



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

D3.4 Intent-based service security and adaptive security techniques.r2

Lead beneficiary	ZHAW	Lead author	Wisseem Soussi
Reviewers	Padial Leonardo (HES-SO) and Joaquín Escudero Garzás (GRAD)		
Type	R	Dissemination	PU
Document version	V1.0	Due date	30/06/2026



Co-funded by
the European Union

6G SNS

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, Wissem Soussi (ZHAW)
Author(s)	Gökcan Cantali (ZHAW), Péter Vörös, Mohammed Alshawki, Aya Khedda (ELTE), Antonios Lalas, Stelios Mpatziakas, Athanasios Papadakis, Virgilios Passas, Sarantis Kalafatidis, Nikolaos Makris, Donatos Stavropoulos, Nikolaos Vakakis, Vangelis V. Kopsacheilis, Thanasis Korakis, Anastasios Drosou (CERTH), Sumeyya Birtane, Shankha Gupta, Mays AL-Naday (UEssex), Maxime Vinh Hoa La (MONT)
Reviewers	Padial Leonardo (HES-SO) and Joaquín Escudero Garzás (GRAD)
Type	R – Document, Report
Dissemination level	PU – Public
Due date	M30 (30/06/2026)
Submission date	30/06/2026

Document version history

Version	Date	Changes	Contributor (s)
r1.v0.1	15/06/2025	Initial table of content (ToC)	ZHAW (Gürkan Gür)
r1.v0.2	18/06/2025	Updated ToC and partner assignments	All authors
r1.v0.3	20/06/2025	Contribution for Section 2	ZHAW
r1.v0.3.1	20/07/2025	Contribution for Section 3 and 5	MONT, ELTE
r1.v.0.4	13/09/2025	Contribution for Section 4 and 6	Mays Al-Naday, Gupta Shankha (UESSEX), Virgilios Passas, Stelios Mpatziakas (CERTH)
r1.v0.5	17/09/2025	TL and WPL-reviewed version	Gürkan Gür (ZHAW)
r1.v0.6	19/09/2025	Reviewed version	Edgardo Montes de Oca (MONT), Vincent Lefebvre (TSS)
r1.v0.7	24/09/2025	Refinement based on review comments	All authors
r1.v0.8	27/09/2025	Quality review	Joachim Schmidt (HES-SO), Leonardo Padial (HES-SO)
r1.v0.9	29/09/2025	Final review and refinements	Antonios Lalas, Virgilios Passas, Stelios Mpatziakas (CERTH) and CERTH team
r1.v1.0	30/09/2025	Final version for submission	Antonios Lalas (CERTH)
r2.v0.1	08/04/2026	Differential table of Contents, first round of contributions	Gürkan Gür, Wissem Soussi (ZHAW), All authors
r2.v0.2	10/04/2026	Updated ToC and partner assignments	All authors
r2.v0.3	08/05/2026	Contribution for Section 2	ZHAW
r2.v0.3.1	12/05/2026	Contribution for Section 3 and 5	MONT, ELTE
r2.v.0.4	28/05/2026	Contribution for Section 4 and 6	Mays Al-Naday, Gupta Shankha (UESSEX), Virgilios Passas, Stelios Mpatziakas, Athanasios Papadakis (CERTH)
r2.v0.6	09/06/2026	Reviewed version	Padial Leonardo (HES-SO) and Joaquín Escudero Garzás (GRAD)
r2.v0.7	17/06/2026	Refinement based on review comments	All authors
r2.v0.8	22/06/2026	Quality review	Joachim Schmidt, Leonardo Padial (HES-SO)
r2.v0.9	26/06/2026	Final review and refinements	Antonios Lalas, Evangelos V. Kopsacheilis (CERTH)
r2.v1.0	30/06/2026	Final version for submission	Antonios Lalas (CERTH)

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 6G-SNS. Neither the European Union nor the granting authority can be held responsible for them. The European Commission is not responsible for any use that may be made of the information it contains.

While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the NATWORK consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the NATWORK Consortium nor any of its members, their officers, employees, or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the NATWORK Consortium nor any of its members, their officers, employees, or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

© NATWORK Consortium. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation, or both. Reproduction is authorised provided the source is acknowledged.

Contents

Contents	5
List of acronyms and abbreviations.....	7
List of figures.....	9
List of tables.....	10
Executive summary	11
1. Introduction.....	12
1.1. <i>Purpose and structure of the document</i>	<i>12</i>
1.2. <i>Intended Audience</i>	<i>13</i>
1.3. <i>Interrelations.....</i>	<i>13</i>
1.4. <i>Advancements with respect to deliverable D3.3</i>	<i>14</i>
2. Resource-optimized MTD	15
2.1. <i>Novelty Beyond state-of-the-art</i>	<i>15</i>
2.2. <i>Live Migration Optimizer</i>	<i>15</i>
2.3. <i>Broader MTD Strategy Optimization</i>	<i>17</i>
2.4. <i>Advancements beyond D3.3</i>	<i>19</i>
3. XAI-Driven Intent-Based Security Monitoring and Enforcement	21
3.1. <i>Novelty Beyond state-of-the-art</i>	<i>21</i>
3.2. <i>Intent-Based Networking</i>	<i>22</i>
3.3. <i>XAI for Intent Verification and Policy Enforcement</i>	<i>26</i>
3.4. <i>Adaptive Security in Multi-Cloud and Dynamic Environments</i>	<i>28</i>
3.5. <i>Advancements beyond D3.3</i>	<i>31</i>
4. Adaptive orchestration optimisation for sustainability	33
4.1. <i>Novelty Beyond state-of-the-art</i>	<i>33</i>
4.2. <i>CTI-Driven Selective Sharing</i>	<i>33</i>
4.3. <i>Workload Prediction for Scheduling</i>	<i>35</i>
4.4. <i>Advancements beyond D3.3</i>	<i>35</i>
5. Secure Data Aggregation	37
5.1. <i>Novelty Beyond state-of-the-art</i>	<i>37</i>

5.2.	<i>Secure Data Aggregation Protocol</i>	38
5.2.1.	Architecture 1: One-Round Noise Aggregation	39
5.2.2.	Architecture 2: Checksum Aggregation	40
5.3.	<i>Attack Resilience</i>	42
5.4.	<i>Aggregation Integration</i>	43
5.5.	<i>Advancements beyond D3.3</i>	43
6.	Microservice profiling and anomaly detection - Microservice behavioural analysis	44
6.1.	<i>Novelty Beyond state-of-the-art</i>	44
6.2.	<i>Technical Description</i>	45
6.2.1.	Algorithms utilized for proposed solution	46
6.3.	<i>Microservice Profiling – Evaluation Results</i>	48
6.4.	<i>AI driven detection mechanism</i>	50
6.5.	<i>Advancements beyond D3.3</i>	54
6.5.1.	Technical Description	54
6.5.2.	Mitigation actions	55
7.	Conclusion	57
	References	58
	<i>Annex A Details on the architectures of the DNN used for Microservice profiling</i>	63

List of acronyms and abbreviations

Abbreviation	Description
ACLs	Access Control Lists
AI	Artificial Intelligence
AMF	Access and Mobility Management Function
ARP	Address Resolution Protocol
CNF	Containerized VNF
CNN	Convolutional Neural Network
CPU	Centralized Processing Unit
CRIU	Checkpoint Restore In Userspace
CTI	Cyberthreat Intelligence
DoST	Denial of Sustainability
DP	Differential Privacy
eDoS	Economic Denial of Sustainability
eNB	Evolved Node B
FL	Federated Learning
GDPR	General Data Protection Regulation (GDPR).
HSS	Home Subscriber Server
IDS	Intrusion Detection System
IoT	Internet of Things
LiMi	Live Migration
LSTM	Long Short-Term Memory
MAIP	Montimage AI Platform
MLP	Multi-Layer Perceptron
MOMDP	Multi-Objective Markov Decision Process
MPC	Multi-Party Computation
MTD	Moving Target Defense
NCL	Network Convergence Laboratory
NLP	Natural Language Processing
NS	Network Slice
PCRF	Policy and Charging Rules Function
RL	Reinforcement Learning
SA	Secure Aggregation
SDN	Software Defined Network
SLA	Service Level Agreement
SMF	Session Management Function
SSH	Secure Shell
SSLA	Security Service Level Agreement
STIX	Structured Threat Information Expression
UE	User Equipment
UPF	User Plane Function

Abbreviation	Description
VNF	Virtual Network Function
XAI	Explainable AI

List of figures

Figure 1: Container live migration setup	16
Figure 2: Data collected and used in the workflow of the MTD Strategy Optimizer.	18
Figure 3 Example of the proposed frontline “statelessification” process	20
Figure 4: An example of a slice isolation SSLA	25
Figure 5: A rule sample to monitor the network traffic in a slice and to act as a firewall.	25
Figure 6: XAI-driven Intent Verification and Policy Enforcement.....	27
Figure 7: Adaptive Security Framework in Multi-Cloud and Dynamic Environments	29
Figure 8: High-level functional view of the CTI selective sharing process.....	34
Figure 9 Secure Aggregation system Overview	39
Figure 10 Secure Aggregation - Architecture 1.....	40
Figure 11 Secure Aggregation - Architecture 2.....	41
Figure 12: Position of the microservice behavioural analysis module and interconnection to other modules.....	46
Figure 13 High level architecture of the proposed solution	48
Figure 14: CPU consumption forecast example for one of the containers contained in the dataset (step t=10).....	52
Figure 15: Memory Consumption (RSS) forecast example for one of the containers contained in the dataset (step t=3)	53

List of tables

Table 1 Evaluation results	48
Table 2 Results of the lightweight CNN that performs binary classification of the microservices resource usage (Normal/Anomalous).....	49
Table 3 Results of the CNN that performs multiclass classification of the microservices resource usage (7 classes: Normal/Five Anomalies/Unknown)	50
Table 4 Common actions used to mitigate the microservice anomalies examined.....	50
Table 5 Features contained in [3] and utilized for forecasting resource prediction	51

Executive summary

Dynamic 6G environments and diverse services demand robust, adaptive security frameworks. One of the aims of NATWORK is to enable secure-by-design composition and configuration of cloud-native in-network security services, and their optimal – dynamic – distribution within 6G edge-to-cloud continuum to meet Net-Zero targets. This deliverable is the successor of NATWORK deliverable D3.3 – “Intent-based service security and adaptive security techniques.r1”.

The current report presents the results and achievements of NATWORK consortium in the frame of Task 3.3 in period M22-M30. It marks a significant technical evolution, into a mature security architecture tailored for the 6G edge-to-cloud continuum. This report details the final design, implementation, and validation of proactive, self-resilient, and sustainable security functions. The technical advancements achieved during this phase span the following five pillars, establishing a robust framework designed to handle the complex, dynamic attack surfaces in 6G architectures:

- i. **Architectural Hardening (MTD):** Introduces "statelessification" to decompose full-stack applications, maximizing stateless components to reduce the predictable attack surface.
- ii. **Intent-Based Security:** Formalizes an automated pipeline translating natural-language intents into executable monitoring rules, hardened against adversarial attacks using SHAP-based feature attribution.
- iii. **Predictive Sustainability:** Integrates a lightweight AI workload-prediction microservice that distinguishes ordinary traffic from Denial of Sustainability (DoSt) energy-exhaustion exploits to enable proactive resource allocation.
- iv. **High-Utility Privacy (SEAN-FL):** Realizes a federated learning framework using distributed Differential Privacy with exact post-aggregation noise removal, eliminating the trade-off between data privacy and model utility.
- v. **High-Fidelity Validation:** Uses a proprietary 5G core/RAN simulation dataset to validate multi-stage AI detection against multi-vector threats, triggering automated remediation like SDN access control and dynamic pod scaling.

