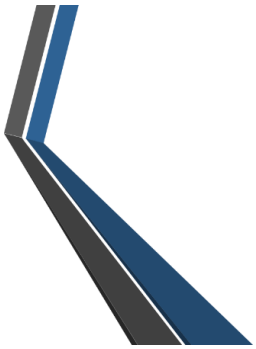

Net-Zero self-adaptive activation of distributed self-resilient augmented services

D3.3 Intent-based service security and adaptive security techniques.r1

Lead beneficiary	ZHAW	**Lead author**	Gürkan Gür
Reviewers	Edgardo Montes de Oca (MONT), Vincent Lefebvre (TSS)		
Type	R	**Dissemination**	PU
Document version	V1.0	**Due date**	30/09/2025

Co-funded by the European Union

6GSNS

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

UK Research and Innovation

Project information

Project title	Net-Zero self-adaptive activation of distributed self-resilient augmented services
Project acronym	NATWORK
Grant Agreement No	101139285
Type of action	HORIZON JU Research and Innovation Actions
Call	HORIZON-JU-SNS-2023
Topic	HORIZON-JU-SNS-2023-STREAM-B-01-04 Reliable Services and Smart Security
Start date	01/01/2024
Duration	36 months

Document information

Associated WP	WP3
Associated task(s)	T3.3
Main Author(s)	Gürkan Gür (ZHAW)
Author(s)	Wissem Soussi, Gökcan Cantali (ZHAW), Péter Vörös, Mohammed Alshawki (ELTE), Antonios Lalas, Stelios Mpatziakas, Virgilios Passas, Sarantis Kalafatidis, Nikolaos Makris, Donatos Stavropoulos, Thanasis Korakis, Anastasios Drosou (CERTH), Sumeyya Birtane, Shankha Gupta, Mays AL-Naday (UEssex), Maxime Vinh Hoa La (MONT)
Reviewers	Edgardo Montes de Oca (MONT), Vincent Lefebvre (TSS)
Type	R – Document, Report
Dissemination level	PU – Public
Due date	M21 (30/09/2025)
Submission date	30/09/2025

Document version history

Version	Date	Changes	Contributor (s)
v0.1	15/06/2025	Initial table of content (ToC)	ZHAW (Gürkan Gür)
v0.2	18/06/2025	Updated ToC and partner assignments	All authors
v0.3	20/06/2025	Contribution for Section 2	ZHAW
V0.3.1	20/07/2025	Contribution for Section 3 and 5	MONT, ELTE
v0.4	13/09/2025	Contribution for Section 4 and 6	UESSEX, Virgilios Passas, Stelios Mpatziakas (CERTH)
v0.5	17/09/2025	TL and WPL-reviewed version	Gürkan Gür (ZHAW)
v0.6	19/09/2025	Reviewed version	Edgardo Montes de Oca (MONT), Vincent Lefebvre (TSS)
v0.7	24/09/2025	Refinement based on review comments	All authors
v0.8	27/09/2025	Quality review	Joachim Schmidt (HES-SO), Leonardo Padial (HES-SO)
v0.9	29/09/2025	Final review and refinements	Antonios Lalas, Virgilios Passas, Stelios Mpatziakas (CERTH) and CERTH team
v1.0	30/09/2025	Final version for submission	Antonios Lalas (CERTH)

Disclaimer

Copyright message

Content

Content **5**

List of acronyms and abbreviations **7**

List of figures **8**

List of tables **8**

Executive summary **9**

1. Introduction **10**

1.1. Purpose and structure of the document *10*

1.2. Intended Audience *11*

1.3. Interrelations *11*

2. Resource-optimized MTD **12**

2.1. Live Migration Optimizer *12*

2.2. Broader MTD Strategy Optimization *14*

3. XAI-Driven Intent-Based Security Monitoring and Enforcement **17**

3.1. Intent-Based Networking *17*

3.2. XAI for Intent Verification and Policy Enforcement *21*

3.3. Adaptive Security in Multi-Cloud and Dynamic Environments *23*

4. Adaptive orchestration optimisation for sustainability **26**

4.1. CTI-Driven Selective Sharing *26*

4.2. Workload Prediction for Scheduling *27*

5. Secure Data Aggregation **29**

5.1. Secure Data Aggregation Protocol *29*

5.2. Attack Resilience *32*

5.3. Aggregation Integration *33*

6. Microservice profiling and anomaly detection - Microservice behavioural analysis **34**

6.1. Technical Description *34*

6.1.1. Algorithms utilized for proposed solution 35

6.2. Microservice Profiling Evaluation Results *37*

6.3. AI driven detection mechanism 39

7. Conclusion 43

References 44

Annex A Details on the architectures of the DNN used for Microservice profiling 46

List of acronyms and abbreviations

Abbreviation	Description
AI	Artificial Intelligence
ARP	Address Resolution Protocol
CNF	Containerized VNF
CPU	Centralized Processing Unit
CTI	Cyberthreat Intelligence
CRIU	Checkpoint Restore In Userspace
DoST	Denial of Sustainability
FL	Federated Learning
GDPR	General Data Protection Regulation (GDPR).
IDS	Intrusion Detection System
IoT	Internet of Things
LiMi	Live Migration
MPC	Multi-Party Computation
MOMDP	Multi-Objective Markov Decision Process
MTD	Moving Target Defense
NS	Network Slice
RL	Reinforcement Learning
SA	Secure Aggregation
SLA	Service Level Agreement
SSH	Secure Shell
STIX	Structured Threat Information Expression
VNF	Virtual Network Function
XAI	Explainable AI

List of figures

Figure 1: Container live migration setup 13
Figure 2: Data collected and used in the workflow of the MTD Strategy Optimizer. 15
Figure 3: An example of a slice isolation SSLA 20
Figure 4: A rule sample to monitor the network traffic in a slice and to act as a firewall. 21
Figure 5: XAI-driven Intent Verification and Policy Enforcement 22
Figure 6: Adaptive Security Framework in Multi-Cloud and Dynamic Environments 24
Figure 7: High-level functional view of the CTI selective sharing process 26
Figure 8: Secure Data Aggregation Process 30
Figure 9: MPC-based aggregation 31
Figure 10: Position of the microservice behavioural analysis module and interconnection to other modules 35
Figure 11 High level architecture of the proposed solution 37
Figure 12: CPU consumption forecast example for one of the containers contained in the dataset (step t=10) 41
Figure 13: Memory Consumption (RSS) forecast example for one of the containers contained in the dataset (step t=3) 42

List of tables

Table 1 Evaluation results 37
Table 2 Results of the lightweight CNN that performs binary classification of the microservices resource usage (Normal/Anomalous) 38
Table 3 Results of the CNN that performs multiclass classification of the microservices resource usage (7 classes: Normal/Five Anomalies/Unknown) 39
Table 4 Common actions used to mitigate the microservice anomalies examined. 39
Table 5 Features contained in [13] and utilized for forecasting resource prediction 40

Executive summary

As 6G networks evolve to support highly dynamic, complex environments and diverse services, ensuring robust and adaptive security becomes essential. This deliverable presents recent advancements within the NATWORK project as part of Task 3.3 (T3.3), focusing on the integration of AI-driven techniques for intent-based security and smart Moving Target Defense (MTD). Moreover, these innovations enable adaptive orchestration optimized for sustainability and support the development of dynamic defense mechanisms tailored for secure microservices. Collectively, these efforts contribute to NATWORK's overarching objective: delivering resource-efficient, intelligent, and adaptive security solutions for the 6G continuum.

1. Introduction

Adaptive and intent-based security is crucial for envisaged 6G networks since they will operate in a very dynamic and complex environment while providing a multitude of services, leading to a large and dynamic attack surface [1]. To address these challenges, one of the technical pillars of the NATWORK project is to contribute to the dynamic, footprint-ideal orchestration and management of secure, complex 6G services over the continuum. In that regard, this deliverable reports on the recent advances in the framework of NATWORK T3.3 to integrate AI-driven techniques for ensuring smart MTD and intent-based security management, to enable adaptive orchestration with optimisation for sustainability, and to develop dynamic defence techniques for secure microservices. Overall, these efforts are serving the NATWORK goal of resource-optimised, adaptive, and smart security functions for 6G continuum.

1.1. Purpose and structure of the document

The purpose of this document is to provide a concise yet encompassing description of the NATWORK contributions in T3.3 towards dynamic, smart and sustainable security for 6G networks and edge-to-cloud continuum. It describes the work regarding Intent-based service security and adaptive security techniques.

Following the Introduction, which sets the stage for the document's purpose, audience, and its interconnections within the project's framework, the structure continues as follows:

Sections:

1. **Section 2 Resource-Optimised MTD:** Describes the NATWORK T3.3 efforts on optimized Moving Target Defence (MTD) and presents the Live Migration Optimizer and MTD strategy optimization work for adaptive and policy-based MTD.
2. **Section 3 XAI-Driven Intent-Based Security Monitoring and Enforcement:** presents the XAI driven monitoring and enforcement framework for intent-based security operation.
3. **Section 4 Adaptive orchestration optimisation for sustainability:** Presents the project's work on adaptive orchestration and its optimisation for sustainability and cybersecurity, and elaborates on two main contributions, namely, CTI-driven selective sharing and workload prediction for scheduling, in that domain.
4. **Section 5 Secure Data Aggregation:** elaborates on secure aggregation techniques which are relevant for Federated Learning (FL) based smart security schemes in 6G networks.
5. **Section 6 Microservice profiling and anomaly detection, and microservice behavioural analysis:** Presents profiling and anomaly detection schemes being developed for adaptive defence in the NATWORK B5G architecture. They also allow microservice behavioural analysis for cybersecurity purposes.

6. **Conclusions:** Wraps up the document, reflecting on the project's strategic orientation and establishing expectations for future milestones.

1.2. Intended Audience

The NATWORK Deliverable *D3.3 Intent-based service security and adaptive security techniques* is for Public Dissemination. It is there devised for the internal and external use of the NATWORK consortium, comprising members, project partners, affiliated stakeholders and the public. This document mainly focuses on the Intent-based service security and adaptive security aspects of the project, thereby serving as a referential tool throughout the project's lifespan.

1.3. Interrelations

The NATWORK consortium integrates a multidisciplinary spectrum of competencies and resources from academia, industry, and research sectors, focusing on user-centric service development, robust economic and business models, cutting-edge cybersecurity, seamless interoperability, and comprehensive on-demand services. The project integrates a collaboration of fifteen partners from ten EU member states and associated countries (UK and CH), ensuring a broad representation for addressing security requirements of emerging 6G Smart Networks and Services in Europe and beyond.

NATWORK is categorized as a "Research Innovation Action - RIA" project and is methodically segmented into 7 WPs, further subdivided into tasks. With partners contributing to multiple activities across various WPs, the structure ensures clarity in responsibilities and optimizes communication amongst the consortium's partners, boards, and committees. The interrelation framework within NATWORK offers smooth operation and collaborative innovation across the consortium, ensuring the interconnection of the diverse expertise from the various entities (i.e., Research Institutes, Universities, SMEs, and Large industries) enabling scientific, technological, and security advancements in the realm of 6G. The *D3.3 Intent-based service security and adaptive security techniques* addresses activities of the NATWORK project related to the design, development, and validation of intent-based service security and adaptive security techniques and mechanisms. As a core technical WP, it relies on the architectural work carried out in WP2, linked with the other security and advancements from WP3, and AI-driven management solutions from WP4. It will feed the integration and validation efforts within WP6, for evaluating and improving the assets presented in this deliverable. That coherent structure entailing the work in other project activities and the adaptive and intent-based service security work described in this deliverable ensures consistency and alignment across the project's technical pillars.

