the egress stage, STEP 3, of the proposed architecture, the research activities have also tested multiple different flavours of Supervised Machine Learning algorithms (e.g., Multivariate techniques, Linear Regressions, Random Forest, XGBoost, Decision Tree) with appreciable results that will be completed and presented in the successive version of this deliverable D3.2, and in the context of WP5.

STEP 3, the stage with the highest overall visibility, is able to learn and distinguish the overall normal network fluctuations of all the monitored variables from their attack indicating anomalies.

By the way, in doing so, it dynamically and run-time automatically follows and adapt its detection to each network evolution or traffic modification without any external setting (e.g., threshold tuning, parameter configurations etc.) improving its performances and making it perfectly suitable for a really one hundred per cent zero-touch solution.

6.1.2 The new regression algorithm

The Machine Learning process of dynamic updating the learnt model, which is undoubtedly a strength point as it automatically allows the forecasts to follow and adapt to evolutions in infrastructure, traffic, etc., constitutes, depending on the characteristics of the time series being analysed at input, an important weight in the total detection time.

For more details on the innovative regression algorithm conceived to reduce it (Patent WO2021190760) [47], please refer to Annex D.

6.1.3 Innovative aspects summarization

In compliance with the requirements outlined in the first part of this chapter, the threat detector has been designed to meet the requests of performances, scalability, speed, flexibility and automation.

The solution conceived and implemented presents several innovative aspects that distinguish it from the state of the art:

1. From the point of view of detection speed, in addition to multiple measures applied for this objective, an innovative algorithm has been implemented for the first time which exploits a weighted linear combination of concurrent regressions working on the original time-series $x(t)$, on its time-shifted versions $x(t-nT)$, and on their derivatives.
2. From the point of view of the detector architecture, a multi-stage ML solution has been identified which, although sometimes applied in the field of Image processing, is atypical in the field of CyberSecurity. This choice has proven particularly valid for its:
 - a. Flexibility - adequacy to detect multiple forms of attack, their combinations or evolutions, new forms of attack (0-day) etc.
 - b. Suitability for fully automated solutions without any need to set and periodically update complex threshold schemes or network configuration parameters, etc.

Additional important innovation aspects are related not to the detector component itself, but to the innovative patterns resulting from interactions with other blocks (e.g., Digital Twin). These aspects will be better described in the following version of this deliverable, D3.2, and in the WP5 context.

6.2 Threat Detector and Mitigation Engine Interfaces

6.2.1 Ingress Interface

This interface, indicated with number 1 in Figure 30, allows the DEME block to receive the input data from the preceding Smart Monitoring block, that continuously and capillary collects them from the 6G infrastructure, and Pre-processing that unifies and standardizes the collected data.

As previously explained, in the context of security, anomaly detection is a time-sensitive task. Operators ideally want to be alerted of potential breaches or system failures within minutes of suspicious signals. Every second counts when dealing with an attacker that is actively

exploiting a system. The other side of the coin is that thorough and very frequent data collection is resource-intensive and can be burdensome for the infrastructure.

For these reasons reasonable trade-offs indicates practical and typical values from 3 minutes to some tens of minutes [46].

The learning phase, considering a common value of 15 minutes, and a realistic training length of at least some thousands of samples, requires more than three weeks making the progressive mode unsuitable for practical real demonstrations.

For this reason, the DEME design and implementation has foreseen two separate micro-services to better separate the history data from the run-time data ingestion:

- DEME-engine: containing the threat detector with the complete logic and its API as described in the following paragraph. The DEME-engine provides different ways to input the history data (e.g., from Data source, from file).
- DEME-proxy: a useful utility microservice that can be used to interface the DEME-engine with a Kafka bus for run-time operations (consume data and publish outcomes).

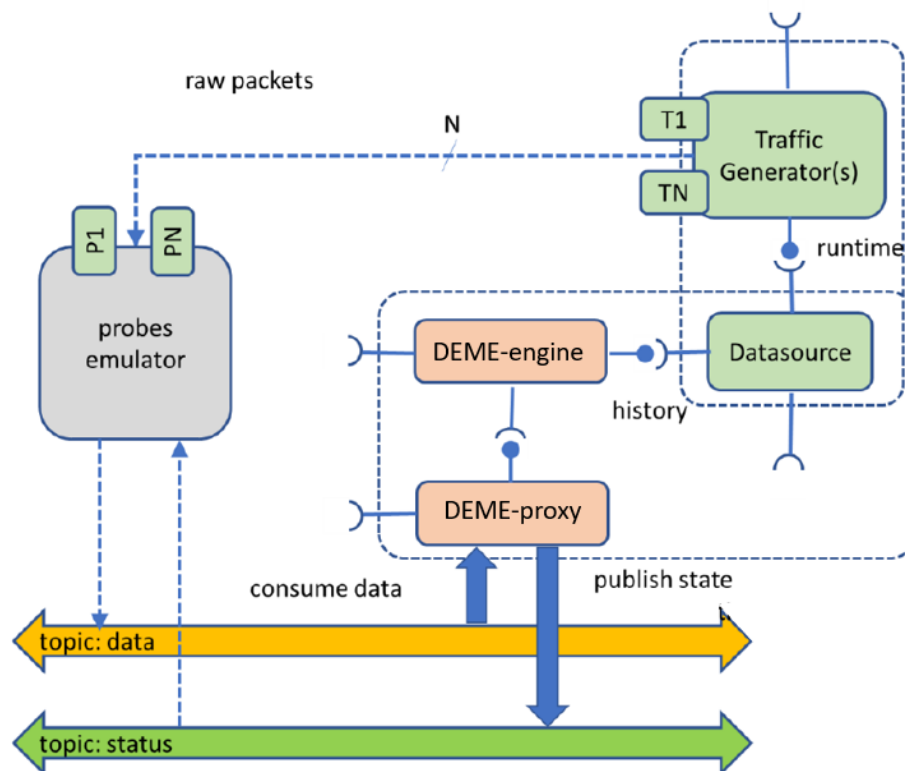


Figure 34. DEME micro-services

The data input format is composed, for each snapshot, by an initial timestamp, followed by the sequence of monitored data for each node composing the infrastructure topology. In the following Figure 35 we see an example with eight values per node and two nodes.