The above innovations move the project from foundational research into a suite of implemented, improved, and experimentally tested security solutions. These integrated assets are fully prepared for deployment within the NATWORK testbed infrastructure, ensuring direct alignment with the envisioned use cases and validation criteria defined for the project.

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 Task 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 *D3.4 Intent-based service security and adaptive security techniques.r2* 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 *D3.4 Intent-based service security and adaptive security techniques.r2* 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.

1.4. Advancements with respect to deliverable D3.3

This deliverable represents a significant technical evolution from the initial frameworks established in D3.3, moving past early-stage conceptual designs and proof-of-concept implementations toward mature, high-fidelity architecture with more robust network security contributions. While the core structure and content of the document remain the same as D3.3, each section now carries a dedicated subsection walking through new technical developments since D3.3.

In the domain of Resource-Optimized Moving Target Defense (MTD), a key innovation introduced here is the principle of "statelessification," which decomposes full-stack applications to maximize stateless components. This facilitates proactive "cleaning" of the attack surface and provides specialized side-channel defenses by dynamically relocating critical stateful components across physical nodes.

Concerning XAI-Driven Intent-Based Security, efforts have been directed toward the formal specification and experimental validation of the Security Service Level Agreement (SSLA) pipeline. This version documents the structured procedure through which natural-language security intents are translated into concrete MMT-probe monitoring rules. In addition, the adversarial robustness of the AI models has been empirically assessed against both evasion and poisoning attacks, with SHAP-based feature attribution now informing the hardening process.

The work on Adaptive Orchestration for Sustainability has been extended through the introduction of a lightweight AI microservice dedicated to workload prediction. This service differentiates between ordinary traffic trends and anomalous signatures, including Denial of Sustainability (DoST) attacks. This predictive capability enables the orchestrator to anticipate demand fluctuations and carry out proactive resource allocation, contributing directly to the project's Net-Zero objectives. Regarding Secure Data Aggregation, the SEAN-FL architecture has reached full realization, adding the integration of distributed Differential Privacy in a configuration that permits exact noise removal following aggregation, thereby resolving the conventional tension between privacy preservation and model utility.

Finally, the Microservice Profiling and Anomaly Detection module has undergone a deliberate methodological transition, moving from open-dataset validation toward analysis of proprietary operational data. A purpose-built dataset was generated from a custom Docker-based simulation environment encompassing a 5G core and RAN setup, providing a high-fidelity basis for evaluation. The module now incorporates multi-stage AI-driven detection mechanisms that have been assessed against targeted attack scenarios, including Slowloris and brute force attacks, with automated mitigation responses such as SDN-based access control and dynamic pod scaling integrated into the remediation workflow.

2. Resource-optimized MTD

As defined in the previous deliverable of this work package (*i.e.*, D3.1-“ Secure-by-design orchestration and management & Data plane computation offloading.r1”), 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. Novelty Beyond state-of-the-art

The MTD Strategy Optimizer advances the state-of-the-art in CNF-based MTD by moving beyond "one-size-fits-all" migration strategies to a highly tailored, ML-driven approach. While current state-of-the-art solutions like CloudHopper [2] apply the same pre-copy migration method to all containers and schedule migrations based solely on checkpoint size, the MTD strategy optimizer leverages ML to select the optimal migration method (between basic, pre-copy, post-copy, or hybrid) for each individual container based on its specific resource profile, *i.e.*, CPU, RAM, disk, and network. This fine-grained optimization allows the MTD strategy optimizer to reduce the live migration downtime compared to existing methods. Additionally, the service employs a regression-based scheduler that predicts migration times for different container loads to ensure all components of a complex microservice application restore simultaneously at the destination, preventing application-level errors during the transition.

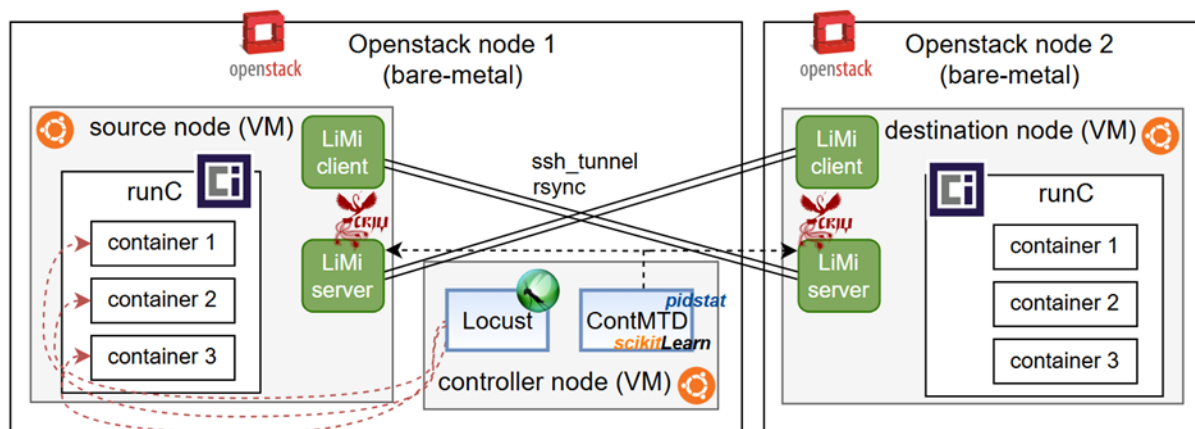
2.2. Live Migration Optimizer

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.

- 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*, a tool command, optimized to transmit only the incremental differences between the new checkpoint and other possibly existing images of the container at the destination.

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.



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 MTD Strategy Optimizer 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.

2.3. 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 [5]. 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, linked to the data collected on the security objective (depicted in Figure 2); the network-state modelling module, using multi-objective Markov Decision Process (MOMDP); and the deep multi-objective RL (MORL) agent [6] 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.

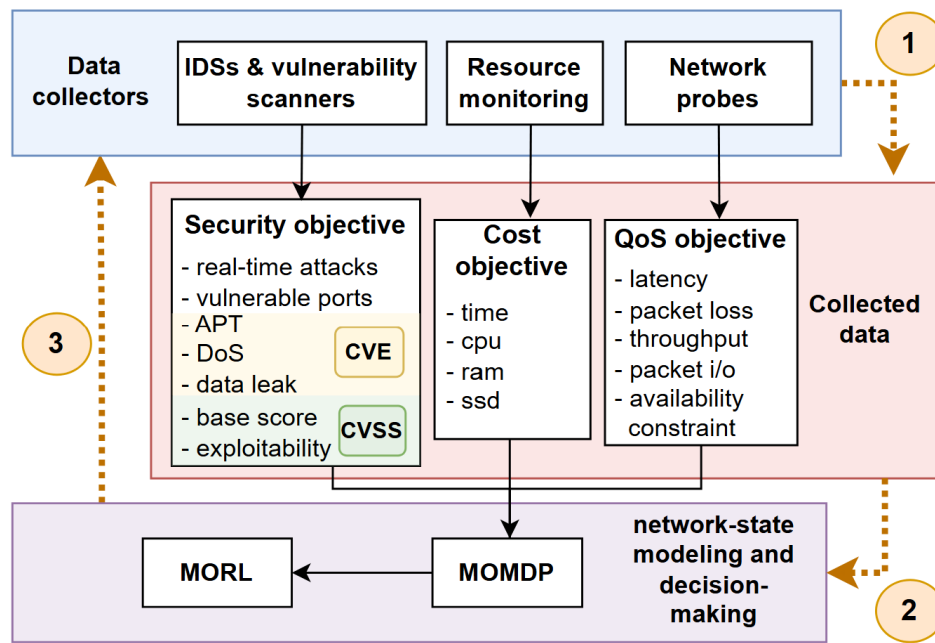


Figure 2: Data collected and used in the workflow of the MTD Strategy Optimizer.

Figure 2 details the workflow and data gathered for the MTD strategy optimization focusing on collecting metrics to quantify the effects of the MTD operations on the network state with respect to the above-mentioned three main objectives. 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)[7]. The ASP is then estimated using the CVSS base score and exploitability score of the vulnerabilities found with the risk assessment [8].

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: $\beta = -0.082$, $\alpha_1 = 0.031$, $\alpha_2 = 0.004$, $\alpha_3 = 0.00006$.

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[9].

To conclude, the solution provides recommendations on when to perform which MTD actions (e.g., migrating a VNF/CNF to another hypervisor when the RI.AS. module detects a side-channel vulnerability) 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.

2.4. Advancements beyond D3.3

This section details the advanced architectural and strategic features of the MTD Strategy Optimizer that extend beyond the LiMi-optimized mechanism implemented and described in D3.3; to provide a proactive, security-oriented defense for microservice environments. A central innovation is the design principle of frontline "statelessification", which is defined as the decomposition of a full-stack application to maximize its stateless components while isolating stateful ones to separate containers. To manage these components, the framework considers implementing MTD strategies where the containers of a service are classified in risk-based security zoning that categorizes containers into three levels of exposure (depicted in Figure 3):

1. Red Zone (High Risk): Includes front-facing containers, such as API gateways and frontend GUIs. These should be stateless CNFs that directly handle external user requests and are most vulnerable to initial exploitation.
2. Orange Zone (Medium Risk): Comprises internal stateful services like authentication and messaging systems, which have a reduced risk profile but can only be migrated via stateful LiMi.
3. Green Zone (Minimal Risk): Consists of the most critical stateful components, such as primary databases and in-memory caches, which are shielded from intrusion by the preceding zones.