2. Resource-optimized MTD

As defined in the previous deliverable of this work package (*i.e.,* D3.1), the MTD framework designed and implemented in this project enhances the security of network functions across the edge-to-cloud continuum via proactive and reactive MTD operations such as live migration and re-instantiation of VNFs and CNFs. Deliverable D3.1 described the operations of the **MTD controller** component and its function of live migrating CNFs running as pods in Kubernetes environments (*c.f.,* Section 4.1 and 5.7 of D3.1), while this deliverable focuses on the **MTD Strategy Optimizer**, the second component of the MTD framework, determining how to live migrate a container and which CNF/VNF to be migrated. Both decisions are made using AI-powered decision agents implemented to enhance the efficiency and efficacy of MTD operations. In the following sections, we describe the two main tasks of the MTD strategy Optimizer component, namely, a live migration optimizer (*c.f.,* Section 2.1) and a broader MTD strategy optimization (*c.f.,* Section 2.2).

2.1. Live Migration Optimizer

The MTD framework performs parallel live migration (LiMi) of containers and microservices and uses the following elements depicted in Figure 1:

- **LiMi client**, which is the application on the source side of the migration, performing the checkpoint and delta-updates of the containers to be migrated.
- **LiMi server**, which receives the checkpoints from the LiMi client and restores them to running containers.
- **L2 networking bridge**, which is used to keep the same allocation of IP address to the containers, even when they migrate to a different node or cluster. This enables the continuation of existing end-to-end sessions even after migration.

During migration, both the LiMi server and client establish a secure SSH tunnel to ensure end-to-end encryption of the container's checkpoint during transfer. The transfer process utilizes rsync, which is optimized to transmit only the incremental differences between the new checkpoint and other possibly existing images of the container at the destination.

The LiMi controller supports four live migration methods: 1) basic or cold migration, 2) pre-copy migration, 3) post-copy migration, and 4) hybrid migration [12]. LiMis are performed via the LiMi client, which directly interfaces with runC, a low-level container orchestrator, and its CRIU integration, used to initiate container checkpointing and restoring for live migrations. The LiMi server also operates through the runC interface to receive the eventual pre-dumps, dumps, and post-copy memory pages from the LiMi client and to restore the container at the destination

node. For stateful migrations, CRIU is configured to preserve TCP-established sockets, allowing existing connections between the container and its clients to remain intact.

Since TCP sessions are defined by IP address and port pairs, the L2 bridge module ensures the migration of the container's IP address at the Address Resolution Protocol (ARP) level. This is critical for maintaining session continuity and minimizing availability disruptions for services that rely on persistent connections, such as databases, SSH sessions, and voice-over-IP (VoIP) applications. Finally, the LiMi client reduces the LiMi downtime by transferring in parallel both the dump/checkpoint delta and the volume delta for container applications with writable root filesystems.

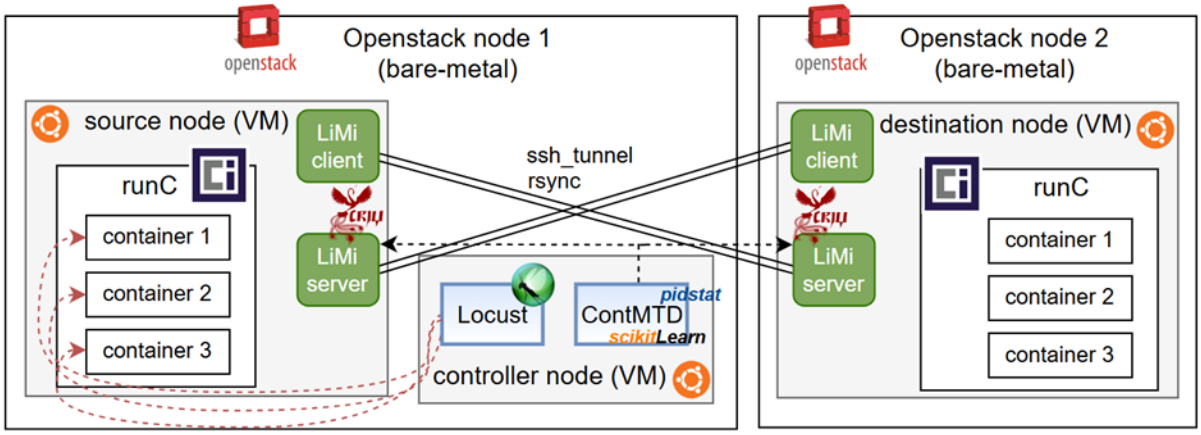

Figure 1: Container live migration setup

In this setup, The MTD Strategy Optimizer uses a ML-based classifier and a ML-based regressor to optimize containers LiMis. Both ML models are trained by using a constructed dataset collecting over 6000 live migrations of containers of three load levels (low, medium, and high) on four types of resources (CPU, RAM, storage, and networking), to characterize the dependency between resource load type and LiMi performance. As networking has an additional type of load (*i.e., zero),* the container categorization space spans to 108 for this configuration (*i.e.*, the combination of load levels for all resource types). However, some combinations are impossible to obtain in practice (*e.g., high RAM and storage read/write operations require medium to high CPU load),* reducing the possible combinations to 34 for this configuration. All combinations of containers are empirically tested using a dynamic containerized application to cover the different loads using the Linux cgroups kernel function set at the container runtime level.

The dataset is formed by performing LiMis on the dynamic containerized application with the 34 resource combinations, using for each combination the different migration algorithms (basic, pre-copy, post-copy, and hybrid). The distribution of the four migration methods per container category is kept uniformly distributed, extracting the following features from each migration: total migration time, pre-dump time and pre-dump size, pre-dump transfer time, dump time and

dump size, dump transfer time, volume size and volume transfer time (the size considers delta optimization of sync transfer), dump and volume deltas parallel transfer time (which also includes the layer 2 level IP transfer), restore time, approximate container downtime (calculated by summing dump time, dump transfer time and restore times), and a precise downtime measured by the locust traffic generator (including the additional downtime from packets queuing). This data is used to train the ML classifier and migration time regressor used in the ContMTD workflow described below.

The ML classifier selects the best LiMi method per container to minimize the migration time and service downtime of the container, while the ML-based regressor is used to estimate the migration time of each container. This estimation is used to schedule the parallel migration of interdependent CNFs running in a broader network service (NS), where migrating a NS requires the parallel migration of its microservices. This scheduling is made to have the interdependent migrating CNFs reach their destination at the same time, minimizing the disruption of the NS migrated.

2.2. Broader MTD Strategy Optimization

When performing MTD actions proactively, no concrete attack is occurring, and actions are performed to statistically reduce the probability that a threat occurs. Thus, there is an inherent optimization problem of finding the right trade-off between three objectives: **1)** increasing security, **2)** decreasing operational costs of MTD, and **3)** decreasing the QoS overhead of protected CNFs/VNFs [13]. On the other hand, applying too many migrations will result in extensive computational overhead and may cause longer service disruption for the users. To conform to service level agreements (SLA) of the service provided, *e.g. an agreed minimal service availability of 99.95%,* a budget (i.e. quota) of MTD operations for each CNF is calculated and defined. The problem in this case is to optimize the periods of MTD actions in such a way that the security is maximized while the quota is not exceeded.

To quantify the reach of the MTD strategy to each of these three objectives, the MTD Strategy Optimizer is composed of the following modules: the risk assessment (RI.AS.) module, the modelling module using multi-objective Markov Decision Process (MOMDP), and the deep-RL agent [14] optimizing and deciding on the MTD policy. Finally, the MTD Strategy Optimizer is interfaced with the MTD controller, requesting some of the near-real-time data gathered by the latter, and then sending to it the decisions of MTD actions to be enforced.

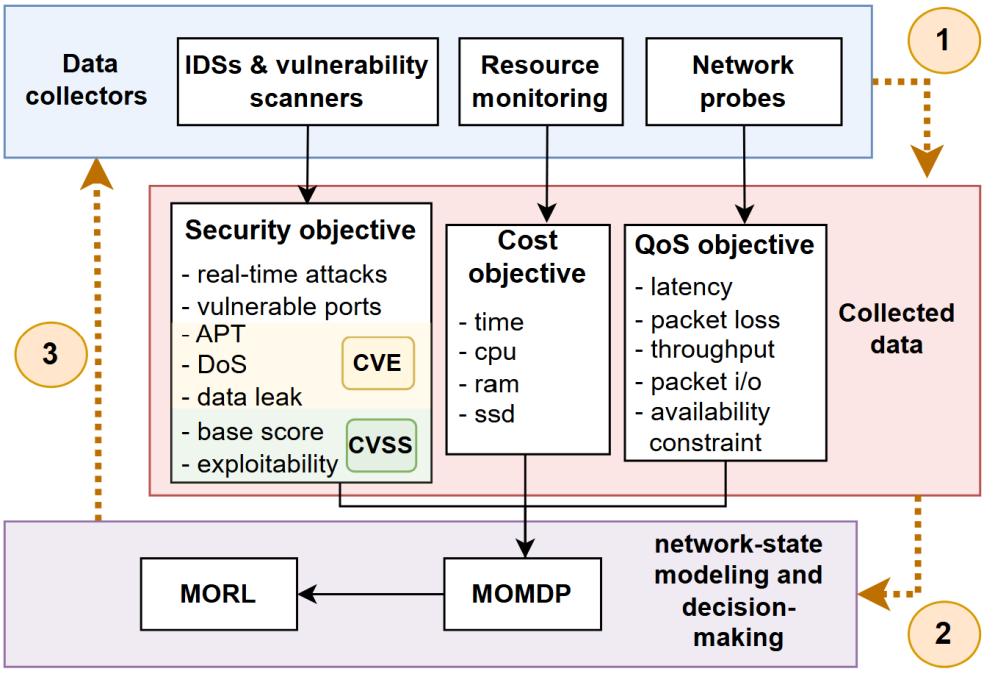

Figure 2: Data collected and used in the workflow of the MTD Strategy Optimizer.

Figure 2 details the workflow and data gathered for the MTD strategy optimization. For the security objective, a risk assessment is performed to collect metrics for proactive estimation of the Attack Success Probability (ASP). Daily threat analysis on all running VNFs/CNFs is performed using a vulnerability scanner, OpenVAS, to identify running services, using the Common Platform Enumeration (CPE), and perform active and passive vulnerability scans, detecting Common Vulnerability Enumerations (CVE), i.e., identified vulnerabilities registered in the National Vulnerability Database (NVD) [22]. The ASP is then estimated using the CVSS base score and exploitability score of the vulnerabilities found with the risk assessment [23].

For the operational cost objective, an empirical study of the cost of virtual resources is done to find the coefficients between CPU cost, RAM cost, and storage cost, based on the definition of:

$$resource_{cost} = \beta + \alpha_1 \times cpu + \alpha_2 \times ram_{gb} + \alpha_3 \times storage_{gb}$$

measured in USD per hour ($\$/hour$), reference to cloud providers' convention as a measurement unit for virtual resources used in their cloud. The prices of over 70 VM offers are collected from four major cloud providers: AWS, Azure, Google Cloud, and OVH. A cloud provider provides VMs with different prices for the same resource depending on the hardware used (*e.g.,* Intel vs. AMD cores). Thus, the coefficients are not found with perfect equations but calculated using linear regression, which gives the following statistically significant (*i.e.,* $p<0.01$) results: β = -0.082, α_1= 0.031, α_2= 0.004, α_3= 0.00006.

For the QoS objective, the MTD Strategy Optimizer collects network metrics for every protected VNF such as the number of user equipment (UE) connected, connection latency, connection throughput, and packet loss-rate, and the number of packets flowing in and out. From these monitored values, we derive the mean packet loss rate increase and the mean latency increase caused by the MTD actions.

Finally, collected data is used to model the network state observation with an MOMDP. The MOMDP comes with three main variations used to adjust the distribution of MTD actions across a certain time window (e.g. a day, a week, or a month) in an optimized manner. To solve MOMDP optimization, the MTD strategy Optimizer uses both single-objective and multi-objective RL (MORL) algorithms [24].