Time	Node id 0	Node id 1
26/11/2023 11:00:00	132, 257, 243, 223, 242, 133, 239, 230, 138, 253, 246, 225, 254, 183, 289, 250	

Figure 35. Ingress data format

6.2.2 Egress Interface

This interface, indicated with number 2 in Figure 30 allows the DEME microservice to provide the outcomes of its detections to the Distributed Trustable AI Engine (DTE). As previously presented in Figure 34, also in this case the DEME engine provides its own API, hereinafter detailed, but also a proxy utility that can be usefully applied for a straightforward connection with a Kafka bus.

The Detector outcomes are updated with the same rate of the ingress snapshots: i.e., in case of, for example, a 3-minute snapshot period from the Smart Monitoring block, the detector will update its outcomes each 3 minutes (a part a very small computational time).

Therefore, also in case of direct interface with the DEME engine API in asynchronous mode the outcomes are expected to be stable inside each ingress sampling period.

The DEME detector provides, for each node in the topology, the type of attack detected (if any), and the detection confidence (with a 0 to 1 value).

As described in the previous sections, multiple attacks (attacks combinations) acting same time to the same node are detectable (also providing a certain gain in terms of detection speed) and the same applies to new forms of attack as long as they can be identified by leveraging the space of the monitored variables.

6.2.3 Digital Twin Interface

The Digital Twin interface, indicated with number 3 in Figure 30, represents a high-level exchange of information between the two blocks that allows conceptually to mutually benefit from the knowledge acquired by the other block.

New forms of attacks or new, previously unseen, combinations detected by DEME will be usefully elaborated by the Digital Twin to complete the “what – if” analysis and, for example, provide suggestions for the best applicable mitigation action.

In turn, new hints from the Digital Twin may motivate changes, for example, in the DEME ingress variables space to increase the detector visibility on new unknown aspects.

This interaction will allow the Horse framework to evolve thanks to the continuous learning of new scenarios.

6.3 DEME Implementation overview

6.3.1 Background

This section describes how to check prerequisites, how to install the toolkit and how to verify its correct installation on your system. It assumes some working knowledge of shell scripting and docker.

We define these terms:

- Client: host from which commands will be run against the Server.
- Server: host where workloads are executed.

All commands are executed on the Client. It is assumed that Client and Server are separate hosts.

6.3.2 Prerequisites

- Docker installed on own machine.
- Move to the root folder of the toolkit in order to proceed with the installation.

6.3.3 Installation

- Launch the script to build the docker image of the server:

```
./build-engine.sh <version>
```

- Now the server is ready to be started by launching:

```
./run-engine.sh <version>
```

<version> is the minor version number.

- To verify that the server is correctly started send a GET request to the address *localhost:8090/is_ok* and check that the response is *{"message": "ok"}*

6.3.4 Interfaces

1. Management Methods

HTTP Method	Path	Action
GET	/is_ok	Healthcheck of Server
GET	/is_ready	Rediness of the Server
GET	/dump	Returns configuration parameters of the Server

2. Operational Methods

HTTP Method	Path	Action
POST	/estimate	Take in input a snapshot of network parameters and it elaborates the attack detection

3. Extraction Methods

These methods are meaningful just after running Operational Methods.

HTTP Method	Path	Action
GET	/ro	For all nodes it returns the accuracy of the detection of the identified attack.
GET	/ro_verbose	For all nodes it returns the accuracy in % of the detection of the identified attack in a verbose mode
GET	/teta	For all nodes it returns the type of the detected attack expressed as angle
GET	/teta_verbose	For all nodes it returns the type of the detected attack in a verbose mode
GET	/detection	For all nodes it returns the type and the probability of detected attack

6.3.5 Usage

In this section it is described an example of the usage of the toolkit. Since in this first iteration the toolkit is running in a standalone way, the history data and a snapshot of network features are locally loaded. So, in this procedure APIs will be sent using CURL command that will be replaced by the operations inserted in the next iterations.

- Start the Server
- Check it is ready using the GET request */is_ready*

```
curl -X GET http://localhost:8090/is_ready
```

If true, the server is ready.

- Send from the Client the POST request/estimate passing as parameter a json containing the snapshot of network parameters in a specific moment.

```
curl -H 'Content-Type: application/json' -d "@atk-root/data/test.json" -X POST http://localhost:8090/estimate
```

Request returns a “done” message when it ends the elaboration.

- To extract the detection related to the current sent snapshot send from the Client the GET request */detection*

```
curl -X GET http://localhost:8090/detection
```

The requests return a json response in this format:

```
[
  {
    "node": "<node>",
```

```

"detection": [
  {
    "attack": "<type>",
    "accuracy": <accuracy>
  }
]
    
```

6.3.6 Debugging tools

For debugging purpose, a simple web application has been developed.

For example, the user can set, using sliders, different combinations of network features. The system will predict in real time the probability of attack. This will be show in the related chart. When the probability of an attack exceeds a fixed threshold, the line of the chart becomes red to notify the user.



Figure 36. HORSE Real-time attack detection web.