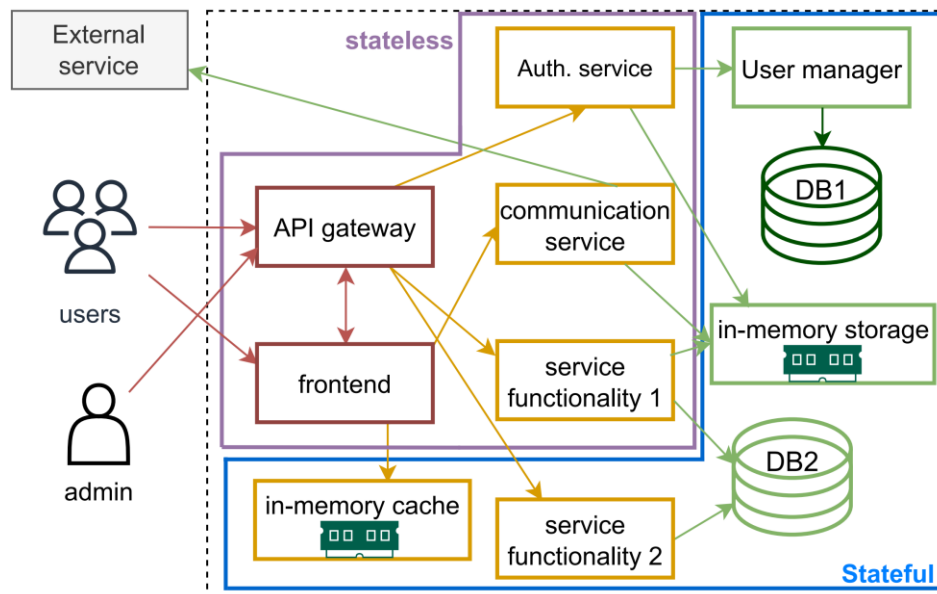


Figure 3 Example of the proposed frontline "statelessification" process

Proactive "Cleaning" and Side-Channel Defense of the MTD Strategy Optimizer leverages this zoning to solve the problem of "stateful infection persistence," where a migrated stateful container simply carries its malware to the new host. The framework enforces an MTD security policy that treats stateless containers as a "buffer security zone". By performing frequent and seamless migrations of these exposed stateless services in the green zone, the system effectively "cleans" the attack surface of undetected malware and Advanced Persistent Threats (APTs) before they can penetrate deeper into the orange and red zones. While these frequent stateless migrations act as an infection-cleansing mechanism, the framework employs stateful LiMIs for the orange and red zones as a proactive measure specifically against side-channel attacks (e.g., hypervisor's escape and passive data exfiltration). By dynamically relocating these critical databases across different physical nodes, the MTD Strategy Optimizer narrows the time window an attacker has to exploit hypervisor vulnerabilities and exfiltrate sensitive data from neighboring containers.

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

The increasing complexity of 5G/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. Novelty Beyond state-of-the-art

The contribution to this section advances the state-of-the-art in intent-based security monitoring along three orthogonal dimensions: formal intent operationalisation, explainability-driven enforcement, and deep integration of network traffic analysis through MMT-probe.

From natural language intents to executable monitoring rules. Existing IBN solutions typically stop at translating intents into configuration templates, relying on operators to manually bridge the gap between policy intent and runtime enforcement logic. The MAIP (Montimage AI Platform, an open-source security monitoring and enforcement framework) approach closes this gap by introducing a structured pipeline that takes a natural-language or semi-formal security intent, extracts its semantic constraints through rule-matching and semantic analysis, and produces executable monitoring rules directly deployable in MMT (Montimage Monitoring Tool). This end-to-end formalisation reduces the risk of misconfiguration and ensures that the enforcement layer faithfully reflects the operator’s original intent, without requiring intermediate manual steps.

Explainability as a first-class enforcement property. State-of-the-art network security systems are increasingly AI-driven, but their decisions remain opaque to operators. MAIP distinguishes itself by embedding XAI techniques — specifically SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations) — directly into the enforcement loop, not as a post-hoc reporting layer, but as an integral component of the detection and remediation pipeline. When an anomaly is detected or a policy violation occurs, the system does not merely raise an alert: it pinpoints which specific intent or policy was breached, identifies the root cause at the traffic feature level, and accompanies any automated remediation action with a human-

readable justification. This is in contrast to prior work, where XAI is typically applied offline or for model debugging, rather than for real-time operational accountability.

Deep integration with MMT-probe for high-performance SSLA enforcement. The Security SLA Assessment Function described in Section 3.2 is operationalised using MMT-probe (Montimage Monitoring Tool), an open-source, high-performance network monitoring engine capable of analysing traffic at line rate in 5G/6G environments. MMT-probe embeds a rule engine that directly executes the formalised SSLA rules — such as the slice isolation firewall rule shown in Figure 4 — in real time, by deep-inspecting packet flows against the declared SSLA metrics. This tight coupling between the SSLA formalisation layer and the low-level monitoring engine is a key differentiator: unlike solutions that rely on generic SNMP (Simple Network Management Protocol) or telemetry collection pipelines, MMT-probe evaluates security intents directly at the packet capture and protocol dissection level, minimising detection latency. The combination of MAIP’s AI and XAI layer with MMT-probe’s real-time rule execution thus provides a uniquely integrated solution that spans from human-readable security intents down to per-packet enforcement.

Adversarial robustness integrated into the monitoring pipeline. Most AI-driven intrusion detection and policy enforcement systems assume a benign data distribution at inference time. MAIP explicitly accounts for adversarial conditions by integrating an adversarial robustness module that evaluates and hardens the underlying AI models against evasion and poisoning attacks — including random label flipping, targeted label flipping, and Generative Adversarial Network (GAN)-based adversarial perturbations. This adversarial training loop is co-designed with the XAI module: SHAP-based feature attribution is used not only to explain predictions but also to identify which input features an adversary could manipulate most effectively, guiding the hardening strategy. To our knowledge, this adversarially-aware, explanation-guided approach to intent-based security monitoring has no direct precedent in the existing literature.

3.2. Intent-Based Networking

Intent-based networking (IBN) [10] 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 5G/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 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 4: 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 4 presents an example of a slice isolation SSLA. Note that this is just one way of defining the isolation of slices.

This is translated to the rule presented in Figure 5 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 5: 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.

3.3. 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, 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.

¹ Montimage AI Platform (MAIP): <https://github.com/Montimage/maip>

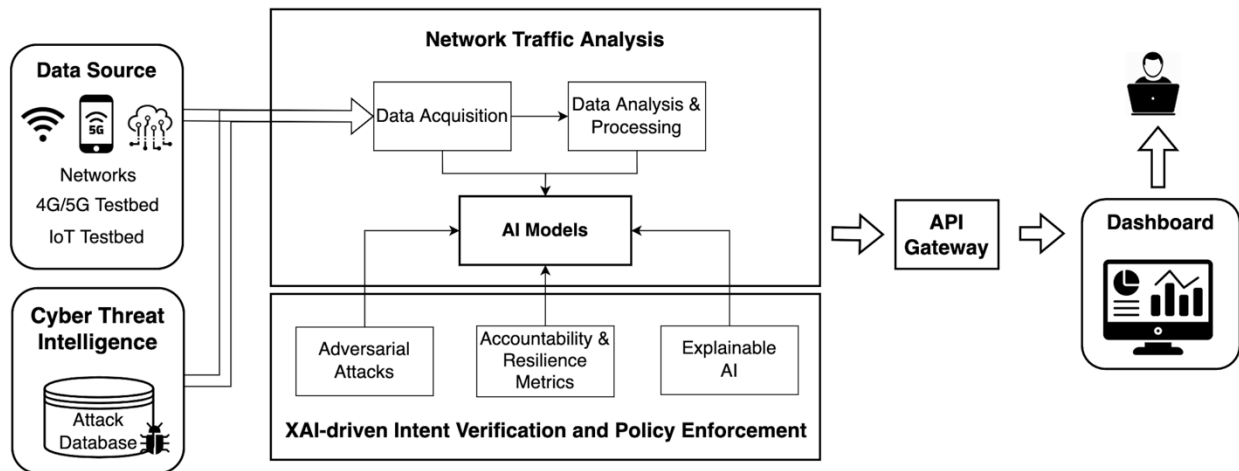


Figure 6: XAI-driven Intent Verification and Policy Enforcement

Figure 6 illustrates how the MAIP operationalises the two-step process described above. In the first step (policy translation and mapping), natural-language security intents are processed using semantic analysis against multiple data sources — including 4G/5G and IoT testbed traffic and cyber threat intelligence feeds — to produce machine-enforceable monitoring rules. In the second step (explainable anomaly detection), network traffic is continuously acquired and analysed by AI models; when a deviation is detected, the XAI module produces human-understandable explanations that identify which intent was violated and why, supporting both automated remediation and operator oversight. The figure also shows the adversarial robustness loop: AI models are hardened against evasion and poisoning attacks, with accountability and resilience metrics fed back to improve enforcement quality over time.

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][11] and LIME (Local Interpretable Model-Agnostic Explanations) [12][13], 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.

3.4. 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.

ADAPTIVE SECURITY IN MULTI-CLOUD AND DYNAMIC ENVIRONMENTS

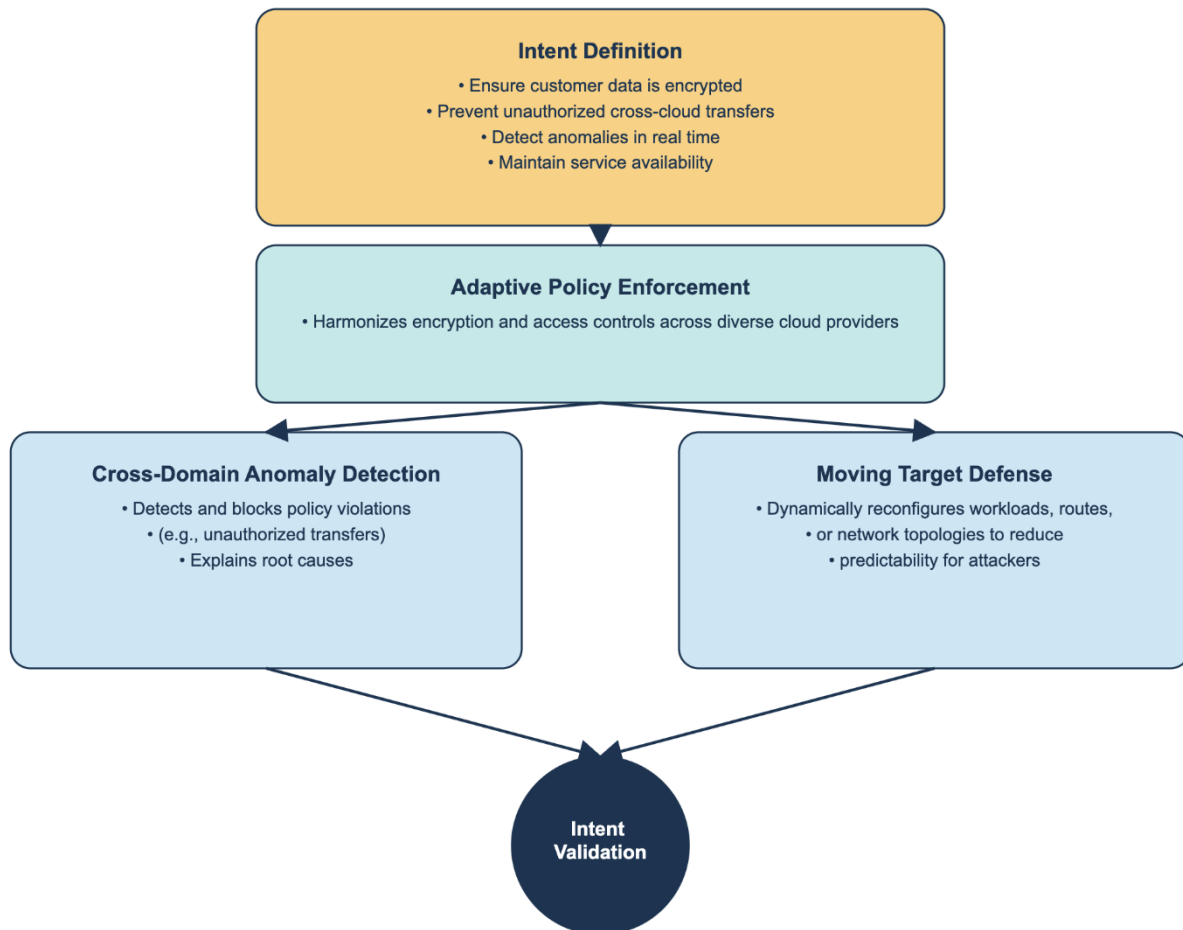


Figure 7: Adaptive Security Framework in Multi-Cloud and Dynamic Environments

Figure 7 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 the Montimage (MAIP) contribution in this section is its integration with the Moving Target Defence framework 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 NATWORK deliverable D3.1-“Secure-by-design orchestration and management & Data plane computation offloading.r1”, 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.