To conclude, the solution provides recommendations on when to perform which MTD actions and also presents what kind of underlying techniques (e.g. container migration method such as pre-copy or post-copy) should be utilized for shorter downtime values.

3. XAI-Driven Intent-Based Security Monitoring and Enforcement

The increasing complexity of B5G/6G networks, combined with the widespread adoption of cloud-native and microservice-based architectures, calls for new paradigms in security management that go beyond traditional rule-based approaches. XAI-Driven Intent-Based Security Monitoring and Enforcement addresses this need by bridging high-level human-defined security intents with enforceable, adaptive policies. This approach leverages intent-based networking to translate user requirements into actionable controls, integrates explainable AI (XAI) to ensure transparency and trust in policy verification and enforcement, and applies adaptive mechanisms to secure dynamic, multi-cloud environments. Together, these capabilities provide a foundation for resilient, trustworthy, and human-understandable security management tailored to the challenges of next-generation networks.

3.1. Intent-Based Networking

Intent-based networking (IBN) [6] is a network architecture that leverages automation and machine learning to help organizations better align their network behaviour with their business objectives. When it comes to security, IBN can play a crucial role in enhancing network security by translating security policies and objectives into automated network configurations and responses. Here's an example of how IBN can be applied to security:

Scenario: Enhanced Security for IoT Devices via Network Slicing

Imagine a smart city deployment where various Internet of Things (IoT) devices, such as surveillance cameras, environmental sensors, and smart traffic lights, are connected through a B5G/6G network. Security and data privacy are top concerns in this scenario. Network slicing can be employed to address these issues. By leveraging network slicing in this smart city scenario, the authorities can achieve both enhanced security and tailored network performance for their diverse IoT deployments. Each network slice operates independently with its own security parameters, reducing the attack surface and mitigating the risk of unauthorized access or data breaches. The following concepts are involved in this scenario:

- Intent Definition: The smart city authorities define the security intent: "Ensure the security and privacy of data from IoT devices while providing low-latency, high-bandwidth connectivity. Isolate different types of IoT traffic for enhanced security."
- Network Slice Creation: Using network slicing capabilities, the B5G/6G network operator creates distinct slices for different types of IoT devices. For example, one slice is dedicated

to surveillance cameras, another for environmental sensors, and a third for traffic management devices.

- Security Policies: For each network slice, specific security policies are defined based on the unique requirements of the IoT devices within that slice. These policies might include traffic isolation, encryption, and access controls.
- Traffic Isolation: Network slicing ensures strict traffic isolation between slices. Data from surveillance cameras is kept separate from environmental sensor data, reducing the risk of data leakage or unauthorized access.
- Encryption: All data transmitted between IoT devices, and the central server is encrypted within each network slice to protect data confidentiality.
- Access Controls: Role-based access controls are implemented within each network slice. Only authorized personnel or systems are granted access to the data generated by IoT devices.
- Real-time Monitoring: Security teams employ real-time monitoring and anomaly detection tools to detect any unusual activity or potential security threats within each network slice.
- Incident Response: In case of a security incident or anomaly, the security teams can respond promptly within the affected network slice while leaving other slices unaffected.
- Data Privacy: Network slicing ensures that data generated by IoT devices is processed and stored within the respective slice, maintaining data privacy and compliance with data protection regulations.
- Customized Security Services: The B5G/6G operator can offer customized security services to the smart city authorities, such as threat detection, vulnerability assessments, and security updates specific to each network slice.

To assess and enforce the specified intents, they first need to be translated to more formal SSLAs that can then be converted to the rules and algorithms that allow analysing the network events in real-time. This monitoring function can be called Security SLA Assessment Function. It is an AI-driven, autonomous component within the 6G core that continuously monitors and enforces SSLAs. It operates by capturing a continuous stream of metrics, such as data availability, geolocation compliance, patch application delays, and isolation integrity between network slices, and correlates this data using advanced analytics to verify that the SSLAs are expected. If any deviation is detected, such as a latency in applying critical patches or a violation of access isolation rules, it will trigger an alarm that can be used to by a security orchestrator to execute the remediation actions, such as reconfiguring resources or revoking access, thus ensuring that security guarantees are dynamically maintained with or without human intervention depending on the type of remediation and the risks involved.

To summarise, the SSLA Assessment Function has as objectives:

- Real-time capture of metrics
- Correlate data
- Verify the defined SSLAs
- Notify and react to any failure

The SSLA metrics that are used include, for instance:

- Data and service availability
- Geo localization of data/services
- Frequency of security analysis
- Number of GTP tunnels per subscriber
- Isolation access from other slices

The security enforcement techniques include, for instance:

- Time to deploy new technique
- Delay in applying patches
- Delay in reconfiguring
- Delay in revoking users/operators
- Delay in replicating services and switching instances

```
<ServiceLevelAgreement xmlns="urn:ngsi-ld:SLA:1.0" id="SLA-Slice-Isolation">
    <name>Isolation Guarantee for Slice</name>
    <description>This SLA guarantees that the Slice is isolated from all other network slices,
                 permitting only whitelisted IP addresses as the source of incoming packets.</description>
    <!-- The specific guarantee term for isolation -->
    <guaranteeTerm>
        <name>NetworkSliceIsolationAndWhitelisting</name>
        <serviceLevelObjective>
            <!-- This KQI defines the the objective -->
            <kpiName>PacketSourceCompliance</kpiName>
            <kpiTarget>
                <!-- The objective is that 100% of packets must be from permitted sources -->
                <value>100</value>
                <unit>percent</unit>
            </kpiTarget>
        </serviceLevelObjective>
        <serviceLevelIndicator>
            <!-- This KPI defines the how to measure it -->
            <kpiName>InvalidSourcePacketCount</kpiName>
            <kpiFormula>
                <!-- The metric to capture: count of packets with source IP NOT in the whitelist -->
                COUNT(ingress_packets WHERE source_ip NOT IN whitelist)
            </kpiFormula>
            <measurementSource>Distributed_Security_Monitoring_Function</measurementSource>
        </serviceLevelIndicator>
        <breachCondition>
            <!-- A breach is triggered if even a single non-whitelisted packet is detected -->
            <formula>InvalidSourcePacketCount > 0</formula>
        </breachCondition>
        <remediationAction>
            <!-- Automated action upon breach -->
            <action>NOTIFY</action>
            <target>SecurityOrchestrator</target>
            <action>LOG</action>
            <action>REJECT_PACKET</action> <!-- Immediate enforcement action -->
        </remediationAction>
    </guaranteeTerm>
</ServiceLevelAgreement>
```

Figure 3: An example of a slice isolation SSLA

To illustrate how an intent can be verified and enforced, we consider in the following the intent “Network slices should be strictly isolated from other slices”. Figure 3 presents an example of a slice isolation SSLA. Note that this is just one way of defining the isolation of slices.

Where KQI=Key Quality Indicator and KPI=Key Performance Indicator

This is translated to the rule presented in Figure 4 that will be used by the Security SLA Assessment Function that monitors the network traffic in a slice and acts as a firewall. Note that this is just one way of implementing the isolation rule.

```
<beginning>
<!-- Isolate Access to a given Slice from other Slices-->
<property value="THEN" property_id="1" type_property="SECURITY" delay_max=0 delay_min=0 if_satisfied=#block_packet_from(ip.src)
     description="Verify source IP and block packet if not in the Slice's whitelist" >
   <event event_id="1" value="COMPUTE"
          description="IP packet detected"
          boolean_expression="(ip.src > 0)"/>
    <event event_id="2" value="COMPUTE"
          description="Detect that source IP is not in the Slice's whitelist"
          boolean_expression="(#whitelist(#get_sliceID(), ip.src) != 1)"/>
</property>
</beginning>
```

Figure 4: A rule sample to monitor the network traffic in a slice and to act as a firewall.

Where we use the following embedded functions:

- "block_packet_from" that blocks the packet
- "get_sliceID" that provide the id of the slice where the probe is running
- "whitelist" that returns 1 if the slice's whitelist does not contain the IP address.

3.2. XAI for Intent Verification and Policy Enforcement

One of the key challenges in intent-based security monitoring is ensuring that high-level security intents are accurately translated into enforceable policies, and that these policies remain effective in dynamic environments such as B5G/6G networks and multi-cloud microservice deployments. Montimage AI Platform (MAIP) [1] addresses this challenge by introducing **explainable AI (XAI)-driven intent verification**, enabling stakeholders to not only enforce policies but also understand how and why enforcement decisions are made. This transparency helps bridge the gap between human-defined objectives and machine-level enforcement, making security management both reliable and trustworthy.

The first step in this process is **policy translation and mapping**. Natural-language security intents—such as “all IoT camera traffic must be encrypted and isolated from public internet traffic”—need to be transformed into concrete, machine-enforceable rules. MAIP uses semantic analysis and rule-matching techniques to verify the consistency between user-defined intents and the actual deployed policies. This guarantees that the enforcement layer faithfully reflects the operator’s original intentions, while minimizing the risks of misconfigurations that could lead to vulnerabilities.

Once policies are deployed, **explainable anomaly detection** ensures that violations and misconfigurations are immediately identified and contextualized. When anomalous traffic patterns or deviations from the intended policy are detected, MAIP provides explanations by pinpointing which intent or policy has been violated, clarifying the reasoning behind detection, and identifying the root cause. For instance, it can reveal that a firewall rule was misconfigured,

[1] Montimage AI Platform (MAIP): https://github.com/Montimage/maip

allowing traffic to bypass encryption, or that a rogue IoT device attempted to communicate outside its designated secure zone. This capability significantly reduces the time needed for incident triage and strengthens operator trust in automated systems.

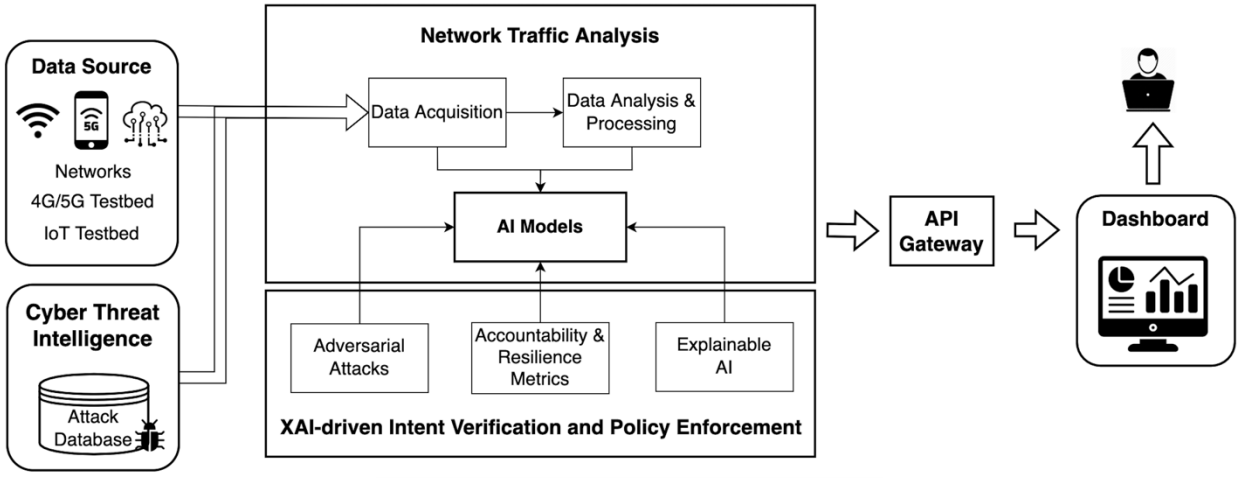

Figure 5: XAI-driven Intent Verification and Policy Enforcement

Figure 5 illustrates how the MAIP integrates multiple data sources—including 4G/5G and IoT testbeds as well as cyber threat intelligence—to enable explainable AI (XAI) for resilient intent-based security enforcement. Network traffic is continuously acquired and analysed, feeding into AI models that support adversarial attack detection, accountability and resilience metrics, and explainable AI outputs. This ensures that security intents are not only enforced but also verified and explained in human-understandable terms.