In order to run the debugging tool:

- Start the toolkit Server and check it is ready.
- Move to the root folder of the debugging tool.
- Run `horse_debug_tool.html` using a browser.

7 Conclusions

This deliverable reflects a summary of the progress of the work done in Work Package 3. All the developments will be in continuous update also along with Work Package 5 and the Task Forces defined to better integrate the work of each partner into the project.

All the final developments and a total integration of the components explained here will be portrayed on the IT-2 document version of the Work Package 3.

References

- [1] Kubernetes <https://kubernetes.io/>
- [2] Kubernetes Network Emulator <https://github.com/openconfig/kne>
- [3] VR Network Lab <https://github.com/vrnetlab/vrnetlab>
- [4] EVE-NG <https://www.eve-ng.net/>
- [5] Prometheus, "Monitoring system & time series database", <https://prometheus.io/>
- [6] OTEL, "OpenTelemetry", <https://opentelemetry.io/>
- [7] Node Exporter, "Exporter for hardware and OS metrics", https://github.com/prometheus/node_exporter
- [8] KSM, "kube-state-metrics", <https://github.com/kubernetes/kube-state-metrics>
- [9] Alertmanager, <https://prometheus.io/docs/alerting/latest/alertmanager/>
- [10] Z. Xiang, S. Pandi, J. Cabrera, F. Granelli, P. Seeling and F. H. P. Fitzek, "An Open-Source Testbed for Virtualized Communication Networks," in IEEE Communications Magazine, vol. 59, no. 2, pp. 77-83, February 2021, doi: 10.1109/MCOM.001.2000578.
- [11] Comnetsemu, <https://git.comnets.net/public-repo/comnetsemu>
- [12] "Mininet website", <http://mininet.org>
- [13] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, Nick McKeown, "Reproducible Network Experiments Using Container-Based Emulation," CoNEXT'12, December 10–13, 2012, Nice, France.
- [14] D. Muelas, J. Ramos and J. E. L. d. Vergara, "Assessing the Limits of Mininet-Based Environments for Network Experimentation," in IEEE Network, vol. 32, no. 6, pp. 168-176, November/December 2018, doi: 10.1109/MNET.2018.1700277
- [15] ETSI TS 123 288 V17.4.0 (2022-05), 5G; Architecture enhancements for 5G System (5GS) to support network data analytics services (3GPP TS 23.288 version 17.4.0 Release 17)
- [16] A. Giannopoulos, S. Spantideas, N. Kapsalis, P. Karkazis and P. Trakadas, "Deep Reinforcement Learning for Energy-Efficient Multi-Channel Transmissions in 5G Cognitive HetNets: Centralized, Decentralized and Transfer Learning Based Solutions," in IEEE Access, vol. 9, pp. 129358-129374, 2021, doi: 10.1109/ACCESS.2021.3113501.
- [17] Synthetic Data Set for Network Data Analytics Function (NWDAF), https://github.com/sevgicansalih/nwdaf_data
- [18] 5GAD-2022 5G attack detection dataset, <https://github.com/IdahoLabResearch/5GAD>
- [19] 5GC PFCP Intrusion Detection Dataset, <https://ieee-dataport.org/documents/5gc-pfcp-intrusion-detection-dataset-0>
- [20] 5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network, <http://ieee-dataport.org/10203>
- [21] Flower – A Friendly Federated Learning Framework, <https://flower.dev/>
- [22] Samarakoon, Sehan, et al. "5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network." arXiv preprint arXiv:2212.01298 (2022).
- [23] OpenFGA, <https://openfga.dev/>
- [24] <https://github.com/Smile-SA/anonymization>

- [25] <https://github.com/dbader/schedule>
- [26] 5G-NIDD – A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless, <https://www.kaggle.com/datasets/humera11/5g-nidd-dataset>
- [27] MinIO, <https://min.io/>
- [28] Horse project consortium, D2.2 “HORSE Architectural Design (IT-1)”, September 2023
- [29] <https://www.astrid-project.eu/publications.html#AnDeliverables>
- [30] Sigma Telecom – How DDoS Affects Telecommunication security - <https://www.sigmatelecom.com/post/ddos-telecommunication-security>
- [31] Sam Cook - 20+ DDoS attack trends and statistics (2018-2023): The rising threat - <https://www.comparitech.com/blog/information-security/ddos-statistics-facts/>
- [32] Rohit Kharat, Avesh Mirza, - What metrics can you use to measure cyber-attack detection and response times? - <https://www.linkedin.com/advice/1/what-metrics-can-you-use-measure-cyber-attack-detection-4ymoc>
- [33] H. Vincent Poor, Princeton University, New Jersey, Olympia Hadjiliadis, Brooklyn College, City University of New York – Quickest Detection - Cambridge University Press - <https://doi.org/10.1017/CBO9780511754678>
- [34] Gene Yoo, Forbes Councils Member, The Importance Of Time And Speed In Cybersecurity, <https://www.forbes.com/sites/forbestechcouncil/2021/01/22/the-importance-of-time-and-speed-in-cybersecurity/?sh=7b4f9e7336a9>
- [35] A Pras, JJ Santanna, J Steinberger, A Sperotto, in International GI/ITG Conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance. Ddos 3.0-how terrorists ring down the internet (Springer, 2016), pp. 1–4
- [36] JJ Santanna, et al., in IFIP/IEEE International Symposium on Integrated Network Management (IM). Booters – An analysis of DDoS-as-a-service attacks, (Ottawa, 2015), pp. 243–251. <https://doi.org/10.1109/INM.2015.7140298>
- [37] Andrea Chebac - DDoS-as-a-service Attacks. What Are They and How Do They Work? - <https://heimdalsecurity.com/blog/ddos-as-a-service-attacks-what-are-they-and-how-do-they-work/#:~:text=Threat%20actors%20offer%20DDoS%2Das,also%20rump%20up%20the%20costs.>
- [38] Lily Hay Newman - Github ddos attack. www.wired.com/story/github-ddos-memcached/.
- [39] Sufian Hameed, Usman Ali - HADEC: Hadoop-based live DDoS detection framework - <https://jis-eurasijsournals.springeropen.com/articles/10.1186/s13635-018-0081-z>
- [40] RiskOptics – What is Cybersecurity Automation? - <https://reciprocity.com/blog/what-is-cybersecurity-automation/#:~:text=Cybersecurity%20automation%20is%20a%20concept,run%20faster%20and%20more%20efficiently.>
- [41] Giovanni Apruzzese, Michele Colajanni, Luca Ferretti, Alessandro Guido, Mirco Marchetti University of Modena, On the Effectiveness of Machine and Deep Learning for Cyber Security - <https://ieeexplore.ieee.org/document/8405026>
- [42] Executing multiple ML models in a chain - <https://learn.microsoft.com/en-us/windows/ai/windows-ml/chaining>
- [43] Mehrdad Hajizadeh, Thomas Bauschert, Technische Universität Chemnitz, Probability Analysis of Successful Cyber Attacks in SDN-based Networks - https://www.researchgate.net/publication/333501063_Probability_Analysis_of_Successful_Cyber_Attacks_in_SDN-based_Networks
- [44] Isabella Sansone, The Damaging Impacts of DDoS Attacks, <https://www.corero.com/the->