3.5. Advancements beyond D3.3

Building on the XAI-driven intent-based security framework presented in NATWORK deliverable D3.3-“ Intent-based service security and adaptive security techniques.r1”, the advancements reported in this deliverable consolidate three lines of progress: the formalisation and experimental validation of the SSLA assessment pipeline, the hardening of the MAIP AI models against adversarial threats, and the deeper integration of MMT-probe as the runtime enforcement engine for intent-derived security rules.

SSLA pipeline formalisation and validation. D3.3 introduced the SSLA Assessment Function and illustrated its operation through a slice isolation use case. In the present deliverable, this pipeline has been further formalised: the mapping from natural-language security intents to SSLA metric definitions, and from SSLA definitions to MMT-probe monitoring rules, is now documented as a structured procedure applicable across a wider range of intent types — including encryption enforcement, geolocation compliance, and access isolation. The intent-to-rule translation has also been extended to cover multi-condition SSLAs, where enforcement depends on the conjunction or disjunction of multiple network observables. This broadens the expressiveness of the SSLA model beyond the single-condition examples illustrated in D3.3 and aligns the formalisation more closely with realistic operator policies in B5G/6G environments.

Adversarial robustness evaluation of MAIP AI models. D3.3 described the adversarial attack and defence capabilities of the MAIP XAI component at a design level. In this deliverable, these capabilities have been evaluated experimentally. The adversarial attacks module was used to inject evasion and poisoning attacks — including random label flipping and targeted label flipping — into the traffic classification models used for anomaly detection. Adversarial training, guided by SHAP-based feature attribution to identify the most attack-sensitive features, was applied to harden the models. The resulting models demonstrate improved robustness while maintaining detection performance, providing a concrete empirical basis for the adversarial resilience claims made in D3.3. These results also inform the identification of traffic features that require careful monitoring in operational deployments, particularly in 4G/5G and IoT testbed environments.

MMT-probe as the SSLA enforcement runtime. A key advancement over D3.3 is the explicit positioning of MMT-probe as the runtime engine that executes the SSLA-derived monitoring rules. While D3.3 described the SSLA Assessment Function as an abstract AI-driven component, this deliverable clarifies how MMT-probe operationalises that function in practice. MMT-probe embeds a high-performance packet capture and protocol dissection engine alongside a rule execution environment that evaluates Boolean expressions over traffic observables in real time. Intent-derived rules — of the form illustrated in Figure 4 — are compiled into MMT-probe’s rule language and executed at line rate, enabling sub-second detection of SSLA violations even in high-throughput B5G/6G traffic scenarios. This closes the loop between the abstract MAIP intent

verification layer and the concrete packet-level enforcement plane and establishes a path towards end-to-end integration testing in the NATWORK testbed environment targeted for WP6.

Strengthened integration with the MTD framework. D3.3 introduced the concept of integrating MAIP’s adaptive enforcement with the MTD Strategy Optimizer developed by ZHAW. In this deliverable, this integration is further articulated through the Adaptive Security Framework described in Section 3.4. Specifically, the interface between MAIP’s cross-domain anomaly detection and the MTD Strategy Optimizer’s live migration decisions has been designed in greater detail: upon detection of a policy violation or anomalous data flow, MAIP can now signal the MTD orchestration layer to trigger a targeted migration or workload reallocation, with the specific action and its rationale logged as an explainable remediation record. This tighter coupling between the XAI-driven enforcement plane and the MTD execution plane strengthens the overall closed-loop security architecture and prepares the ground for joint evaluation in the NATWORK integration phase.

4. Adaptive orchestration optimisation for sustainability

The optimisation strategy plugged into the orchestration level combines cybersecurity and sustainability objectives. As introduced in NATWORK deliverable D3.1-“Secure-by-design orchestration and management & Data plane computation offloading.r1” , 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. Novelty Beyond state-of-the-art

NATWORK introduces novel metadata models that enables domains to express the sensitivity levels of their CTI data, while specifying the necessity requirements for CTI data published by their peers. The two data models are of similar structure, each one provides a scale-based ranking of every data element in a CTI object (STIX Data Object – SDO). A novel governance sharing strategy is then developed to reconcile the sensitivity restrictions with the necessity requirements and calculate a trade-off that maximise the satisfaction of necessities of subscribers to CTI without violating the sensitivity constraints of the publisher counterpart. The solution has been evaluated over the Network Convergence Laboratory (NCL) testbed and shows the allocation of necessary 6G slice resources across the edge-cloud domains, to achieve sustainable improvement in sharing cyber hygiene information for observability. This has further enabled novel ability to optimise the distribution of a 6G slice over edge-cloud domains, based on the cybersecurity requirements and status of both the slice and participating domains.

4.2. CTI-Driven Selective Sharing

The CTI framework provides mechanism to calculate hygiene scores from vulnerability data (the calculation function may differ for each domain, depending on domain’s objectives, but generally they can be a direct function of the vulnerability distribution), 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.

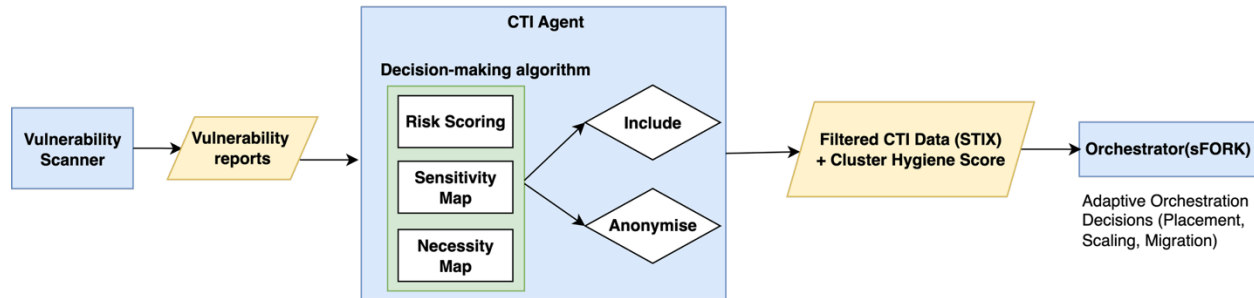


Figure 8: High-level functional view of the CTI selective sharing process

The overall workflow of the CTI selective sharing framework is illustrated in Figure 8. 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.

4.3. Workload Prediction for Scheduling

The workload prediction service runs as a lightweight AI microservice integrated with the orchestration layer. It interacts with Prometheus [14] 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 the 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-“Secure-by-design orchestration and management & Data plane computation offloading.r1”: 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.

4.4. Advancements beyond D3.3

Complementary to CTI adaptive sharing, NATWORK has developed novel AI-based workload prediction which takes into account the adversity of demand at the edge, leading to bursty workload patterns. Novel solution has been developing for realistic aggregation and sampling of workload indicators. Particularly, aggregating over short periods then sampling them within a longer period. This has allowed for capturing the bursty nature of edge workload in a realistic manner without inflating resource provisions due to aggregation over long periods. The solution

has been adopted into a federated learning system and currently being validated in UESSEX NCL testbed. Initial results show considerably higher adaptability to workload needs compared to classic ML solutions. It shows that a number of algorithms, including XGBoost, LSTM and ARIMAX to achieve comparable prediction accuracy. However, XGBoost has the lowest training cost and delay to achieve the observed performance.

5. Secure Data Aggregation

In 5G/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.

5.1. Novelty Beyond state-of-the-art

Existing privacy-preserving federated learning approaches mainly rely on either Differential Privacy (DP) [15] or Secure Multiparty Computation (MPC). DP-based methods protect client updates by adding noise either locally after the model training, *i.e.*, Local Differential Privacy (LDP) [18][19], or at the server during aggregation, *i.e.*, Central Differential Privacy (CDP) [20][21]. Although effective for privacy preservation, these approaches often degrade model accuracy due to the irreversible impact of noise. Several works attempt to reduce this trade-off using techniques such as gradient clipping [22][23], adaptive privacy budgets[24][25], or reinforcement learning-based parameter tuning [26].

MPC-based approaches provide stronger confidentiality guarantees by securely aggregating client updates without revealing them to any single party. However, fully MPC-based aggregation introduces significant communication and computational overhead. To improve scalability, prior

works propose hierarchical aggregation [27][28], chain masking schemes [29][30], and compression and transformation techniques [31][32][33]. Despite these optimizations, repeated MPC interaction during training remains costly for large-scale deployments.

More recent hybrid DP-MPC approaches combine both mechanisms to improve privacy preservation. Some methods rely on pairwise client communication [34][35] and distributed noise sharing [36], while others integrate DP with secure aggregation protocols [37][38]. However, these solutions still either suffer from accuracy degradation caused by permanent DP noise or require substantial communication overhead.

In contrast, the proposed architectures A1 and A2 (see next section) combine DP and MPC in a lightweight manner that preserves both privacy and model utility. Instead of performing full aggregation inside MPC, the proposed systems retain server-based aggregation and use MPC only for auxiliary tasks.

5.2. Secure Data Aggregation Protocol

The proposed protocol (SEAN-FL) supports two hybrid aggregation architectures: Architecture 1 (A1), based on one-round noise parameter aggregation, and Architecture 2 (A2), based on partitioned noise aggregation and integrity verification. Both approaches combine Differential Privacy with MPC-based coordination to achieve privacy-preserving and scalable federated learning, while avoiding the high overhead of fully MPC-based aggregation. The key idea is to inject noise at the client side for privacy, while using MPC nodes only for tasks such as noise reconstruction and verification, rather than performing full model aggregation within MPC. Figure 9 shows the system overview of the secure aggregation protocol.