Incorporating XAI provides transparency and accountability in intent-driven security management. High-level intents are translated into enforceable policies, continuously monitored through anomaly detection, and validated against actual system behaviour. When violations occur, the system offers explanations that clarify which policies were breached and why, reducing the risks of opaque, black-box enforcement. Moreover, the integration of accountability metrics with adaptive orchestration enables closed-loop remediation actions—such as rerouting, workload migration, or policy adjustments—while maintaining service availability. By linking data acquisition, threat intelligence, and explainable AI reasoning, MAIP builds trust in automated enforcement and strengthens resilience across complex multi-cloud and B5G/6G environments.

The XAI component aims to enhance the robustness of AI models built within the Network Traffic Analysis module, making them more resilient against various types of adversarial machine learning attacks. The Adversarial Attacks module focuses on injecting various evasion and poisoning adversarial attacks, such as random label flipping, label flipping attacks and Generative Adversarial Networks (GANs) attacks or integrating existing AI-based attack libraries for the

robustness analysis of AI models. The Explainable AI module aims to produce post-hoc global and local explanations of predictions generated by our model. Specifically, we employ popular model-agnostic post-hoc XAI techniques, such as SHAP (SHapley Additive exPlanations) [1][3] and LIME (Local Interpretable Model-Agnostic Explanations) [4][5], to explain predictions of our models. This module plays a crucial role in improving the transparency of the decision-making process of our models, enhancing interpretability, and, most importantly, ensuring the reliability of the predictions. Furthermore, we incorporate defence mechanisms, such as adversarial training and leveraging XAI techniques, to prevent attacks against the AI models.

To further enhance resilience, MAIP supports **closed-loop orchestration**. When a policy violation occurs, the system does not simply raise an alert but can also automatically trigger adaptive remediation actions. These may include rerouting traffic through secure gateways, tightening access control rules, or even migrating workloads to more trusted infrastructure. Importantly, each action is accompanied by a clear, human-readable explanation, allowing operators to validate and trust the AI's response rather than viewing it as an opaque "black-box" decision.

Finally, MAIP ensures **auditability and transparency** by logging all verification and enforcement actions along with their corresponding explanations. This creates an auditable trail of decisions and responses, supporting regulatory compliance, post-incident investigations, and long-term accountability. Such transparency is particularly critical in highly regulated environments, where operators must not only ensure strong security but also demonstrate compliance to external authorities.

By combining intent verification, explainable anomaly detection, adaptive orchestration, and comprehensive auditability, MAIP ensures that operators can enforce intent-driven security policies with full visibility into the AI's decision-making process. This reduces the risks associated with black-box automation and fosters trust in intelligent security management for complex 6G and IoT ecosystems.

3.3. Adaptive Security in Multi-Cloud and Dynamic Environments

In multi-cloud and dynamic B5G/6G environments, organizations face the challenge of enforcing uniform security policies across highly heterogeneous infrastructures. Applications and services are often deployed across multiple providers, each with its own security mechanisms, controls, and interfaces. This fragmentation complicates the task of maintaining consistent protection for sensitive assets, particularly customer data that may traverse multiple domains. To address this, we introduce adaptive enforcement mechanisms that integrate with moving target defence (MTD), ensuring both resilience against threats and transparency in enforcement.

The security intent in such environments is clear: protect customer data through strong encryption, prevent unauthorized cross-cloud transfers, detect anomalies in real time, and

maintain service availability even under attack or system reconfiguration. We translate these high-level security intents into actionable policies that harmonize encryption standards and access controls across different cloud providers. Doing so reduces the risk of configuration drift and prevents attackers from exploiting inconsistencies between platforms.

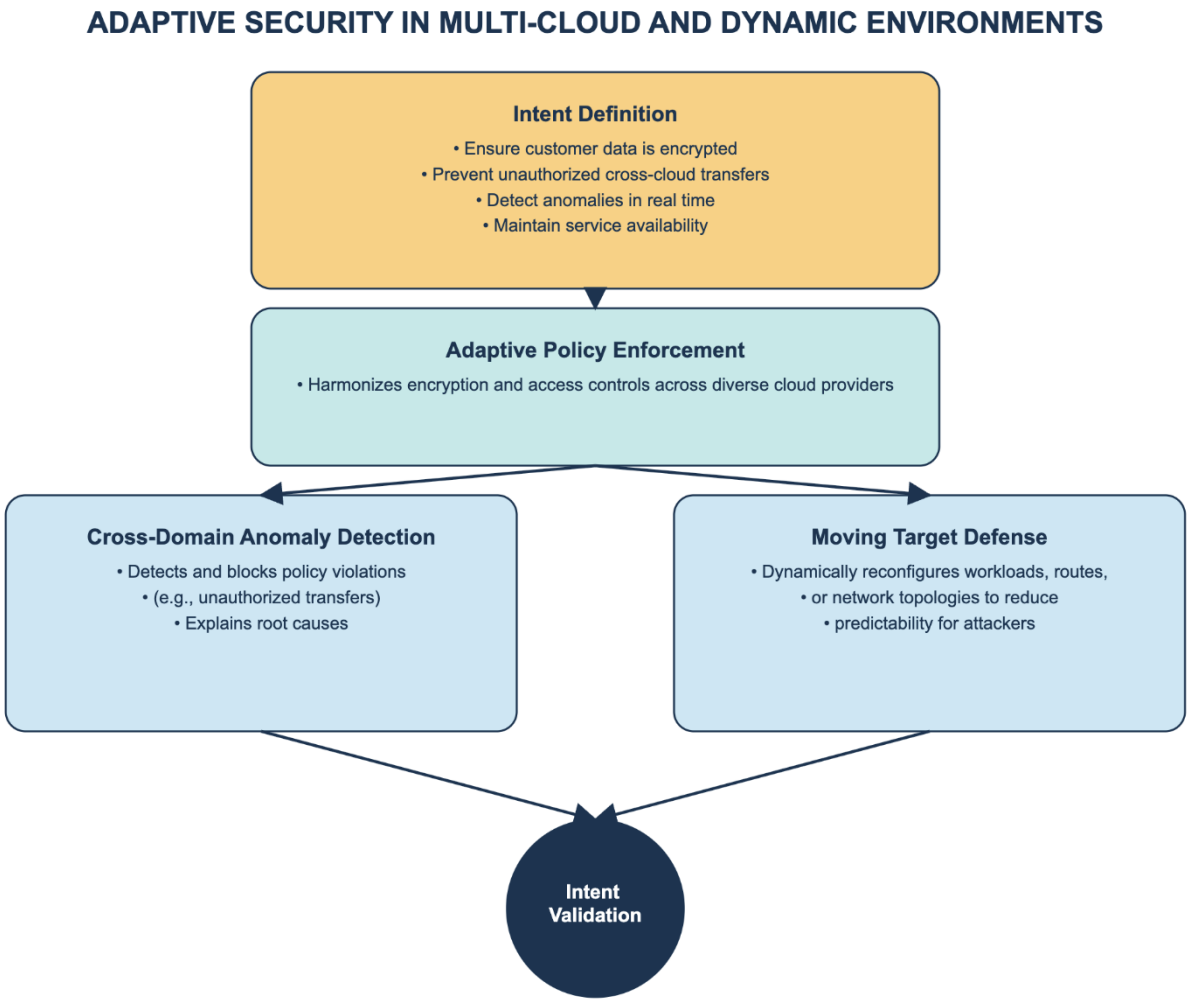

Figure 6: Adaptive Security Framework in Multi-Cloud and Dynamic Environments

Figure 6 illustrates how we enforce high-level security intents across heterogeneous multi-cloud infrastructures by combining adaptive policy enforcement, cross-domain anomaly detection, and moving target defence (MTD). Intent definitions, such as ensuring data confidentiality, preventing unauthorized transfers, detecting anomalies in real time, and maintaining availability, are translated into concrete security actions. Adaptive enforcement harmonizes encryption and access controls across diverse cloud providers, while anomaly detection identifies and explains policy violations. Simultaneously, MTD dynamically reconfigures workloads, routes, or network topologies to minimize attacker predictability. Finally, intent validation ensures that these

adaptive measures remain consistent with the overarching security objectives, preserving both resilience and transparency.

Beyond static policy enforcement, we leverage cross-domain anomaly detection to monitor data flows and application behaviour continuously. This enables the system to identify violations such as unauthorized transfers or suspicious activity that spans multiple clouds. Importantly, we do not simply block such events—it also provides transparent explanations of the root cause, allowing security teams to understand why an action was taken and how it relates to the organization's intent.

A critical feature of our solution is its integration with moving target defence provided by ZHAW. Rather than presenting a static attack surface, the system dynamically reconfigures workloads, routes, or even network topologies, making it more difficult for adversaries to predict and exploit system behaviour. These changes are orchestrated in a way that preserves service continuity and availability, ensuring that legitimate users remain unaffected while attackers face increasing uncertainty.

The adaptive enforcement builds upon recent advances in resource-optimized MTD strategies, such as those developed at ZHAW in the scope of the "Resource-optimised MTD" framework. As defined in the previous deliverable D3.1 [24], the MTD architecture enhances the security of network functions across the edge-to-cloud continuum through proactive and reactive operations such as live migration and re-instantiation of virtualized and containerized network functions. While earlier work focused on the MTD controller and its ability to orchestrate live migrations of containers in Kubernetes clusters, the current focus lies on the MTD Strategy Optimizer. This component leverages AI-based decision-making agents to determine how and when migrations should occur, balancing resilience with performance overhead. Integrating such approaches allows security policies to be enforced adaptively without compromising availability or transparency, ensuring that high-level security intents remain aligned with underlying reconfiguration actions.

By aligning adaptive policy enforcement with advanced MTD strategies, we harmonize encryption and access control policies across cloud providers while dynamically reducing attack predictability through live migration and workload reallocation. The use of AI-powered optimizers ensures that these migrations are not only secure but also efficient, minimizing downtime through techniques such as pre-copy, post-copy, or hybrid live migrations. This results in a security fabric that is both resilient against persistent threats and sensitive to service-level objectives.

4. Adaptive orchestration optimisation for sustainability

The optimisation strategy plugged into the orchestration level combines cybersecurity and sustainability objectives. As introduced in deliverable D3.1 [24], the Secure-by-Design Orchestration framework applies two optimisation strategies that guide orchestration decisions while strengthening network function security. These optimisation strategies are CTI-driven selective sharing of vulnerability data and hygiene scores, and an AI-based workload prediction service. Both of which has the objective of minimizing the energy overhead of 6G services on the underlying edge-cloud infrastructure while maximizing the cybersecurity posture of said services.

4.1. CTI-Driven Selective Sharing

The CTI framework provides hygiene scores from vulnerability data, supporting orchestrator to place CNFs only in domains with an acceptable security posture. From our work on CTI exchange, we developed a sensitivity–necessity mapping mechanism that scores vulnerability metadata to decide what information should be shared between domains. Orchestration decisions are guided by trustworthiness (hygiene scores), avoiding unnecessary exposure of sensitive data. This selective CTI sharing becomes a feedback input to the orchestrator, helping to prevent insecure payload placement and enabling security-per-construction orchestration.