[damaging-impacts-of-ddos-attacks/](#)

[45] Akamai – Web Attacks and Gaming Abuse - <https://www.akamai.com/site/en/documents/state-of-the-internet/soti-security-web-attacks-and-gaming-abuse-report-2019.pdf>

[46] Clarence Chio, David Freeman – Machine Learning and Security – O'Reilly, ISBN: 9781491979907

[47] Gemme Luciano, Cappelli Marco, Ericsson Patent WO2021190760 “Determining an Alarm Condition” <https://www.patentguru.com/search?q=WO2021190760>

Appendix A: Meaningful use cases

Reflection/Amplification DDoS attacks

“Telecommunication biggest issue was and will be denial of service” [29]

Distributed denial of service (DDoS) attacks, are among the most common attacks that telecommunications companies must dodge on a daily basis [30]

The internet and telecom industry was among the industries that experienced the largest increase in DDoS attacks, up 210% on a yearly base [31]

The types of attacks that are most dangerous in network management are indeed the ones that can potentially paralyze the entire network, or a large part of it, and that require long recovery times impacting the activities of a very large number of users, and among these, in first place, there are undoubtedly the DDoS attacks.

DDoS attacks are still on the rise [44], and important actors in the CyberSecurity field like Norton calls them “one of the most powerful weapons on the internet”.

Denial-of-service attacks can come at any time, impact any part of infrastructures operations or resources, and lead to massive amounts of service interruptions and huge financial losses (for example in terms of remediation costs, lost revenue, lost productivity, loss of market share, and damage to brand reputation [44]).

For example, Yahoo!, who experienced one of the first major DDoS flooding attacks, saw their services offline for about 2h [29].

The same happened in occasion of the “largest ever” internet attack, when a 9 of the 13 domain name system (DNS) root servers were shut down for an hour-long DDoS flooding attack [30], or the notorious IoT device based DDoS attack that involved malicious DNS lookup requests from tens of millions of IP addresses that rendered major Internet platforms and services unavailable to large swaths of users in Europe and North America for several hours throughout the day [31].

DDoS attacks are characterized by a sharp increase in the malicious traffic with which the networks are flooded, and the network operators’ mitigation actions need to be very timely because these kinds of attacks are able, in a very short time, to paralyze the whole network making it not responsive.

Considering that any delay in detecting the flooding attacks risks making any mitigation action useless, these are the most challenging cases to effectively provide a near-real time attack detection able, at the same time, to face the explosive increase in the volume of Internet traffic and evolving sophistication of the attacks.

Among this family a particularly dangerous subset is undoubtedly represented by the reflection amplification attacks, mainly due to the fact that they are extremely cheap for hackers who exploit the victim’s own infrastructure (please note that the attack probability of occurrence is directly proportional to the attack cost coefficient [43]).

The same GitHub attack mentioned in Appendix C, as one of the largest known, was in fact precisely A DDoS reflection amplification attack coming from thousands of different autonomous systems and tens of thousands of unique endpoints.

This subtype of attack is a combination of two malicious factors.

First, the attacker simulates a request from the targeted server by putting its IP address into the request, ultimately using a public server as a “reflector.” The server receives the request indicating the targeted server and returns a response to it, thus “reflecting” the request. A lot of data can be requested, which means the response of the DNS server can become many times larger, and the ratio is indicated with “amplification ratio” or “gain ratio” (that in the GitHub attack case reached 51000, please refer also to Table 4). Finally, traffic is maximized by querying through a botnet so that the bandwidth of the targeted server is overloaded.