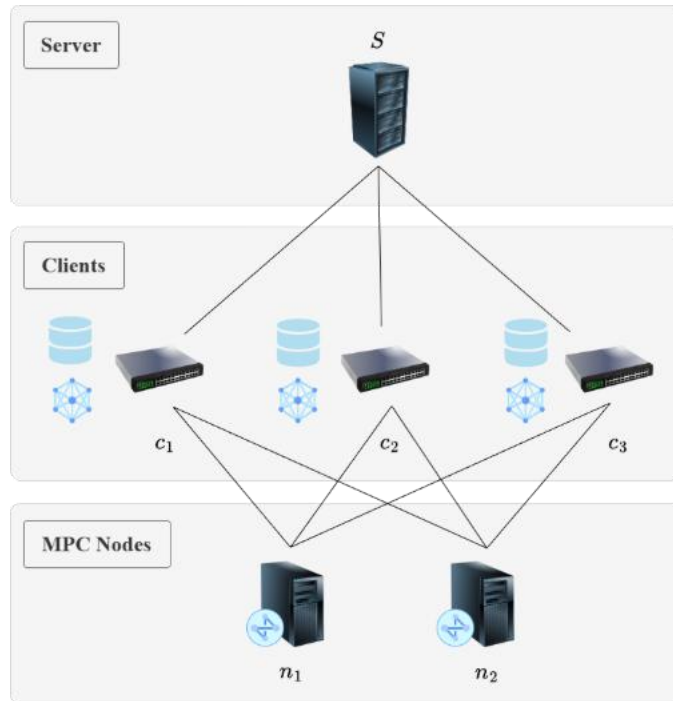


Figure 9 Secure Aggregation system Overview

5.2.1. Architecture 1: One-Round Noise Aggregation

In Architecture 1, the protocol operates in two distinct phases designed to minimize interaction with MPC nodes while preserving privacy guarantees. The protocol has been illustrated in Figure 10. During the initialization phase, clients securely aggregate noise parameters using MPC nodes only once before the federated learning process begins. Each client locally computes its Differential Privacy noise parameters and splits them into additive secret shares, which are distributed across multiple MPC nodes. These nodes aggregate the received shares without accessing the original values and return partial aggregates to the clients. By combining these partial results, all clients reconstruct a common set of global noise parameters that will be reused throughout training.

After initialization, the training process proceeds independently of the MPC nodes. In each training round, all clients generate the same noise value using the shared parameters and a common randomness seed. Each client perturbs its local model update with this noise before sending it to the federated learning server. The server performs Federated Averaging on the noisy updates and returns the aggregated noisy global model. Because the same noise is consistently applied across all client updates, it allows clients to remove it locally after receiving the result. This ensures that the exact global model is recovered without loss of accuracy. By avoiding repeated MPC interaction on all rounds, this architecture significantly reduces communication and computational overhead while maintaining correctness and privacy under the semi-honest assumption.

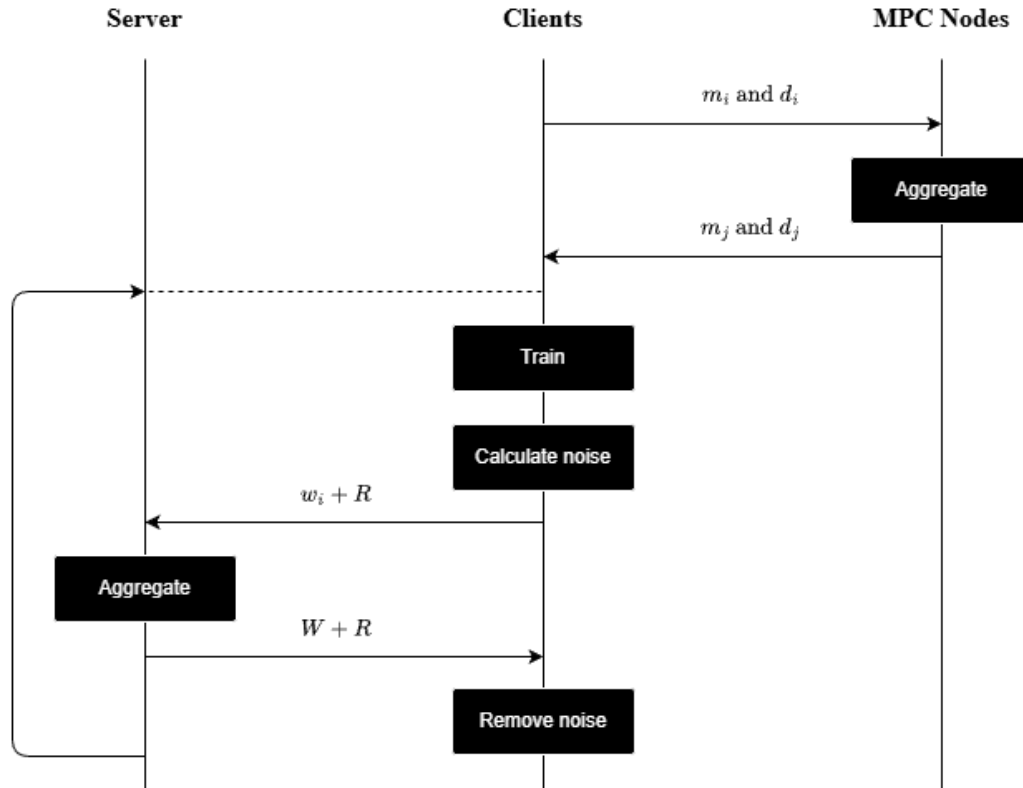


Figure 10 Secure Aggregation - Architecture 1

The server performs Federated Averaging (FedAvg) on the noisy updates and returns the aggregated noisy global model to the clients. Since the same noise was added by all clients, the clients can locally remove the noise after aggregation and recover the exact global model. This architecture preserves the correctness of aggregation while significantly reducing communication overhead. Since MPC interaction occurs only once during initialization, the runtime cost during training remains close to baseline federated learning.

5.2.2. Architecture 2: Checksum Aggregation

Architecture 2 as shown in Figure 11 strengthens security by introducing distributed noise partitioning and checksum-based integrity verification. In each training round, every client generates a local random noise vector and shuffles it using a shared permutation pattern. The shuffled noise is partitioned into multiple shares and distributed among the MPC nodes. This shuffling step hides the positional relationship between noise values and model parameters, preventing any single node from inferring meaningful information even if it receives multiple shares. The partitioning further ensures that reconstructing the full noise requires collaboration across all nodes.

While MPC nodes aggregate noise shares, clients simultaneously perform local model training and compute a checksum over their model parameters. Clients send noisy model updates to the

federated learning server while checksum shares and noise partitions are transmitted to MPC nodes. The checksum is designed to be lightweight and linear, enabling efficient aggregation without revealing individual model parameters, while still capturing both magnitude and structural information of the updates. The server performs Federated Averaging on the noisy updates and returns the aggregated noisy global model.

After receiving the aggregation results from the MPC nodes, clients reconstruct the aggregated noise, remove it from the received global model, and compute a checksum over the recovered model. The aggregation is accepted only if the reconstructed checksum matches the checksum reconstructed from MPC outputs. This verification step introduces an integrity check, ensuring that any deviation in the aggregation process is detected by all clients independently. This design provides stronger protection against malicious behavior. Additionally, because noise is generated independently at each round and never reused, this architecture mitigates risks associated with noise reconstruction and provides improved resistance against server-client collusion compared to Architecture 1.

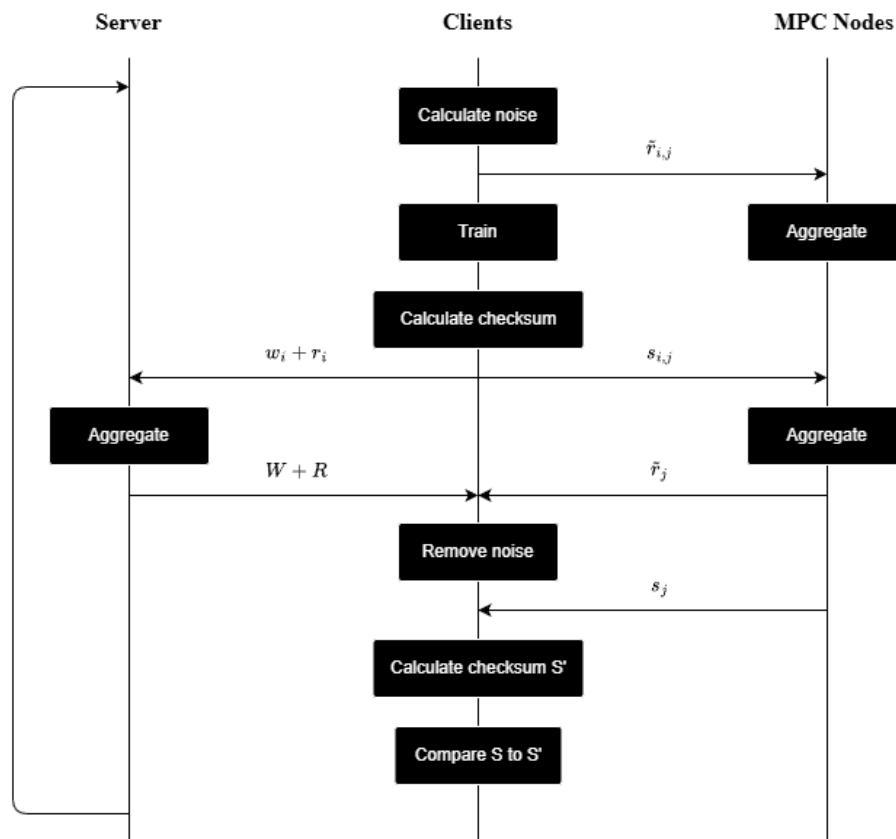


Figure 11 Secure Aggregation - Architecture 2

In parallel, MPC nodes aggregate the checksum shares and noise partitions and send the resulting aggregates back to the clients. Clients reconstruct the aggregated noise, remove it from the

received global model, and compute a checksum over the recovered model. The aggregation is accepted only if the reconstructed checksum matches the checksum reconstructed from MPC outputs.

This design provides stronger protection against malicious behaviour. Even if the server attempts to manipulate the aggregation result, clients can detect inconsistencies through checksum verification.

5.3. Attack Resilience

The secure aggregation of federated data provides resilience against several types of adversarial attacks, commonly encountered in federated learning and particularly in dynamic B5G/6G environments. In this section, some major security attack [39] against the described protocol have been slightly highlighted.

The proposed hybrid aggregation architectures provide resilience against adversarial threats commonly encountered in federated learning systems [40]. The primary privacy objective of both architectures is to protect local model updates from being reconstructed by the aggregation server or external adversaries. This protection is achieved through Differential Privacy noise injection, since the server only receives noisy updates, it cannot directly recover sensitive client information.

In Architecture 1, privacy is preserved under a semi-honest threat model as long as the server does not collude with a participating client. Since all clients share the same reconstructed noise parameters, a compromised client could reveal the common noise and enable the server to recover the true updates. Consequently, A1 prioritizes scalability and low overhead over strong collusion resistance.

Architecture 2 provides stronger resilience against malicious aggregation. Client-generated noise is shuffled, partitioned, and distributed among MPC nodes, making reconstruction significantly more difficult. Even if the server colludes with some entities, reconstructing the original updates requires access to multiple MPC nodes and knowledge of the private partitioning and permutation structure.

The checksum verification mechanism further strengthens security by ensuring soundness of computation. A malicious server attempting to manipulate the aggregated global model cannot forge a valid checksum without being detected. Clients independently verify the integrity of the recovered global model before accepting the aggregation result.

5.4. Aggregation Integration

The proposed secure aggregation architectures can support privacy-preserving AI services and distributed analytics in next-generation 5G/6G environments. Since federated learning relies on aggregating distributed model updates without exposing local data, efficient and scalable privacy-preserving aggregation becomes an essential enabling component.