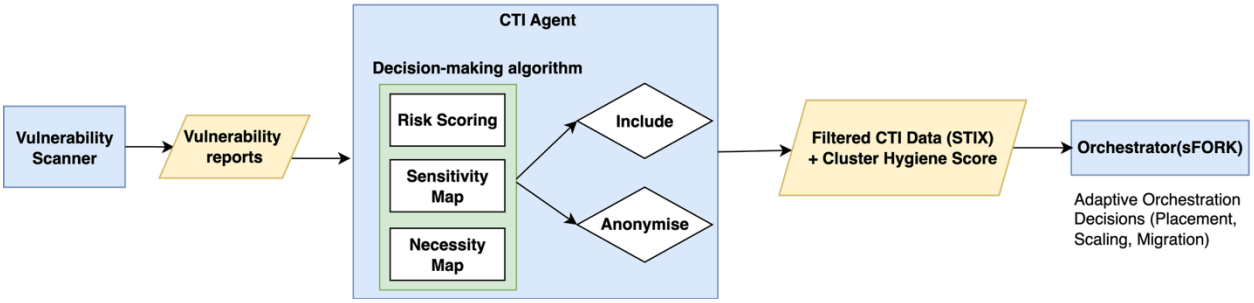

Figure 7: High-level functional view of the CTI selective sharing process

The overall workflow of the CTI selective sharing framework is illustrated in Figure 7. Hygiene scores, derived from vulnerability data, act as a filter for placing functions only in domains with an acceptable security posture. To support this, we developed a sensitivity–necessity mapping mechanism that scores metadata and decides what information to share across domains.

Vulnerability scanners in each cluster send raw reports to the CTI Agent. Each vulnerability is analysed and given a risk score that reflects its threat level. The CTI Agent then applies a decision-making algorithm that combines this risk score with two maps: *sensitivity* and *necessity*. Based on these three values, the algorithm decides whether each field should be shared or anonymised. The filtered data is then serialised in STIX format and exchanged with peer clusters. Cluster hygiene scores are calculated from the shared CTI data together with severity scores. These

scores are sent to the orchestrator, which uses them to guide CNF placement, scaling, and migration.

The algorithm works on metadata fields extracted from vulnerability reports. Each field is evaluated against the *sensitivity map* (defined by the CTI publisher) and the *necessity map* (defined by the CTI subscriber). The sensitivity map shows which fields are too confidential to share, while the necessity map highlights the fields the subscriber needs to see. This dual mapping balances the priorities of both sides, making the exchange more controlled and useful. The risk score adds another dimension. It combines parameters such as the severity and age of a vulnerability to estimate its exploitability. This score acts as a weighted factor in the decision process, giving more importance to high-risk vulnerabilities.

The algorithm evaluates every vulnerability key-value pair with two possible outcomes: “include” or “anonymise.” The resulting CTI data, together with severity values, feeds into the calculation of a cluster-wide hygiene score. This score reflects how trustworthy a cluster is and is directly used by the orchestrator when making deployment decisions. This approach allows orchestration to be guided by trustworthiness, while avoiding disclosure of sensitive details. In T3.4, selective CTI sharing is used as feedback to the orchestrator, preventing insecure placements and enabling security-per-construction orchestration.

4.2. Workload Prediction for Scheduling

The workload prediction service runs as a lightweight AI microservice integrated with the orchestration layer. It interacts with Prometheus [26] monitoring system which collects telemetry from both **services (containers/pods)** and **cluster nodes**. At the service level, Prometheus monitors CPU usage, memory consumption, network traffic (Tx/Rx), and disk I/O over time. At the node level, including both control plane and worker nodes, it tracks aggregate resource utilisation and workload distribution. Using this historical data, the AI model generates short-term forecasts that capture both expected usage trends and unusual anomalies. These predictions allow the orchestrator to anticipate demand rather than simply reacting to it.

Each workload indicator is analysed against two maps: typical fluctuation patterns and anomaly signatures. The prediction model distinguishes between normal growth (e.g., daily traffic peaks) and abnormal behaviours (e.g., DoST-like attacks). Forecast outputs include expected load ranges and anomaly likelihood scores. Based on these, the orchestration engine decides whether to scale, migrate, or throttle workloads in advance.

The prediction service exposes its results through a standard API, enabling seamless integration with scheduling logic. Forecasts are packaged as structured metadata and consumed by the orchestrator to guide placement and migration strategies. When anomaly likelihood is high, the

system prioritises resilience; when predictions show stable trends, the system can favour energy-efficient scheduling.

In T3.4, these AI-driven forecasts act as proactive signals that align payload mobility with both security posture and sustainability goals, ensuring that orchestration decisions are not only reactive but also predictive.

This capability directly addresses the challenge highlighted in D3.1 [24]: without prediction, schedulers risk either over-allocating resources (wasting energy) or under-allocating (causing SLA loss). By providing foresight into both normal and anomalous workloads, the AI service enables the orchestrator to act proactively, improving efficiency, resilience, and sustainability.

5. Secure Data Aggregation

In B5G/6G networks, the number of connected devices and the amount of data they produce will be massive. To develop AI-based service orchestration in such environments, enabling technologies are essential to ensure that large-scale model training can occur without compromising privacy. Secure data aggregation (SDA) is one such enabler: while not a direct orchestration or training technique itself, it supports frameworks like federated learning (FL) by ensuring that model updates from distributed entities can be combined securely and efficiently. Federated learning as a decentralized training framework aims to enhance data privacy by enabling model training directly on local datasets, without requiring raw data to be shared [15]. Participants retain control over their data and perform local training, transmitting only updated model parameters to a central server or several distributed nodes for aggregation into a global model. This framework not only safeguards privacy but also addresses data silo challenges, promoting efficient data utilization between organizations while ensuring compliance with regulations [16] such as the General Data Protection Regulation (GDPR).

However, the transmission of model parameters in FL may lead to the risk of information leakage, as gradients can carry sensitive participant data. This vulnerability, called the gradient leakage attack, arises when the server is either malicious or honest-but-curious, potentially exploiting gradients to infer private information. To address this issue and improve the practical viability of FL, the Secure Aggregation protocol needs to be utilized [17]. This protocol safeguards the aggregation process by employing techniques such as encryption, randomization, and scrambling, ensuring that sensitive information remains confidential during both transmission and aggregation of model updates. Additionally, intent-based service security can define requirements such as “aggregate updates in a way that hides every device’s data.” The system then selects Multi Party Computation (MPC) protocols that satisfy this intent, such as additive secret sharing or homomorphic encryption.

5.1. Secure Data Aggregation Protocol

This protocol supports two interchangeable approaches: centralized data aggregation and MPC-based secure aggregation. Both are designed to provide strong privacy guarantees, robustness against malicious participants, and reliable aggregation even when clients drop out or experience delays.

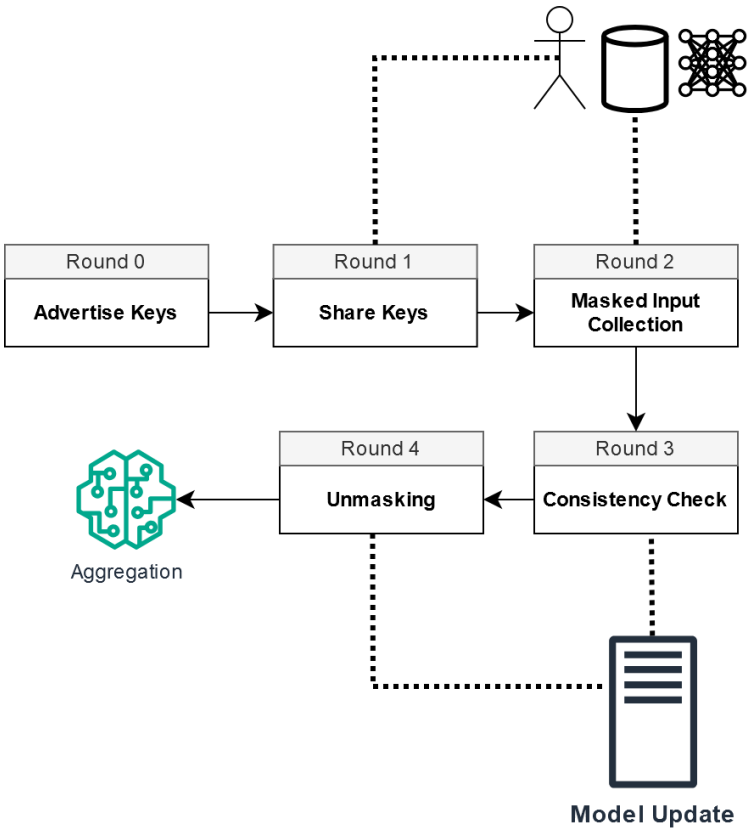

Figure 8: Secure Data Aggregation Process

In the centralized data aggregation protocol, as shown in Figure 8 a central server computes the sum of client inputs while keeping each individual input confidential. This version of the protocol assumes that either the server or clients could act maliciously and actively prevent data leakage or manipulation. Each client first generates a private and public key in round 0 and establishes shared secrets with every other client in round 1. These shared secrets are used to create masking values in round 2 that hide the true input. The client then sends the masked input to the server. The server collects all masked inputs in round 3 and combines them to compute the correct aggregated sum in round 4. If some clients drop out, the server can reconstruct the missing masking values using a cryptographic secret sharing mechanism. The protocol also ensures that delayed client submissions cannot be reconstructed prematurely, preserving privacy. This design guarantees that the aggregation remains both robust and private, even under adversarial conditions.

MPC-based aggregation depicted in Figure 9 removes reliance on a single trusted server by distributing computation across multiple parties. This comes at the cost of slightly more computation overhead. Clients first generate keys and establish shared secrets, then mask their inputs using the same cryptographic primitives as in the centralized protocol. Each client splits its masked input into encrypted shares and distributes them among several computation parties.

The computation parties jointly perform the aggregation without accessing the full input from any single client. After the aggregation is completed, the result is reconstructed and shared with all clients. This approach ensures that no individual party, including a central server, can access complete client data. Similar to centralized aggregation, missing inputs due to client dropouts can be reconstructed securely, and delayed inputs remain protected. The distributed nature of computation increases resilience against malicious participants, as collusion between multiple parties would be required to compromise the aggregation.

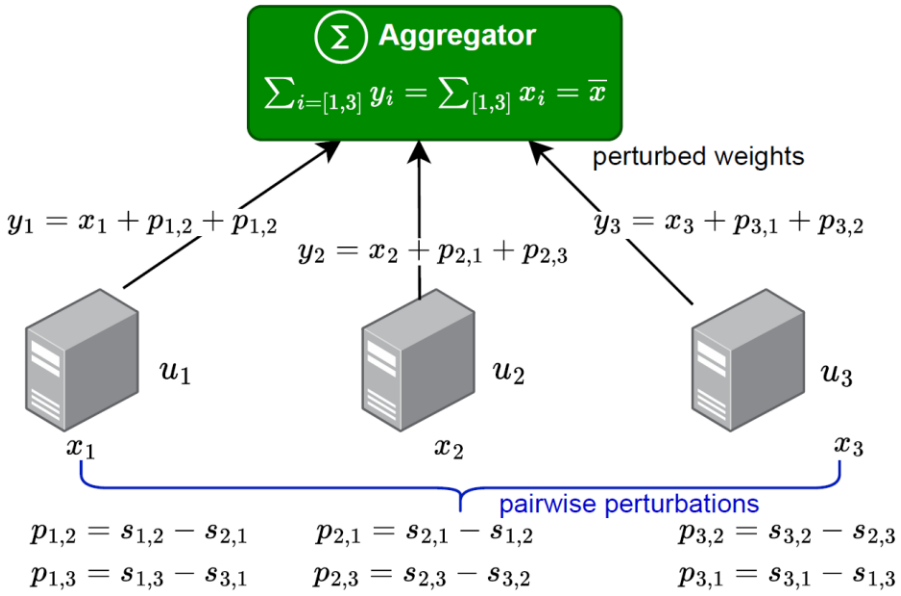

Figure 9: MPC-based aggregation

Centralized aggregation is efficient and well-suited when a trusted server is available, providing privacy and robustness through masking and secret sharing. MPC-based aggregation [18] offers stronger privacy and security, as it eliminates single points of trust and distributes computation, making it ideal for untrusted or adversarial environments. Although MPC introduces additional computational and communication overhead, it ensures that even in high-risk scenarios, client data remains confidential. By supporting both methods, a federated learning system can flexibly choose the aggregation approach that best meets its current operational and security requirements. This dual capability allows the system to balance efficiency, privacy, and robustness, ensuring secure and reliable model training across diverse environments.