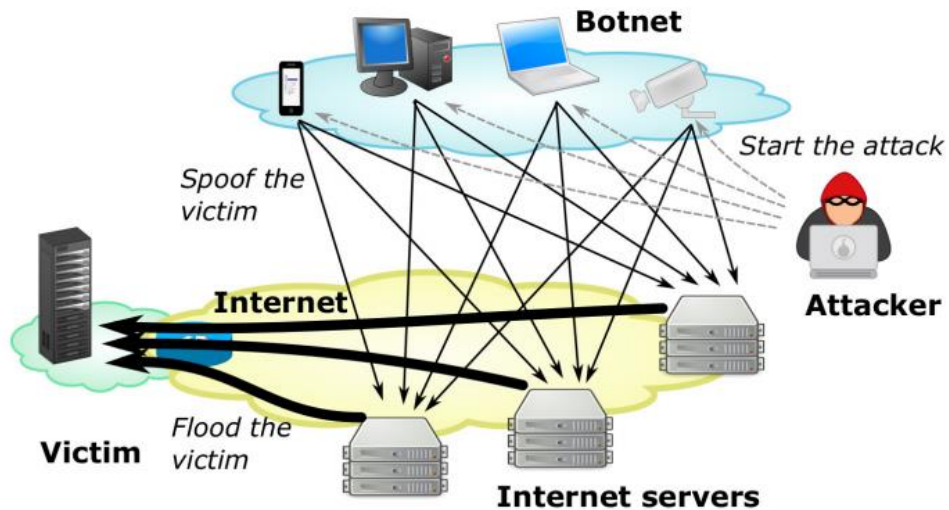


Figure 37. Reflection Amplification Attack Scheme

Table 4. Amplification factors

Protocol/Server	Amplification factor
Memcached	10000 to 51000
NTP	556.9
ChartGEN	358.8
QOTD	140.3
RIPv1	131.24
CLDAP	56 to 70
LDAP	46 to 70
DNS	28 to 54

Quake Network Protocol	63.9
TFTP	60
SSDP	30.8
MSSQL	25
Kad (P2P)	16.3
Portmap (RPCbind)	7 to 28
SNMP	6.3
Stream Protocol	5.5
NetBIOS	3.8
BitTorrent	3.8
Multicast DNS (mDNS)	2 to 10

Among the most interesting example cases, in addition to the GitHub worldwide famous DNS case, stands out the NTP one, first of all because it touches the network synchronization servers, that are crucial for the operation of the radio network, and secondarily because it implies a more complex flow analysis (i.e., NTP “MONLIST” packets, NTP type=7) that however can lead to detection speed improvements.

API Vulnerability Protection (e.g., RNAA attack)

According to a 2019 Akamai study [45], 83% of all web traffic was produced by APIs. Since APIs are now the simplest approach to provide functionality and data in an information system, this trend will likely continue.

But while the usage of APIs has expanded and produced more sophisticated apps that enhance productivity and use in the business world, it has also sharply increased the risk of cyberattacks. In fact, because of their exposure and "critical" role in handling sensitive data, they are easy targets for cyberattacks. Thus, a cybersecurity strategy's primary component needs to be API security.

The same level of attention must be paid not only to web applications but also to 5G/6G SBA Network function (e.g., Network Exposure Function (NEF) expose the 3GPP network capabilities via APIs).

The most common forms of API attacks comprise:

- DoS attacks trying to exhaust the limited resource that an API needs to respond to legitimate requests. By flooding an API with false requests, its resources are blocked from responding to those requests and not to others.
- Brute force attacks trying to test all possible combinations of a parameter, through a process of trial and error in the hope of guessing right. The objectives can be multiple: brute force of an authentication form to steal an account, brute force of a login to retrieve sensitive data, brute force of a secret, etc.
- Code injections: If attackers know the programming language used by an application or API, they can inject code through text input fields to force the web server to execute the desired instructions.

DoS signalling attack from 5G SMF/UPF (PCF) impacting on data plane & slicing

A particularly dangerous enhancement of this attack is its fusion with a variant of the PCF Flood Attack.