Architecture A1 is particularly suitable for large-scale deployments where communication efficiency and scalability are critical. Because MPC interaction is limited to a one-time initialization phase, the architecture achieves latency close to baseline federated learning while still protecting both local updates and the aggregated global model. This makes A1 highly appropriate for environments involving a large number of distributed clients or resource-constrained edge devices, where repeated interaction with MPC nodes would introduce unacceptable communication overhead. Architecture A2 is better suited for scenarios requiring stronger adversarial robustness and integrity guarantees. Through checksum aggregation, shuffled noise partitioning, and distributed reconstruction, A2 enables clients to verify the correctness of the aggregation result while preserving privacy. This makes the architecture particularly valuable in environments where the aggregation server cannot be fully trusted or where malicious manipulation of model updates is considered a realistic threat.

Another important advantage of the proposed systems lies in their flexibility. Since A1 and A2 provide different trade-offs between efficiency and security, the aggregation mechanism can be selected dynamically according to operational requirements. In low-risk environments, A1 can provide lightweight and efficient aggregation with minimal communication overhead. In contrast, A2 can be deployed in higher-risk environments where integrity verification and stronger collusion resistance are required. This adaptability allows the federated learning system to balance performance, scalability, and security according to the current threat model and deployment scenario.

5.5. Advancements beyond D3.3

The advancements following D3.3 include the full proposal of SEAN-FL architectures that introduce several key advancements over standard federated learning aggregation mechanisms. First, they integrate distributed Differential Privacy in a way that preserves model accuracy by enabling exact noise removal after aggregation, eliminating the traditional privacy - utility tradeoff associated with DP. In Architecture A1, this is achieved through a one-time aggregation of noise parameters, allowing all clients to generate identical noise during training while avoiding repeated interaction with MPC nodes. Architecture 2 introduces distributed noise partitioning with shuffling and integrity verification. The shuffled noise partitions prevent reconstruction of client updates, while the checksum allows clients to detect malicious behavior from the server.

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. Novelty Beyond state-of-the-art

Current research in microservice security and performance monitoring has established a foundation for detecting system deviations through multi-stage identification and temporal-spatial analysis. For instance, two-stage detection frameworks have been proposed that first utilize LSTM networks to learn sequential execution patterns from service logs, followed by a second stage that integrates these log representations with spatial service query traces to define normal sample distributions using one-class classifiers [41]. Other approaches, have demonstrated the utility of MLP models in distinguishing between application-level infrastructure stress (such as CPU and disk exhaustion) and service-level functional anomalies (like increased response times), showing that service-specific metrics often yield higher detection accuracy [42].

Beyond static classification, state-of-the-art systems have increasingly focused on resource behaviour forecasting to support autoscaling integrity. Existing mechanisms use univariate autoregressive models to predict near-future CPU and network resource usage, enabling the identification of illegitimate workloads like Economic Denial of Sustainability or Economic Denial of Service (eDoS) attacks that trick autoscaling engines into wasting resources [43]. Furthermore, the application of NLP techniques to distributed tracing data like the Linux Trace Toolkit Next Generation (LTTng) which allows for the detection of performance anomalies and regressions by treating system events as words following specific rules [44]. These methodologies often leverage SDN-based infrastructures, to monitor and scale microservices within containerized environments [45].

Building upon these established methodologies, the proposed microservice behavioural analysis module introduces several innovations that distinguish it from traditional monolithic security and monitoring solutions. Unlike single-layer detection systems, this approach utilizes a multi-stage hierarchical detection method featuring a two-stage 1D-CNN architecture. A lightweight first stage performs rapid binary classification to identify anomalous resource usage, followed by a more robust second stage that discerns specific anomaly types, identifies false positives, or flags "unknown" patterns that require operator inspection.

To move beyond reactive detection, the integration of a LSTM deep neural network enables proactive temporal forecasting. By capturing temporal dependencies and nonlinear patterns in time-series metrics, the module generates regression-based predictions of near-future CPU and memory consumption, facilitating proactive autoscaling and performance optimization. The framework further utilizes cross-domain telemetry correlation by linking infrastructure-level system resources with application-level performance metrics.

Another central novelty is the SDN-enabled automated mitigation facilitated by the closed-loop integration between the AI detection mechanism, the microservice orchestrator, and the SDN controller. This allows for the dynamic execution of mitigation actions, like traffic rerouting via the SDN controller, immediately upon the detection of deviations. Finally, the architecture features 5G microservices, specifically designed for the decentralized nature of B5G/6G networks where functions are independent, autonomous components to ensure real-time adaptability and efficient resource management.

6.2. 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 12 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 12 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 [46].

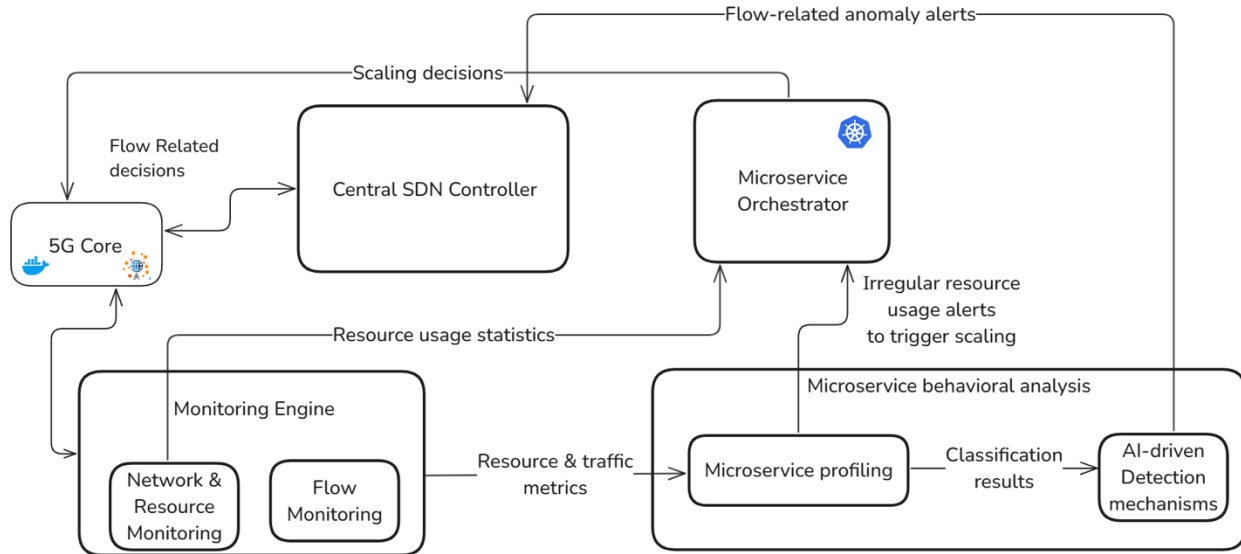


Figure 12: Position of the microservice behavioural analysis module and interconnection to other modules

6.2.1. 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 [47]. 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 [46].

A two-stage anomaly detection approach is followed. First, a lightweight 1D-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.
- Unknown, 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 [4][48]. 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 13 presents the envisioned high-level architecture of the proposed solution.

Annex A contains detailed information on the architecture of the Neural networks utilized.

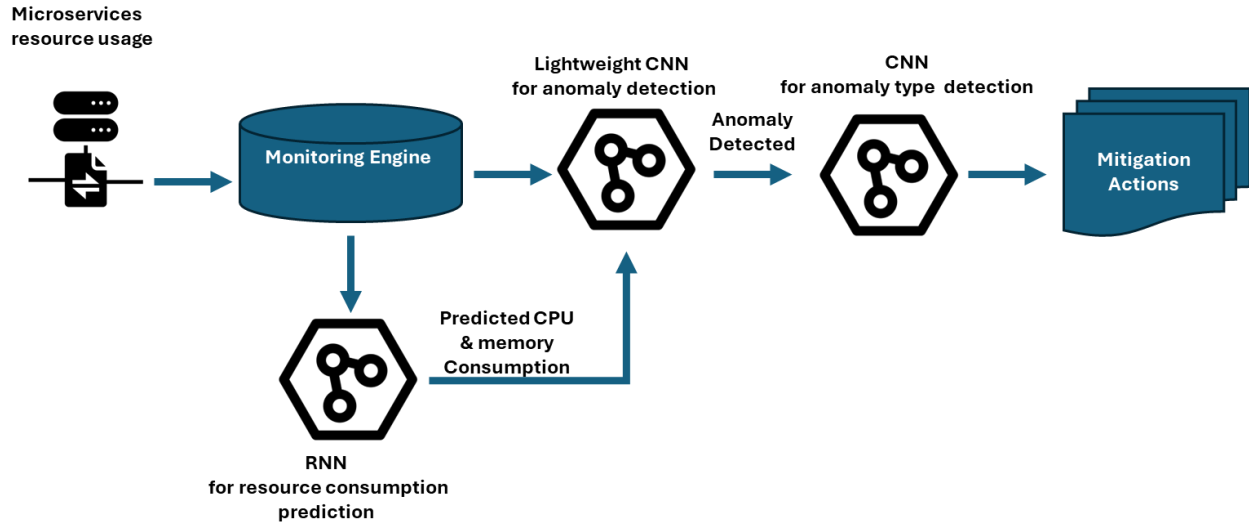


Figure 13 High level architecture of the proposed solution

6.3. 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 [49].

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 [49]. 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.

6.4. 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 [3], 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 [3] 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.
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 14.

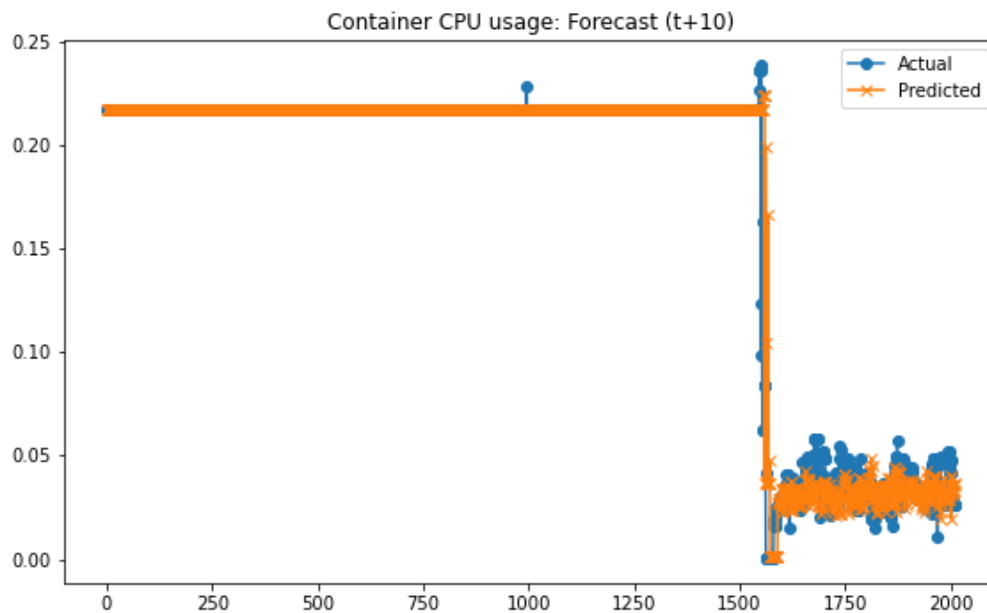


Figure 14: CPU consumption forecast example for one of the containers contained in the dataset (step $t=10$)

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 15.