5.2. Attack Resilience

The secure aggregation protocol demonstrates strong resilience against several types of adversarial attacks commonly encountered in federated learning, particularly in dynamic B5G/6G environments. In this section, some major security attack [19] against the described protocol have been slightly highlighted. Impersonation attacks, where malicious clients attempt to masquerade as legitimate participants, are mitigated by the protocol's robust key exchange and authentication mechanisms. Each client generates unique private and public keys and establishes shared secrets with other participants, making it extremely difficult for an attacker to inject fake updates without being detected. Combined with the masking process, this ensures that even if an adversary joins the network, they cannot meaningfully influence the aggregated result or recover other clients' private data.

Label-flipping attacks where malicious clients intentionally mislabel their local data to degrade the global model, represents a significant threat against model accuracy. Experiments have shown that while label-flipping attacks can drastically reduce accuracy in non-secure configurations, integrating secure aggregation with MPC mitigates this impact [20]. By distributing the aggregation process across multiple computation nodes, the protocol reduces the influence of any single malicious participant. Even when a substantial fraction of clients behaves maliciously, the MPC framework preserves a higher level of model accuracy compared to non-MPC setups, particularly over longer training iterations. Although some degradation still occurs, the secure aggregation process provides a buffer against extreme manipulation of the model.

Other sophisticated adversarial behaviours, such as min-max attacks or general poisoning attempts, can also be addressed through a combination of masking, secret sharing, and distributed computation. While min-max attacks tend to have a more gradual impact on model performance, the protocol's adaptive security capabilities further enhance resilience. By continuously monitoring updates for anomalies, dynamically adjusting privacy mechanisms, and re-weighting or excluding suspicious contributions, the system can respond in real time to emerging threats. This adaptive approach ensures that defences evolve alongside the environment, making it much harder for attackers to succeed in dynamic, high-mobility networks typical of B5G/6G scenarios.

The integration of adaptive security techniques is particularly valuable in highly dynamic federated learning settings. By combining policy-driven intent with automated anomaly detection, cryptographic protections, and trust management, the system forms a closed feedback loop: user-defined security goals guide adaptive measures, which continuously enforce and adjust protections in response to observed threats. This loop not only enhances resilience against known attacks like impersonation and label-flipping but also strengthens the system's

ability to defend against emerging or unforeseen attack vectors. The combined effect of MPC-based aggregation, cryptographic masking, and adaptive security ensures that the federated learning framework remains both privacy-preserving and robust, even under significant adversarial pressure.

5.3. Aggregation Integration

The secure aggregation protocols can play a key role in enhancing other NATWORK components such as MTD strategies [21] and performing behavioural analysis. Since both mentioned areas require the collection and processing of sensitive data from multiple network components without exposing individual data sources, privacy-preserving aggregation is an essential requirement.

In MTD, network configurations and system parameters are continuously shifted to make attacks harder to execute and predict. Monitoring and analysing these dynamic changes across multiple nodes require aggregating information such as traffic patterns, node status, or potential anomalies. By using centralized secure aggregation, network controllers can collect masked telemetry data from multiple devices to calculate global metrics, detect potential attack surfaces, and guide configuration changes, all without exposing the details of any single device. When the trust in the central controller is limited or higher security is needed, MPC-based aggregation can distribute the computation among multiple monitoring agents. This ensures that no single entity can access complete network data, resulting in a significantly harder surface for attackers to exploit the defence system or infer sensitive behaviours.

For data plane behavioural analysis where the goal is to detect anomalous traffic patterns, each switch or device generates local statistics, such as packet counts, flow records, or latency measurements. Secure aggregation allows these measurements to be combined into global models without exposing the raw data from any single device. Using the protocols, either the central controller (centralized aggregation) or multiple computation parties (MPC-based aggregation) can compute network-wide metrics, identify suspicious behaviour, and trigger automated responses. This approach not only preserves privacy but also enhances robustness against compromised devices that might attempt to inject false data into the analysis.

By integrating the secure aggregation protocols into MTD and behavioural monitoring, networks gain several benefits: they can adapt dynamically to threats, maintain privacy of individual nodes, and ensure that the analysis remains resilient even in adversarial conditions. This makes federated, privacy-preserving aggregation a powerful tool for next generation networks, where both adaptability and confidentiality are critical.

6. Microservice profiling and anomaly detection - Microservice behavioural analysis

The NATWORK B5G architecture follows a microservice-based approach. Microservices architecture is a fundamental enabler of flexible and scalable 6G network services. Unlike monolithic applications, in microservice-based applications, network functions are decomposed into smaller, independent components that operate autonomously, allowing for scalable deployment, real-time adaptability, and efficient resource management, making them well-suited for dynamic network environments. In the following section, a module that monitors the performance of microservices in a continual manner to ensure the efficient operation of the system is presented. The proposed module assesses the impact of microservice on compute and network resources to quickly detect anomalies in resource consumption and solve them before the systems' performance is affected. The outputs of the mechanism will be used in microservices orchestration platforms to enable resource-efficient and reliable placement of microservices in 6G environments.

6.1. Technical Description

The Microservice behavioural analysis module performs continuous microservice performance monitoring to ensure efficient operation of the system. In NATWORK, this involves leveraging runtime metric collectors and packet sniffers to continuously track key performance indicators, such as CPU and memory usage, ingress/egress traffic, etc. The module analyses real-time monitoring data, to determine whether microservices meet predefined performance requirements and if abnormal traffic flows occur in the network. If deviations are detected by an AI-based Detection Mechanism, automated scaling decisions and elasticity actions will be triggered to maintain optimal resource utilization and prevent service degradation. Figure 10 shows the position of the microservice behavioural analysis module and its interconnection to other modules. This module is comprised of microservice profiling techniques and AI-driven anomaly detection mechanisms for enhanced microservice profiling and threat detection, shown in Figure 10 as *Microservice Behavioral Analysis*. It interacts with the monitoring engine, to collect real-time data on microservices resource usage and traffic metrics, the microservice orchestrator to trigger scaling decisions dynamically based on detected anomalies and the SDN controller to enforce mitigation actions. The monitoring engine has been presented in deliverable D4.1 [8].

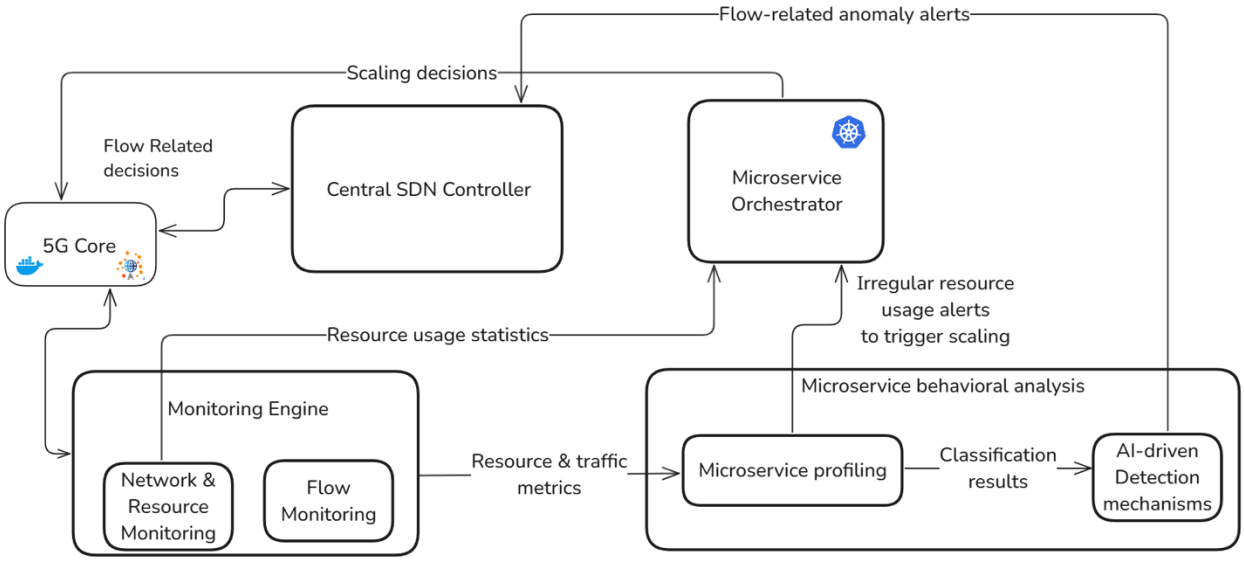

Figure 10: Position of the microservice behavioural analysis module and interconnection to other modules

6.1.1. Algorithms utilized for proposed solution

While microservices offer significant advantages, their distributed nature makes them inherently more vulnerable to threats such as denial-of-service (DoS) attacks, privilege escalation, and unauthorized access. Given these risks, security is another critical aspect, making behavioural analysis essential in microservice-based architectures to detect anomalies and provide protection against potential breaches.

To effectively analyse microservice behaviour, it is necessary to monitor their performance both on a temporal and a periodic basis, aiming to identify any deviations from normal operation [9]. On the one hand, tracking how network traffic patterns change over time allows for capturing anomalies that may evolve gradually or threats that are identifiable only by analysing a certain period. On the other hand, sudden changes in microservice behaviour, such as unexpected spikes in ingress traffic or unusual increases in resource consumption, may indicate malicious activity and should be contained immediately.

To address these challenges, microservice profiling techniques and AI-based anomaly detection mechanisms are employed to analyse system behaviour in real-time and identify anomalous behaviours. The microservice profiling technique can identify anomalous resource usage for the microservices deployed at any moment. For microservices with anomalies detected or some critical microservices, the AI-powered detection mechanisms start to continuously analyse telemetry monitoring data to establish behavioural models of a) normal microservice resource consumption and b) traffic and flow related data offered by the Monitoring engine that was presented in [8].

A two-stage anomaly detection approach is followed. First, a lightweight 1D- Convolutional Neural Network (CNN) discerns anomalous resource usage. By profiling CPU, memory, and network usage under typical conditions, the system can classify microservice behaviour as normal or irregular, identify deviations that indicate potential ongoing attacks or unexpected system behaviour that aims at exhausting the network resources. Then, in the case of anomalous data occurrence a more robust 1D-CNN discerns the type of the anomaly, with three different types of output:

- The Anomaly type when the pattern of the resource use corresponds to a known anomaly type,
- False Positive Occurrence when the pattern of resource consumption corresponds to normal behaviours i.e., in the case the first CNN produced a false positive.
- Uknown, when the pattern of metrics examined does not correspond to any of the previous cases, which warrants more inspection by the system operator.

A regression-based forecasting mechanism was designed to model CPU and memory consumption for containerized microservices, enabling proactive resource management in cloud-native environments [10]. The proposed approach leverages a lightweight Long Short-Term Memory (LSTM) deep neural network, chosen for its ability to capture temporal dependencies and nonlinear patterns in time-series resource metrics. By continuously monitoring historical CPU and memory usage, the LSTM model learns workload dynamics and generates regression predictions that estimate near-future consumption levels. The lightweight design ensures minimal overhead, making it suitable for deployment within resource-constrained microservice ecosystems. Such predictive modeling supports autoscaling, anomaly detection, and performance optimization, ultimately improving service reliability while reducing infrastructure costs. Figure 11 presents the envisioned high level architecture of the proposed solution.