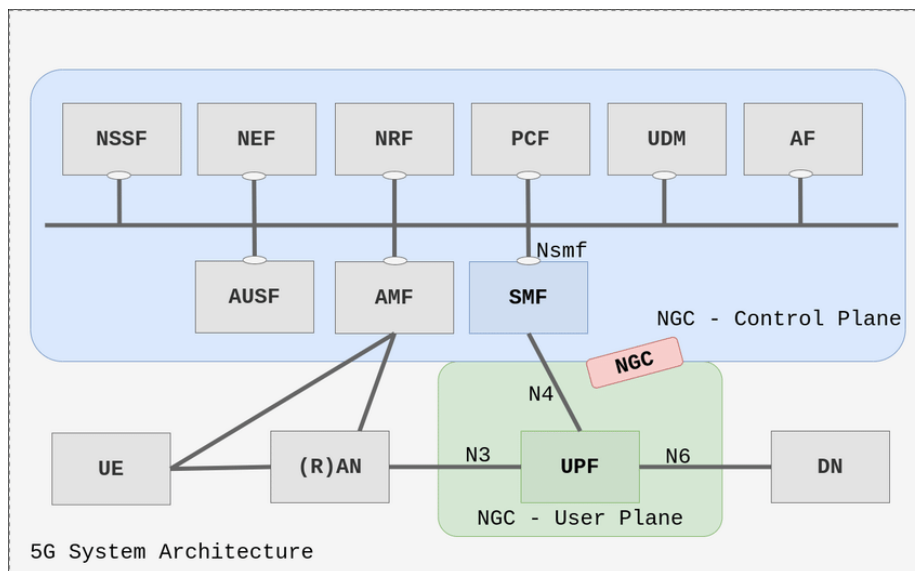


Figure 38. SMF / UPF N4

Assuming that a malicious user has gained access to the SMF NF and wishes, for example, to interrupt the connectivity of UEs without targeting a particular subscriber, they can run the session deletion attack numerous times with incrementally increasing SEIDs. As no other identifier is requested by PCF for the deletion of a session by UPF, a malicious SMF can instantiate a flood of session deletion request, carrying either random or increasing SEIDs. This allows the easy automation of attacks, as only a single identifier is required for the control of subscribers' sessions. This flood-based variation of the PCF Session Deletion attack is described by Algorithm 1 in Figure 39.

```

Algorithm 1 Unauthorised PFCP Session Deletion Request Flood
1: procedure MASSSESSIONDELETION ▷ Execution of the attack
2:    $SEID \leftarrow 0x1$  ▷ Initialization of the SEID value
3:    $SMFaddress \leftarrow SMFinwardsFacingInterfaceAddress$ 
4:    $UPFaddress \leftarrow SMFarpResponse$ 
5:    $delCounter \leftarrow 0x0$ 
6:   while  $SubscriberSessions = active$  do
7:      $pktPayload \leftarrow SessionDeletionRequest(SEID)$ 
8:      $SendRequest(src \leftarrow SMFaddress, dst \leftarrow UPFaddress, pktPayload)$ 
9:     if  $Cause.SessionDeletionResponse = "RequestAccepted"$  then
10:       $delCounter \leftarrow delCounter + 1$  ▷ Increment deletion counter by 1
11:       $SEID \leftarrow SEID + 1$  ▷ Increment SEID value by 1
    
```

Figure 39. Session Deletion Attack

Man in the Middle – TCP Session Hijacking Attack

The well-known background of the TCP Session Hijacking attack is based upon TCP packet spoofing. The TCP spoofed packets, whose signature matches that of an existing TCP session are accepted by the target machine if the four elements shown in Figure 40 match with the signature of the session and if provided with an acceptable sequence number.

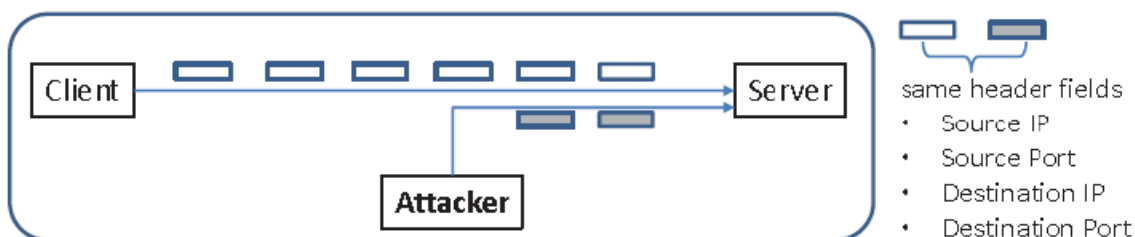


Figure 40. Injection data into a TCP connection

Appendix B. PAG Requirements

The functional requirements related to PAG, according to deliverable D2.1, are the following and are presented in this annex as a reminder to the reader:

REQ-F-23 - Access Management - The user must be able to define and then the HORSE platform must enforce access policies in real time and ensure that the information is only available to authorised users (applies to PAG and IBI).

REQ-F-25 - Granularity of the access - The HORSE platform should be able to support different roles in accessing the system (applies to PAG and IBI).

REQ-F-35 - Data anonymisation - The HORSE platform must execute data anonymisation operations on collected data assets.

REQ-F-36 - Data Encryption - The HORSE platform must support end-to-end data encryption for data in transit.

REQ-F-37 - Observability - The HORSE platform should allow the user to monitor the status (successful or failed execution) and view an incident summary of all AI pipelines.

REQ-F-38 - Data Retention - The user should be able to define and then the HORSE platform should execute data retention operations (e.g., automated deletion after a certain due date) on collected data assets.

Appendix C. KPI related to the detector performances

Before delving into the solution conceived and implemented specifically for the HORSE platform, it is certainly useful and appropriate to review the main requirements and KPIs of a modern, effective and high-performance threat detector.

Measuring the performance of a detection algorithm is crucial in order to evaluate its effectiveness and the specialized literature reports several types of metrics used for this objective:

- Detection Rate (DR)
- Precision (P)
- False positive rate (FPR)
- Accuracy rate (AR)
- F1 score (F1)
- Fitness function (F)
- Time to Detect (TTD)

DR: Detection rate, also known as recall, is the ratio of correctly predicted positive observations to all observations in the actual class

$$DR = \frac{TP}{TP + FN}$$

P: Precision is the ration of correctly predicted positive observations to the total predicted positive observations.

$$P = \frac{TP}{TP + FP}$$

FPR: False positive rate, also known as fall-out or false alarm ratio, is the ratio between the number of negative events wrongly categorized as positive and the total number of actual negative events.

$$FPR = \frac{FP}{FP + TN}$$

AR: Accuracy rate is simply a ratio of correctly predicted observation to the total observations.

$$AR = \frac{TP + TN}{TP + TN + FN + FP}$$

F1: Weighted Average of detection rate and precision.

$$F1 = 2 * \frac{DR * P}{DR + P}$$

F: Fitness is a statistical metric used to evaluate the accuracy of the developed solution. A detection system will determine which subset of features are the best according to the fitness:

$$F = 2 * \frac{\alpha * DR}{\beta - (100 - FPR)}$$

Where:

- True positive (**TP**) is the number of attacks that have been classified properly.
- True negative (**TN**) is the number of normal records that have been classified properly.
- False positive (**FP**) is the number of normal records that have been classified as attacks.
- False negative (**FN**) is the number of attacks that have been classified as normal.
- $\alpha(0,1]$ e $\beta=1-\alpha$ are two parameters that respectively specify the importance between DR and FPR.

KPI related to the lifecycle of a cyber attack

The lifecycle of a cyber-attack is referred to as the time which has elapsed between the initial detection and containment of the breach or attack. The detection time, or time to detect (TTD), is the length of time that it takes for a business to identify a cyber incident has occurred, whilst the response time, or time to respond (TTR), is how long it takes to restore networks or services once a cyber incident has been initially detected.