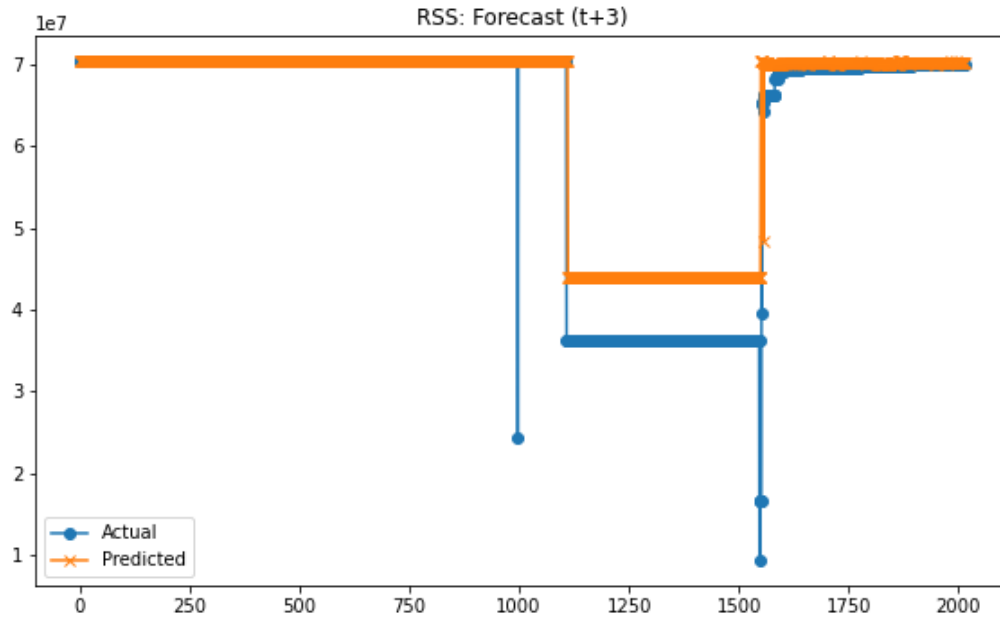


Figure 15: Memory Consumption (RSS) forecast example for one of the containers contained in the dataset (step $t=3$)

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).

6.5. Advancements beyond D3.3

The advancements following *D3.3 Net-Zero self-adaptive activation of distributed self-resilient augmented services* represent a strategic transition from open-dataset validation toward real-world deployment and custom infrastructure analysis. Moving beyond the utilization of existing open datasets, a proprietary dataset has been generated from our end-to-end, Docker-based 5G core and RAN simulation environment. The establishment of our custom 5G microservice infrastructure and the generation of the dataset serve as the foundational testbed for validating the proposed behavioral analysis framework. Building upon this robust, real-world data generation pipeline, the following Technical Description details the core components of the module. It outlines the specific infrastructure and dataset setup, elaborates on the AI-driven multi-stage algorithms employed for anomaly detection and resource forecasting, presents the evaluation results of the microservice profiling, and defines the automated mitigation actions designed to ensure continuous network resilience.

6.5.1. Technical Description

For this module we integrated a 5G infrastructure with a dataset generated from a Docker-based simulation, enabling precise behavioral profiling under both normal and adversarial conditions.

6.5.1.1. Infrastructure utilized

Our infrastructure enables full-stack 5G microservice profiling by incorporating a live 5G core (AMF, SMF, UPF, HSS, and PCRF) and radio access network simulators (UE, gNB), providing a high-fidelity environment for behavioral analysis. The custom dataset covers a continuous 65-minute execution window and captures twelve key performance indicators through reliable telemetry monitoring stack. This stack utilizes a specialized docker stats script and Prometheus to monitor infrastructure-level attributes, like including CPU utilization, memory usage, virtual memory size, disk I/O throughput, number of processes running per interval and network transmission/reception rates, alongside application-level metrics such as request throughput and response latency.

6.5.1.2. Attacks and Dataset

The generated proprietary dataset covers a continuous 65-minute execution window, carefully structured to capture both normal microservice behavior and a diverse range of attack scenarios. By systematically introducing different attack vectors, the dataset provides a rich basis for evaluating the module's ability to discern specific anomaly types.

The execution timeline is organized into distinct phases:

1. The dataset begins with reconnaissance sweeps originating from a malicious node (192.187.3.201) targeting specific internal network components (192.187.3.6 and

192.187.3.2). This phase tests the system's ability to detect early-stage network probing before significant resource exhaustion occurs.

2. A targeted brute force attack is launched against port 5909 on target 192.187.3.202, designed to create localized CPU and network spikes.
3. A 17-minute window of purely normal traffic is observed with no attack activity. This critical period allows the monitoring engine to establish behavioral models of normal microservice resource consumption.
4. A Slowloris attack targets 192.187.3.202:5909. This application-layer DoS vector is specifically included to test the module's capacity to detect attacks that hold connections open, typically resulting in anomalous thread counts or file descriptor usage rather than massive volumetric bandwidth spikes.
5. The final phase introduces a sequence of sophisticated, Scapy-crafted network attacks targeting 192.187.3.202:5909:
 - MAC Flooding: Designed to overwhelm the SDN switch architecture and force abnormal flow routing behaviors.
 - Heartbeat Attack: Simulates targeted data exfiltration/memory leaking vectors.
 - UDP Volumetric Attack: A high-bandwidth flood aimed at overwhelming the network RX/TX limits and CPU processing capacities of the microservices.

Overall, the structured progression of attack scenarios combined with periods of normal operation ensures a comprehensive and realistic evaluation environment for the proposed framework. This diverse dataset not only enables precise detection and classification of multiple anomaly types but also supports the validation of adaptive response strategies.

6.5.2. Mitigation actions

In response to specific malicious actions captured in the dataset, a multi-layered mitigation strategy can be enforced through the closed loop integration of the AI detection mechanism, the microservice orchestrator, and the central SDN controller. The following mapping details the targeted mitigation actions-reactions triggered for each identified attack phase:

Reconnaissance Phase:

- Attack: Network sweeping and probing from malicious IP 192.187.3.201 targeting internal IPs 192.187.3.6 and 192.187.3.2.
- Mitigation: Dynamic SDN ACLs. Upon detecting abnormal scanning patterns, the central SDN controller dynamically pushes flow rules to the edge switches to block or heavily rate-limit the offending source IP, preventing further topology discovery and internal mapping.

Brute Force:

- Attack: Targeted, high-frequency access attempts against specific service ports (192.187.3.202:5909).
- Mitigation: Orchestrator Scaling & SDN IP Isolation. The microservice orchestrator temporarily scales up the targeted authentication pods to handle the anomalous load without crashing. Simultaneously, the SDN controller isolates the malicious IP by dropping its specific flows, effectively halting the brute-force attempt while legitimate users remain unaffected. Additionally, we have implemented a traffic sink mechanism, which allows us to redirect malicious traffic, effectively absorbing and neutralizing the attack without impacting system performance.

Slowloris Attack:

- Attack: Application-layer exhaustion attack holding partial connections open to exhaust server thread pools at 192.187.3.202:5909.
- Mitigation: Service Timeout Adjustments and resource management. Because this attack generates low bandwidth but high connection counts, the SDN controller is bypassed in favor of orchestrator reaction defense. The orchestrator can adjust the service configurations to set aggressive connection timeouts, terminating the slow, malicious requests. Moreover, it is possible to give additional resources to maintain the overall service availability based on the attacked services.

Scapy Attacks:

- Attack: Multi-vector assault utilizing Scapy. It begins with malformed protocol requests simulating targeted data extraction and memory leaking (Heartbeat), before rapidly escalating into a massive arrival of UDP packets designed to completely overwhelm network bandwidth and the microservices' CPU processing queues.
- Mitigation: Anomalous Flow Dropping, Blackholing, and Traffic Rerouting. The AI detection mechanism first identifies the irregular frequencies characteristic of the Heartbeat exploits, instructing the SDN controller to insert high-priority rules that match and drop those specific malformed requests. As the attack transitions to a volumetric UDP flood, the SDN controller executes immediate edge-level blackholing of the malicious UDP flows. Concurrently, the microservice orchestrator and SDN controller work together to reroute legitimate traffic to backup microservice instances deployed in unaffected network segments, thereby protecting the internal 5G core network from being crippled.

The final results related to the mechanism will be presented in the upcoming deliverable D6.3.

7. Conclusion

The advancements detailed in this deliverable mark significant progress for the NATWORK project, moving from the foundational conceptual frameworks established in D3.3 toward implemented, improved, and experimentally tested security solutions. By integrating "Advancements beyond D3.3" in each technical section, this version of the document details the project's solutions to provide proactive, self-resilient, and sustainable security for the 6G continuum. Key achievements include:

- **Architectural Hardening and Proactive Defense:** The MTD framework has evolved to include "statelessification" and risk-based zoning strategies, effectively reducing the reach and predictable attack surface of stateful microservices.
- **Operational Intent-Based Security:** The translation of natural-language security intents into executable MMT-probe monitoring rules is now a formalized procedure, supported by empirical evidence of adversarial robustness in the underlying MAIP AI models.
- **Predictive Sustainability:** Adaptive orchestration has been enhanced with lightweight AI-driven workload prediction to anticipate resource demand and identify DoST signatures before they impact the network.
- **High-Utility Privacy:** The full realization of the SEAN-FL architectures (A1 and A2) successfully eliminates the traditional trade-off between privacy and utility in federated learning through distributed Differential Privacy and checksum-based integrity verification.
- **High-Fidelity Validation:** The extension of open-source datasets with a new proprietary one on a 5G core and RAN simulation environment ensures that the microservice profiling and anomaly detection mechanisms are validated against realistic, multi-vector threats such as Slowloris and volumetric Scapy-based attacks.

Collectively, these technical advancements establish the finalized design and implementation baseline for the security solutions detailed in this deliverable. These integrated assets are now prepared for experimental validation within the NATWORK testbed infrastructure, ensuring full alignment with the operational use cases defined in D6.1 and D6.2.