Appendix I contains detailed information on the architecture of the Neural networks utilized.

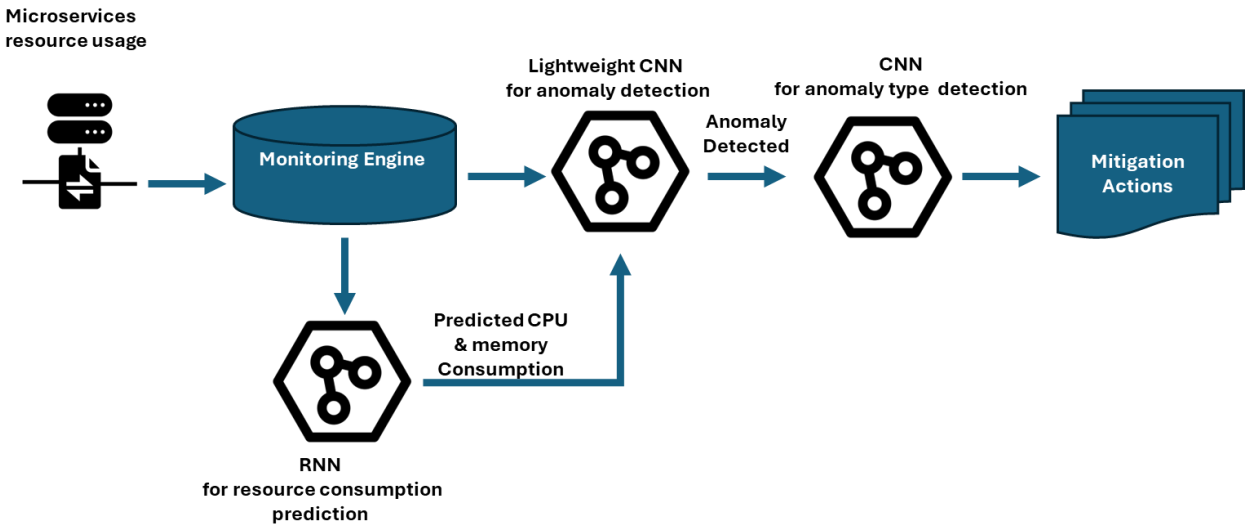

Figure 11 High level architecture of the proposed solution

6.2. Microservice Profiling Evaluation Results

Concerning microservice resource use, a Microservice Profiling tool was developed that monitors twelve different metrics, which cover both the infrastructure level, i.e., the consumption of system resources, e.g., CPU and memory utilization, and application-related metrics related to latency, throughput, and errors concerning the applications. Then based on this data it detects anomalous resource consumption patterns. These metrics are listed in Table 1. This approach is in line with the practice proposed in [7].

Table 1 Evaluation results

Category	Metric	Description
Infrastructure-level (System Resources)	Disk Read Throughput	Data read from disk per unit time; indicates I/O demand.
	Disk Write Throughput	Data written to disk per unit time; reflects storage usage.
	Memory Usage (RSS)	Actual physical memory (RAM) used by the container's processes.
	Memory Usage (VSize)	Total virtual memory reserved (address space); may highlight leaks.
	CPU Utilization	Percentage of CPU time consumed; shows processing demand.
	Network RX (bytes/ns)	Rate of incoming network traffic (data received).
	Network TX (bytes/ns)	Rate of outgoing network traffic (data sent).

Category	Metric	Description
Application-level (Performance & Reliability)	Latency p50	Median response time; typical user experience.
	Latency p90	Response time below which 90% of requests fall; shows tail performance.
	Latency p99	Response time below which 99% of requests fall; highlights worst-case scenarios.
	Request Throughput	Number of successful requests processed per unit time.
	Error Rate	Rate of failed requests; critical for reliability monitoring.

To present a proof of concept for the underlying algorithms of the modules, related to microservice profiling, we utilized the open dataset presented in [7]. This dataset contains five different types of microservice related anomalies: High CPU utilization, High Memory consumption, Sudden spike in user traffic, Gradual step increase in user load, High Network latency.

Table 2 presents the results of the lightweight CNN that performs binary classification of the microservices resource usage (Normal/Anomalous), along with the performance of other AI/ML algorithms commonly used for the same task, a Multi-Layer Perceptron DNN, a Random Forest and an SVM. The proposed method outperforms all other methods in terms of metrics commonly utilized to evaluate classification tasks.

Table 2 Results of the lightweight CNN that performs binary classification of the microservices resource usage (Normal/Anomalous)

Algorithm	Accuracy	Precision	Recall	F1-Score
Lightweight 1-D CNN	**0.9712**	**0.94655**	**0.8567**	**0.8967**
MLP	0.9631	0.9171	0.8541	0.8844
Random Forest	0.9658	0.9618	0.8157	0.8827
SVM	0.9601	0.9529	0.8013	0.8789

Table 3 presents the results of the 1-D CNN that performs multiclass classification of the microservices resource usage (5 anomalies, normal, unknown), along with the performance of other AI/ML algorithms commonly used for the same task, a Multi-Layer Perceptron DNN, a Random Forest and an SVM. In this case the proposed algorithm outperforms all other

approaches in terms of Accuracy, Recall and F1 Score, however it is slightly outperformed by the Random Forest implementation for the Precision metric.

Table 3 Results of the CNN that performs multiclass classification of the microservices resource usage (7 classes: Normal/Five Anomalies/Unknown)

Algorithm	Accuracy	Precision	Recall	F1-Score
1-D CNN	**0.9077**	0.8956	**0.9048**	**0.8964**
MLP	0.8718	0.8605	0.8718	0.8584
Random Forest	0.8956	**0.8961**	0.8956	0.9034
SVM	0.8410	0.8062	0.8410	0.8192

Once a specific anomaly is detected, an automated mitigation action can be triggered to counter it. Two example mitigation actions per anomaly are presented in Table 4.

Table 4 Common actions used to mitigate the microservice anomalies examined.

Anomaly	Example Mitigation Actions
High CPU Utilization	• **Horizontal Pod Autoscaler** Automatically increase/decrease pod replicas based on CPU usage. • **Vertical Pod Autoscaler :** Automatically adjust CPU requests/limits to optimize pod performance.
High Memory Consumption	• **Pod Memory Limits + Auto-restart**: Kubernetes evicts/restarts pods exceeding memory limits. • **Vertical Pod Autoscaler** : Adjust memory requests automatically based on usage trends.
Sudden Spike in User Traffic	• **Horizontal Pod Autoscaler:** Scale pods dynamically based on traffic metrics. • **Cluster Autoscaler**: Automatically add nodes when pods can't be scheduled due to resource constraints.
Gradual Step Increase in User Load	• **HPA with Custom Metrics**: Scale based on app-specific metrics (e.g., requests per second). • **Predictive Autoscaling**: Automatically adjust resources based on historical traffic patterns.
High Network Latency	• **Service Mesh Retry & Load Balancing**: Automatically reroute or retry requests on slow connections. • **Pod Affinity / Topology-Aware Scheduling**: Automatically schedule pods close to dependencies to reduce latency.

6.3. AI driven detection mechanism

This section presents evaluation results for an AI based regression mechanism, based on a LSTM type RNN which was developed to forecast the consumption of memory (RSS) and CPU by the

user. The experiments presented utilized an open dataset [11], which measured multiple resource related features for two web app microservices, explained in Table 5. The measurements were taken every 1 second, both in normal conditions and under various attacks (two DDoS attack variants, A brute Force attack and SQL injection).

Table 5 Features contained in [13] and utilized for forecasting resource prediction

Feature	Description
container_cpu_cfs_periods_rate	Number of CPU scheduling periods (under Linux CFS) per second allocated to the container.
container_cpu_cfs_throttled_periods_rate	Number of CPU periods per second in which the container's CPU usage was throttled (limited).
container_cpu_cfs_throttled_seconds_rate	Total time per second (in seconds) that the container was throttled by CPU limits.
container_cpu_system_seconds_rate	Rate of CPU time consumed by system (kernel) processes on behalf of the container.
container_cpu_usage_seconds_rate (target)	Total CPU usage rate (in seconds per second) consumed by the container.
container_cpu_user_seconds_rate	Rate of CPU time spent by user-level processes inside the container.
container_file_descriptors	Current number of open file descriptors (files, sockets, etc.) held by processes in the container.
container_memory_failures_rate	Rate of memory allocation failures (e.g., due to hitting limits) inside the container.
container_memory_rss (target)	Resident Set Size: amount of non-swappable physical memory (RAM) used by the container.
container_memory_usage_bytes	Current total memory usage of the container (includes cache + buffers).
container_memory_working_set_bytes	Actively used memory that cannot be reclaimed easily (excludes cache).
container_network_receive_bytes_rate	Rate of incoming network traffic in bytes per second received by the container.
container_network_receive_packets_rate	Rate of incoming network packets per second received by the container.
container_network_transmit_bytes_rate	Rate of outgoing network traffic in bytes per second sent by the container.
container_network_transmit_packets_rate	Rate of outgoing network packets per second sent by the container.
container_sockets	Number of active network sockets opened by the container.

Feature	Description
container_threads	Number of active threads running inside the container.

We trained two different LSTMs to predict Total CPU usage rate per second consumed by the container and Resident Set Size memory i.e. the amount of non-swappable physical memory (RAM) used by the container. The networks were trained for 30 epochs, and we experimented with the steps in the future the algorithms predict. Here, we present results for the last period where the prediction of the algorithm has an R^2>0.9. R^2, called coefficient of determination, measures the proportion of variance in a dependent variable that is explained by the independent variables in a regression model. It is expressed as a value between 0 and 1, where a higher R^2 value indicates that the model provides a better fit to the data and explains more of the variability.

For Container CPU usage Rate, the forecast step was 10 seconds into the feature, and the related metrics were Mean absolute Error (MAE: 0.0028), Root Mean Squared Error (RMSE: 0.0117) and R^2: 0.9769. A plot for the forecast values is shown in Figure 12.

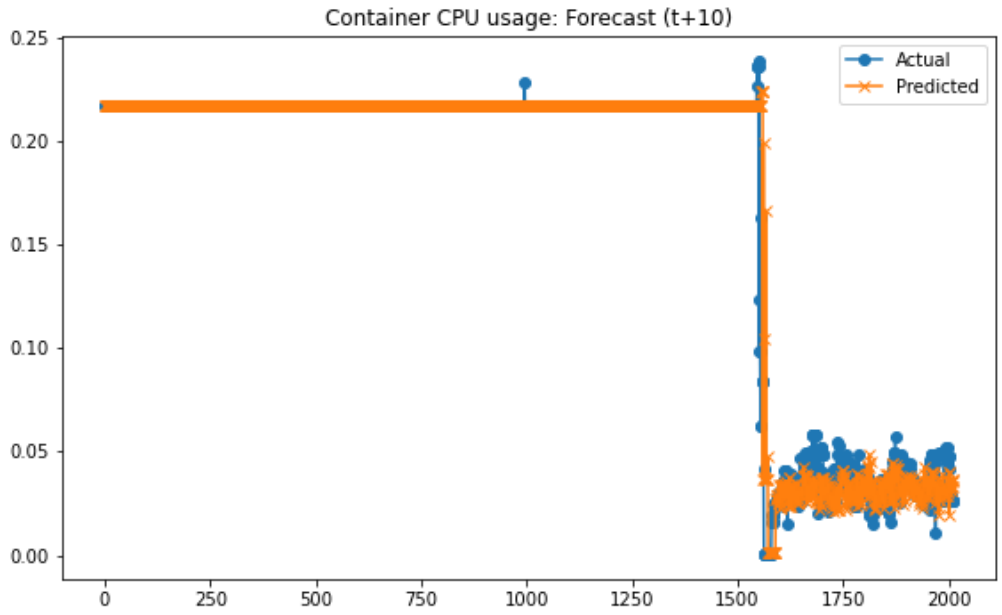

Figure 12: CPU consumption forecast example for one of the containers contained in the dataset (step t=10)

For RSS, the forecast step was 3 seconds into the feature and the related metrics were Mean absolute Error (MAE: 23815.3061), Root Mean Squared Error (RMSE: 16443.2474) and R^2: 0.9056. A plot for the forecast values is shown in Figure 13.