In cybersecurity, speed defines the success of both the defender and the attacker. Professional cybercriminal groups, nation-state actors and advanced persistent threat groups are evolving, as are their tools, tactics and procedures and nowadays only need seconds to exploit a vulnerability.

Such situations took place, for example, in winter 2019 when threat actors were mass-scanning the internet for network hosts with particular VPN products having a critical vulnerability to perform remote code execution (RCE). In minutes, thousands of enterprises were compromised. [34]

Some specific, but unluckily very common forms of attacks (e.g., the availability resources / DDoS ones), are characterized by a sharp increase in the malicious traffic with which the networks are flooded, and the network operators' mitigation actions need to be very timely because these kinds of attacks are able, in a very short time, to paralyze the whole network making it not responsive (please see, for example, Figure 41).

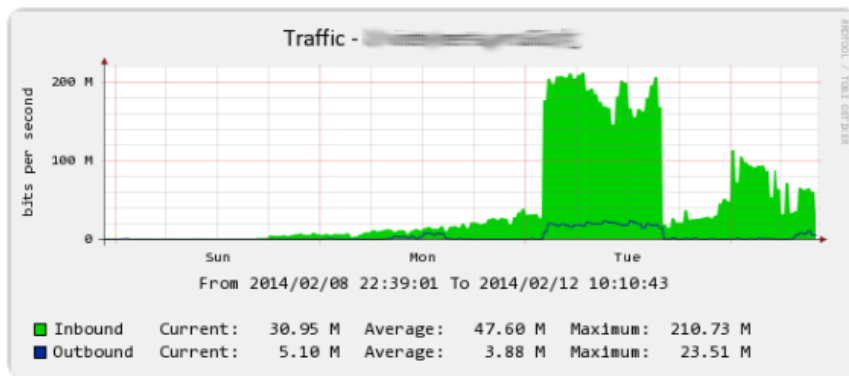


Figure 41. Real case of an NTP amplification DDoS attack

Considering that any delay in detecting the attack risks to make any mitigation action useless, the challenge is to effectively provide a near-real time attack detection able, at the same time, to face the explosive increase in the volume of Internet traffic and evolving sophistication of the attacks.

TTD : Time to detect is one of the most important metrics for cyber attack detection. It measures how long it takes to discover an incident. The shorter the TTD, the faster it is possible to contain and remediate the attack, and the lower the potential damage. A rapid response plays a pivotal role in mitigating the potential damage caused by cyber attacks [32] [33].

TTR : Another key metric for cyber attack response is the time to respond (TTR), which measures how long it takes for the system to take action after detecting an incident. The shorter the TTR, the more effective the response strategy, and the higher the chances of minimizing the impact of the attack. To calculate the TTR, the system need to track the time from when the incident was detected to when it was resolved or closed [32] [33].

Additional requirements / KPI

- Scalability**: Recent analysis reveals that with little effort, next generation cybercrime tools would be able to enact attacks that are thousand times stronger than the ones we have seen so far [34]. One of the major concerns is that performing very effective attacks (as the DDoS ones) is extremely simple with websites known as Booters or Stressers that offer them as a service. These booters provide cheap services, and the costs to perform a series of attacks is typically just a few dollars [35].

DDoS-as-a-service, for example, is part of the cybercrime-as-a-service model and implies a hacker providing DDoS (Distributed Denial of Service) attacks for money. The vendor usually owns a botnet and advertises his services on the Dark Web. The buyer – who can be another hacker or a random individual – selects the target, the type, and the duration of the attack. The fee they agree upon is most of the time paid in cryptocurrency. The anonymity of this transaction is guaranteed, as there is no contact between the hacker and the buyer. [36]

A further example of how impressive the modern attacks that hackers are able to carry out can be, we can mention reflection amplification attacks.

Recently, GitHub was hit by 1.35 Tbps of traffic all at once [37].

It was one of the most powerful attacks in the history of DDoS, no botnet was required to achieve such high traffic volume. The attackers spoofed GitHub's IP addresses and took control of its memcached (a distributed memory system known for high performance and demand) instances. The memcached systems then return 50 times the data of the requests back to the victim.

GitHub was lucky enough to afford a robust DDoS mitigation service (Akamai Prolexic) and the assault dropped off after 8 min but it is now crystal clear that the explosive increase in the volume of Internet traffic and the sophistication of modern attacks have posed serious challenges on how to implement threat detections in a scalable and accurate manner. [38]

- **Automation:** Cybersecurity automation is closely linked to reaction speed, because leveraging on it organizations deal with (and disarm) cyber threats before those threats can disrupt their operations. Cybersecurity automation takes human-driven and repeatable tasks that could be handled by the devices without human interaction and automates that work. Put another way, cybersecurity automation streamlines manual and time-consuming tasks into automated workflows, making network security processes more efficient and less prone to human error. With enhanced efficiency, faster decisions can be made, which also can improve an organization's entire security posture. For example, automation can monitor and scan networks for security loopholes and potential vulnerabilities and generate reports that the security teams can use to assess the severity of the issue and determine a solution to mitigate it, can monitor compliance states making easy for organizations to identify and solve potential compliance problems, and, above all, can automate the process for responding to security incidents. Automated incident response systems use pre-planned and custom rules to respond to an incident, all without human intervention. This can help organizations respond to incidents more quickly and reduce the overall impact of a security incident. Other benefits of automated incident response are optimized threat intelligence, streamlined operations, and automated reporting and metrics capabilities [39]. The advantages of using automated cybersecurity systems include:
 - **Increased efficiency.** Cybersecurity automation allows for the rapid detection and response to potential threats, reducing the time it takes to mitigate them.
 - **Improved accuracy.** Automated systems can process massive amounts of data and uncover patterns that may be difficult for humans to discover, leading to fewer false positives or negatives.
 - **24/7 monitoring.** Automated systems can monitor networks and systems continuously for potential threats, providing round-the-clock protection.
 - **Scalability.** Automation can be used to scale security operations to satisfy the requirements of organizations of all sizes, allowing for more effective security management while keeping costs as low as possible.