References

- [1] Porambage, P., Gür, G., Osorio, D. P. M., Liyanage, M., Gurtov, A., & Ylianttila, M. (2021). *The roadmap to 6G security and privacy*. IEEE Open Journal of the Communications Society, 2, 1094–1122. <https://doi.org/10.1109/OJCOMS.2021.3078081>
- [2] Benjaponpitak, T., Karakate, M., & Sripanidkulchai, K. (2020). Enabling live migration of containerized applications across clouds. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications* (pp. 2529–2538). IEEE. <https://doi.org/10.1109/INFOCOM41043.2020.9155403>
- [3] Morsli, R., Kara, N., Ould-Slimane, H., & Lahlou, L. (2025). Multidimensional intrusion detection system for containerized environments. In *2025 IEEE 11th International Conference on Network Softwarization (NetSoft)* (pp. 546–554). IEEE. <https://doi.org/10.1109/NetSoft64993.2025.11080585>
- [4] Soussi, W., Gür, G., & Stiller, B. (2024). Democratizing container live migration for enhanced future networks - A survey. *ACM Computing Surveys*, 57(4), 1–37. <https://doi.org/10.1145/3641823>
- [5] Soussi, W., Gür, G., & Stiller, B. (2024). Moving target defense (MTD) for 6G edge-to-cloud continuum: A cognitive perspective. *IEEE Network*
- [6] Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6), 26–38
- [7] National Institute of Standards and Technology. National Vulnerability Database. U.S. Department of Commerce, 2024, <https://nvd.nsit.org>. Accessed 16 Sept. 2025.
- [8] Forum of Incident Response and Security Teams (FIRST). Common Vulnerability Scoring System (CVSS). 2024, <https://www.first.org/cvss/>. Accessed 16 Sept. 2025.
- [9] Hayes, C. F., et al. (2021). A practical guide to multi-objective reinforcement learning and planning. *arXiv*. <https://arxiv.org/abs/2103.09568>
- [10] 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>
- [11] 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>
- [12] 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>
- [13] Official LIME Python library: <https://github.com/marcotcr/lime>
- [14] J. Turnbull, *Monitoring with Prometheus*. Turnbull Press, 2018.

- [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] Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q. S., & Poor, H. V. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15, 3454–3469. <https://doi.org/10.1109/TIFS.2020.2988575>
- [19] Zhang, Z., Ma, X., & Ma, J. (2023). Local Differential Privacy Based Membership-Privacy-Preserving Federated Learning for Deep-Learning-Driven Remote Sensing. *Remote Sensing*, 15(20), 5050. <https://doi.org/10.3390/rs15205050>
- [20] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 308–318). <https://doi.org/10.1145/2976749.2978318>
- [21] Hu, R., Guo, Y., & Gong, Y. (2021). Concentrated differentially private federated learning with performance analysis. *IEEE Open Journal of the Computer Society*, 2, 276–289. <https://doi.org/10.1109/OJCS.2021.3099108>
- [22] Ranaweera, K., Nguyen, D. C., Pathirana, P. N., Smith, D., Ding, M., Rakotoarivelo, T., & Seneviratne, A. (2025). Federated learning with differential privacy: An utility-enhanced approach. *arXiv*. <https://arxiv.org/abs/2503.21154>
- [23] Islamov, R., Horváth, S., Lucchi, A., Richtárik, P., & Gorbunov, E. (2025).[Text Wrapping Break]Double momentum and error feedback for clipping with fast rates and differential privacy. *arXiv*. <https://arxiv.org/abs/2502.11682>
- [24] Jin, Z., Xu, C., Wang, Z., & Sun, C. (2025). Toward robust differential privacy in adaptive federated learning architectures. *IEEE Transactions on Consumer Electronics*, 71, 4087–4099. <https://doi.org/10.1109/TCE.2024.3525084>
- [25] Ni, Z., & Zhou, Q. (2025). Differential privacy in federated learning: An evolutionary game analysis. *Applied Sciences*. <https://doi.org/10.3390/app15062914>
- [26] Lu, X., Liu, Z., Xiao, L., & Dai, H. (2025). Reinforcement learning-based personalized differentially private federated learning. *IEEE Transactions on Information Forensics and Security*, 20, 465–477. <https://doi.org/10.1109/TIFS.2024.3515814>

- [27] Kanagavelu, R., Wei, Q., Li, Z., Zhang, H., Samsudin, J., Yang, Y., Goh, R. S. M., & Wang, S. (2022). Ce-Fed: Communication efficient multi-party computation enabled federated learning. *Array*, 15, 100207. <https://doi.org/10.1016/j.array.2022.100207>
- [28] Kanagavelu, R., Li, Z., Samsudin, J., Yang, Y., Yang, F., Goh, R., Cheah, M., Wiwatphonthana, P., Akkarajitsakul, K., & Wang, S. (2020). Two-phase multi-party computation enabled privacy-preserving federated learning. In *2020 IEEE/ACM International Symposium on Cluster, Cloud, and Internet Computing (CCGRID)* (pp. 410–419). <https://doi.org/10.1109/CCGRID49817.2020.00052>
- [29] Fallatah, S., Grasso, F., & Lisitsa, A. (2024). MC-PPHFL: Privacy-preserving hierarchical federated learning with a secure multi-chain aggregation. In *2024 2nd International Conference on Federated Learning Technologies and Applications (FLTA)* (pp. 208–215). <https://doi.org/10.1109/FLTA63145.2024.10839609>
- [30] Nazemi, N., Tavallaie, O., Chen, S., Mandalari, A. M., Thilakarathna, K., Holz, R., Haddadi, H., & Zomaya, A. Y. (2025). Access-FL: Agile communication and computation for efficient secure aggregation in stable networks for FLaaS. In *2025 IEEE International Conference on Web Services (ICWS)* (pp. 1–7). <https://doi.org/10.1109/ICWS67624.2025.00036>
- [31] Chen, L., Xiao, D., Yu, Z., & Zhang, M. (2024). Secure and efficient federated learning via novel multi-party computation and compressed sensing. *Information Sciences*, 667, 120481. <https://doi.org/10.1016/j.ins.2024.120481>
- [32] Chen, L., Xiao, D., Xiao, X., & Zhang, Y. (2024). Secure and efficient federated learning via novel authenticable multi-party computation and compressed sensing. *IEEE Transactions on Information Forensics and Security*, 19, 10141–10156. <https://doi.org/10.1109/TIFS.2024.3486611>
- [33] Chen, R., Dong, Y., Liu, Y., Fan, T., Li, D., Guan, Z., Liu, J., & Zhou, J. (2025). FLOCK: Robust and privacy-preserving federated learning based on practical blockchain state channels. *Proceedings of the ACM Web Conference 2025*. <https://doi.org/10.1145/3696410.3714666>
- [34] Zheng, C., Wang, L., Xu, Z., & Li, H. (2024). Optimizing privacy in federated learning with MPC and differential privacy. In *Proceedings of the 3rd Asia Conference on Algorithms, Computing and Machine Learning (CACML '24)* (pp. 165–169). <https://doi.org/10.1145/3654823.3654854>
- [35] Byrd, D., & Polychroniadou, A. (2021). Differentially private secure multi-party computation for federated learning in financial applications. In *Proceedings of the First ACM International Conference on AI in Finance (ICAIF '20)*. <https://doi.org/10.1145/3383455.3422562>
- [36] Mugunthan, V., Byrd, D., Balch, T.H., Morgan, J.P., & Research, A. (2019). SMPAI: Secure Multi-Party Computation for Federated Learning. <https://api.semanticscholar.org/CorpusID:220598116>

- [37] Wei, W., Jammine, A., & Nait-Abdesselam, F. (2024). Enhancing privacy protection for federated learning with distributed differential privacy. In *2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (pp. 408–413). <https://doi.org/10.1109/WiMob61911.2024.10770430>
- [38] Mou, W., Fu, C., Lei, Y., & Hu, C. (2021). A verifiable federated learning scheme based on secure multi-party computation. In Z. Liu, F. Wu, & S. K. Das (Eds.), *Wireless Algorithms, Systems, and Applications* (pp. 198–209). Springer.
- [39] 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>
- [40] Abdullah, Y., Alshawki, M.B., Ligeti, P., Soussi W., & Stiller B., (2025). Byzantine-Resilient Federated Learning: Evaluating MPC Approaches IEEE ICDCSW.
- [41] Zuo, Y., Wu, Y., Min, G., Huang, C., & Pei, K. (2020). An intelligent anomaly detection scheme for micro-services architectures with temporal and spatial data analysis. *IEEE Transactions on Cognitive Communications and Networking*, 6(2), 548-561.
- [42] Nobre, J., Pires, E. J. S., & Reis, A. (2023). Anomaly detection in microservice-based systems. *Applied Sciences*, 13(13), Article 7891. <https://doi.org/10.3390/app13137891>
- [43] Ravichandiran, R., Bannazadeh, H., & Leon-Garcia, A. (2018). Anomaly detection using resource behaviour analysis for autoscaling systems. In *2018 4th IEEE conference on network softwarization and workshops (NetSoft)* (pp. 192-196). IEEE.
- [44] Desnoyers, M., & Dagenais, M. R. (2006). The lttng tracer: A low impact performance and behavior monitor for gnu/linux. In *OLS (Ottawa Linux Symposium)* (Vol. 2006, pp. 209-224).
- [45] Kohyarnejadfard, I., Aloise, D., Azhari, S. V., & Dagenais, M. R. (2022). Anomaly detection in microservice environments using distributed tracing data analysis and NLP. *Journal of Cloud Computing*, 11(1), 25.
- [46] NATWORK Consortium. (2025). *Payload security per runtime, intelligent runtime selection and attestation* (Deliverable D4.1).
- [47] Bugshan, N., Khalil, I., Kalapaaking, A. P., & Atiquzzaman, M. (2023). Intrusion detection-based ensemble learning and microservices for zero touch networks. *IEEE Communications Magazine*, 61(6), 86–92. <https://doi.org/10.1109/MCOM.001.2200535>
- [48] Passas, V., Makris, N., Wang, Y., Apostolaras, A., Mpatziakas, A., Drosou, A., Korakis, T., & Tzovaras, D. (2022). Artificial Intelligence for network function autoscaling in a cloud-native 5G network. *Computers and Electrical Engineering*, 103, Article 108327. <https://doi.org/10.1016/j.compeleceng.2022.108327>
- [49] Fernando, D., Rodriguez, M. A., & Buyya, R. (2024). iAnomaly: A toolkit for generating performance anomaly datasets in edge-cloud integrated computing environments. In *2024*

IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC) (pp. 236–245). IEEE. <https://doi.org/10.1109/UCC63386.2024.00041>

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 6.2.1.

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

Layer (Type)	Configuration / Hyperparameters	Activation	Output Shape
Input	Sequence length T, Channels C	—	(T, C)
Conv1D_1	16 filters, kernel size 5	ReLU	(T, 16)
BatchNorm	Axis defaults	—	(T, 16)
SepConv1D	32 filters, kernel size 3	ReLU	(T, 32)
MaxPool1D	Pool size 2	—	(T/2, 32)
Conv1D_2	64 filters, kernel size 3	ReLU	(T/2, 64)
GlobalAvgPool	Over temporal dimension	—	(64)
Dropout	Rate 0.25	—	(64)
Dense_1	32 units	ReLU	(32)
Dense_2 (Output)	2 units (classification head)	Softmax	(2)

Architecture for 1D-CNN for multiclass classification

Layer (Type)	Configuration / Hyperparameters	Activation	Output Shape
Input	Sequence length T, Channels C	—	(T, C)
Conv1D_1	32 filters, kernel size 5	ReLU	(T, 32)
BatchNorm	Axis defaults	—	(T, 32)
Conv1D_2	64 filters, kernel size 3	ReLU	(T, 64)
MaxPool1D	Pool size 2	—	(T/2, 64)
Conv1D_3	128 filters, kernel size 3	ReLU	(T/2, 128)
GlobalAvgPool	Over temporal dimension	—	(128)
Dropout	Rate 0.30	—	(128)
Dense_1	64 units	ReLU	(64)
Dense_2 (Output)	7 units (classification head)	Softmax	(7)

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)