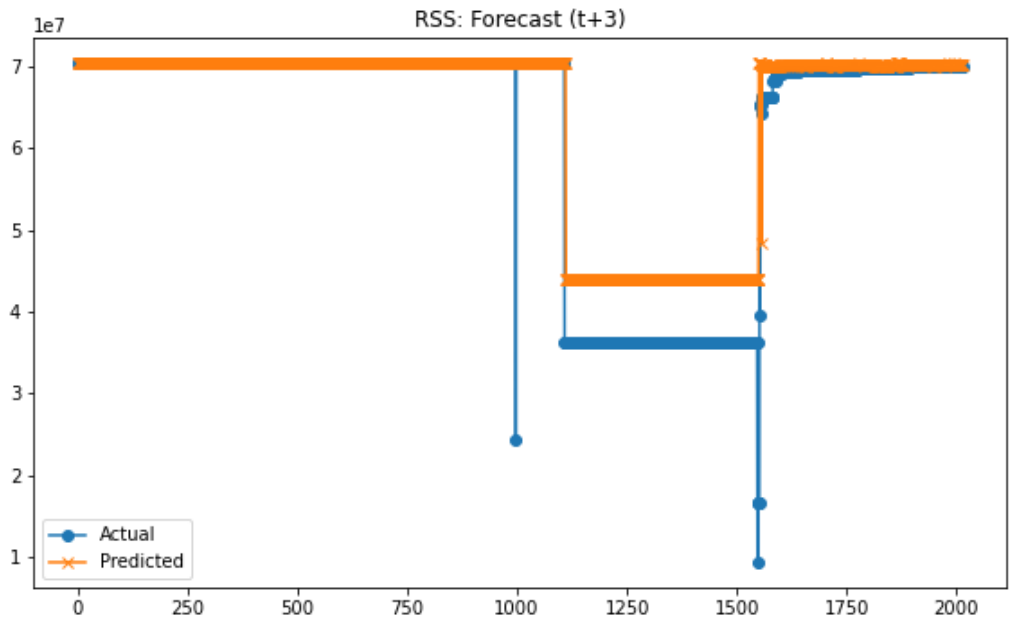

Figure 13: Memory Consumption (RSS) forecast example for one of the containers contained in the dataset (step t=3)

The output of these neural networks will be integrated to that of the CNN to further enhance its' performance and add temporal capabilities to the tool. It will also be integrated with the Monitoring Engine to try to correlate Flow Related data with resource anomalies and produce microservice flow-related anomaly alerts that will be fed to the SDN controller.

The following steps will be carried out during the upcoming months:

- Advance the capabilities of Microservice Profiling by expanding it to cover interactions between microservices and their impact on computational and network resources via the utilization of prediction algorithms.
- Develop the interfaces for automated mitigation action enforcement.
- Finalize the integration of Microservice profiling with the AI-driven detection mechanisms, including AI-enabled IDS developed by CERTH in WP4.
- Perform Scalability and Deployment Testing, i.e., a) Test the framework's scalability with increasing numbers of microservices and varying workloads in simulated 6G environments, and b) Transition from simulated tests to pilot deployments in real-world setups to validate framework reliability and efficiency.
- Continuously integrate with relevant tasks and UC4.4, i.e., the AI-enabled IDS (T4.3) and the Optimized resource allocation for microservices (T4.2).

7. Conclusion

To meet the security and NetZero challenges of future 6G networks, a key technical focus of the NATWORK project is to enable dynamic and footprint-efficient orchestration and management of secure, distributed, and resilient services across the 6G continuum. This deliverable outlines recent outcomes within the NATWORK T3.3, including the integration of AI-based approaches for intelligent MTD and intent-aware security management, the facilitation of adaptive orchestration with sustainability optimization, and the development of agile defense mechanisms for secure microservices towards this overarching goal. Collectively, these efforts advance NATWORK's objective to deliver resource-efficient, adaptive, and intelligent security capabilities for the 6G continuum. In the following period, we will build on these assets and develop them further with evaluation and experimental results, which will be reported in D3.4 to be published in 2026 and future deliverables about integration, testing and evaluation.

References

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

[2] Lundberg, S. M., & Lee, S.-I. (2017). *A Unified Approach to Interpreting Model Predictions.* Advances in Neural Information Processing Systems (NeurIPS), 30. https://arxiv.org/abs/1705.07874

[3] Lundberg, S. M. et al. (2020). *From local explanations to global understanding with explainable AI for trees.* Nature Machine Intelligence, 2, 56–67. https://www.nature.com/articles/s42256-019-0138-9

[4] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *"Why Should I Trust You?" Explaining the Predictions of Any Classifier.* Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). https://arxiv.org/abs/1602.04938

[5] Official LIME Python library: https://github.com/marcotcr/lime

[6] Clemm, A., & Medved, J. (2017). *Intent-Based Networking—Concepts and Definitions.* Cisco Systems. https://tools.ietf.org/id/draft-irtf-nmrg-ibn-concepts-03.html

[7] D. Fernando, M. A. Rodriguez and R. Buyya, "iAnomaly: A Toolkit for Generating Performance Anomaly Datasets in Edge-Cloud Integrated Computing Environments," *2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC)*, Sharjah, United Arab Emirates, 2024, pp. 236-245, doi: 10.1109/UCC63386.2024.00041.

[8] NATWORK Consortium – Deliverable D4.1 "Payload security per runtime, intelligent runtime selection and attestation.r1", March 2025

[9] N. Bugshan, I. Khalil, A. P. Kalapaaking and M. Atiquzzaman, "Intrusion Detection-Based Ensemble Learning and Microservices for Zero Touch Networks," in *IEEE Communications Magazine*, vol. 61, no. 6, pp. 86-92, June 2023, doi: 10.1109/MCOM.001.2200535.

[10] V. Passas, N. Makris, Y. Wang, A. Apostolaras, A. Mpatziakas, A. Drosou, T. Korakis, and D. Tzovaras, "Artificial Intelligence for network function autoscaling in a cloud-native 5G network," *Computers and Electrical Engineering,* vol. 103, p. 108327, 2022, doi: 10.1016/j.compeleceng.2022.108327

[11] R. Morsli, N. Kara, H. Ould-Slimane and L. Lahlou, "Multidimensional Intrusion Detection System for Containerized Environments," 2025 IEEE 11th International Conference on Network Softwarization (NetSoft), Budapest, Hungary, 2025, pp. 546-554, doi: 10.1109/NetSoft64993.2025.11080585.

[12] Soussi, Wissem, Gürkan Gür, and Burkhard Stiller. "Democratizing container live migration for enhanced future networks - A survey." ACM Computing Surveys 57.4 (2024): 1-37.

[13] Soussi, Wissem, Gürkan Gür, and Burkhard Stiller. "Moving Target Defense (MTD) for 6G Edge-to-Cloud Continuum: A Cognitive Perspective." IEEE Network (2024).

[14] Arulkumaran, Kai, et al. "Deep reinforcement learning: A brief survey." IEEE signal processing magazine 34.6 (2017): 26-38.

[15] Yan, Y., Alshawki, M. B., Zoltay, M., Gal, M., Hollos, R., Jin, Y., ... & Tenyi, A. (2024). Fedlabx: a practical and privacy-preserving framework for federated learning. Complex & Intelligent Systems, 10(1), 677-690. https://doi.org/10.1007/s40747-023-01184-3

[16] Truong, N., Sun, K., Wang, S., Guitton, F., & Guo, Y. (2021). Privacy preservation in federated learning: An insightful survey from the GDPR perspective. Computers & Security, 110, 102402. https://doi.org/10.1016/j.cose.2021.102402

[17] Liu, Z., Guo, J., Yang, W., Fan, J., Lam, K. Y., & Zhao, J. (2022). Privacy-preserving aggregation in federated learning: A survey. IEEE Transactions on Big Data. https://doi.org/10.1109/TBDATA.2022.3190835

[18] Gehlhar, T., Marx, F., Schneider, T., Suresh, A., Wehrle, T., & Yalame, H. (2023, May). SafeFL: MPC-friendly framework for private and robust federated learning. In 2023 IEEE Security and Privacy Workshops (SPW) (pp. 69-76). IEEE. https://doi.org/10.1109/SPW59333.2023.00012

[19] Zhang, J., Zhu, H., Wang, F., Zhao, J., Xu, Q., & Li, H. (2022). Security and privacy threats to federated learning: Issues, methods, and challenges. Security and Communication Networks, 2022(1), 2886795. https://doi.org/10.1155/2022/2886795

[20] Abdullah, Y.,Alshawki, M.B., Ligeti, P., Soussi W., & Stiller B., (2025). Byzantine-ResilientFederated Learning: Evaluating MPC Approaches IEEE ICDCSW

[21] Lei, C., Zhang, H. Q., Tan, J. L., Zhang, Y. C., & Liu, X. H. (2018). Moving target defense techniques: A survey. Security and Communication Networks, 2018(1), 3759626. https://doi.org/10.1155/2018/3759626

[22] National Institute of Standards and Technology. National Vulnerability Database. U.S. Department of Commerce, 2024, https://nvd.nsit.org. Accessed 16 Sept. 2025.

[23] Forum of Incident Response and Security Teams (FIRST). Common Vulnerability Scoring System (CVSS). 2024, https://www.first.org/cvss/. Accessed 16 Sept. 2025.

[24] Hayes, Conor F., et al. "A practical guide to multi-objective reinforcement learning and planning." arXiv preprint arXiv:2103.09568 (2021)

[25] NATWORK Consortium – Deliverable D3.1 “Secure-by-design orchestration and management & Data plane computation offloading.r1”, June 2025

[26] J. Turnbull, *Monitoring with Prometheus*. Turnbull Press, 2018.

Annex A Details on the architectures of the DNN used for Microservice profiling

The following subsection presents details on the architectures of the Neural networks utilized for the microservice profiling tool discussed in Section 6.1.1.

Architecture for Lightweight 2-class 1D-CNN (binary classification)

Input (T x C)

|

Conv1D (16 filters, kernel=5, ReLU)

|

BatchNorm

|

SeparableConv1D (32 filters, kernel=3, ReLU)

|

MaxPooling1D (pool=2)

|

Conv1D (64 filters, kernel=3, ReLU)

|

GlobalAveragePooling1D

|

Dropout (0.25)

|

Dense (32, ReLU)

|

Dense (2, Softmax)

1D-CNN multiclass classification

Input (T x C)

|

Conv1D (32 filters, kernel=5, ReLU)

|

BatchNorm

|

Conv1D (64 filters, kernel=3, ReLU)

|

MaxPooling1D (pool=2)

|

Conv1D (128 filters, kernel=3, ReLU)

|

GlobalAveragePooling1D

|

Dropout (0.3)

|

Dense (64, ReLU)

|

Dense (7, Softmax)

Lightweight LSTM Regression Model (forecasting CPU/Memory)

Input (T x C)

|

Conv1D (32 filters, kernel=3, causal, ReLU)

|

BatchNorm

|

Dropout (0.1)

|

LSTM (64 units, return_sequences=True)

|

Dropout (0.2)

|

LSTM (32 units, return_sequences=False)

|

Dense (32, ReLU)

|

Dense (16, ReLU)

|

Dense (h*C, Linear Output)