The HORSE threat detector aims, as described below, to further enhance the level of automation to automatically learn the characteristics of the network and its traffic dynamically and continuously, improving, in this way, the overall performance and not requiring dedicated parameter configurations by cybersecurity experts.

- **Flexibility/Upgradability:** Products based on machine learning are often promoted by vendors as catch-all solutions to a broad array of cyberattacks. However, unbiased experimental results have shown so far that ML algorithms may provide superior performance when they focus on specific threats instead of trying to detect multiple threats at once [40]. The other side of the coin is that specific detectors present some important weaknesses:
 - Slower detection in case of combined attacks, due to their limited visibility.

- In spite of the fact that ML based approach are theoretically preferable on older ones for their detection speed, in case of new form of attacks, zero-day, the necessity to design, implement, train and tune new specific detectors makes in these cases the specific detectors solutions unresponsive and ineffective.

Based on these considerations, a modern detector must be designed to be easily upgradable and to be easily allocated in plug-and-play mode in a framework suitable to accommodate further detection services over time according to the evolution of Cybercrime techniques.

Furthermore, it must be provided with maximum visibility to be able to identify both specific attacks and their combinations, as well as the maximum possible number of new forms of attacks, zero-day.

Appendix D. Threat detector innovative ML algorithm

Studying in detail the different impacts of various computations in the overall timing budget, it has been demonstrated that the replacement of standard Machine Learning regressions (e.g., Linear, ARIMA, SARIMA etc.) with a weighted linear combination of concurrent regressions working on the original time-series $x(t)$, on its time-shifted versions $x(t-nT)$, and on their derivatives (i.e., interpolations of their residuals, defining residuals the differences among the received and predicted values) is able to provide an important gain in terms of detection speed [47].

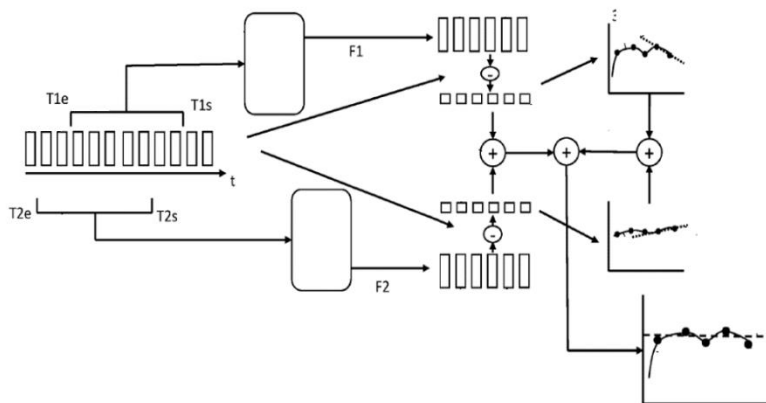


Figure 42. Weighted linear combination of concurrent regressions [47].

Figure 43, from top to bottom, presents:

- Real data (orange), ARIMA forecasts (green [original time-series] and yellow [time shifted time-series])
- Δ_{Tot} Overall result of the overall linear combination (plus 200 to plot it)
- A warning-signal elaborated starting from Δ_{Tot} (red) (plus 150 to plot it)
- Difference between regressions forecasts and real values (green and yellow) (plus 100 to plot it)
- Derivative functions of difference series (green and yellow) (plus 50 to plot it)

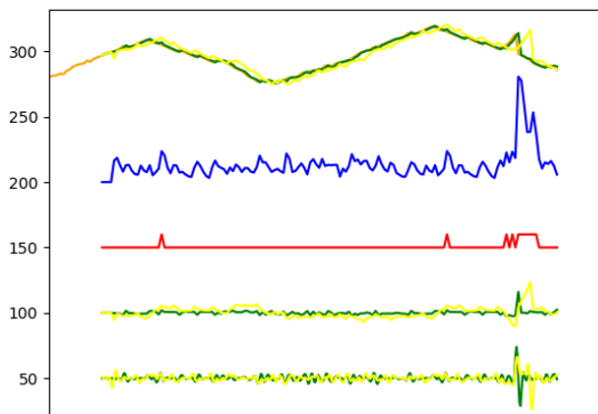


Figure 43. Waveforms

It is possible to appreciate the very low latency from the anomaly condition to the warning signal in comparison to a standard regression applications, Figure 44.

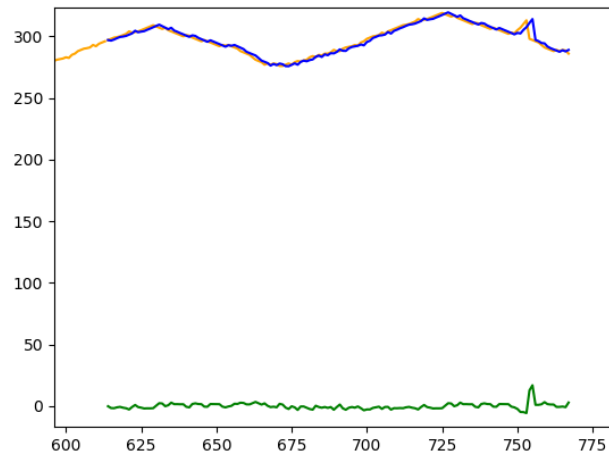


Figure 44. Standard regression detection time.