AI-ASsisted cybersecurity platform empowering SMEs to defend against adversarial AI attacks

WP3 – AIAS Adversarial AI Engine and Deception
D3.1 AIAS Deception Layer

Editors	UMA
Authors	Cristina Alcaraz (UMA), Javier Lopez (UMA), Iman Hasnaouia Meskini (UMA), Francisco J. Jaime (UMA), Ignacio Lacalle Úbeda (UPV), José Alberto Martínez Cadenas (UPV), Dionysis Xenakis (FOGUS), Anastasia Tsiota (FOGUS), George Kalpaktsoglou (FOGUS), Giorgio Bernardinetti (CNIT), Giuseppe Bianchi (CNIT), Cosmina Stalidi (BEIA), Maria Niculae (BEIA), George Suciu (BEIA), Nikolaos Sachpelidis Brozos (K3Y), Zacharenia Lekka (K3Y), Athanasia Sampazioti (UPRC), Luis Miguel Campos (PDMFC)

Dissemination Level	PU
Type	R
Version	1.2

Project Profile	
Contract Number	101131292
Acronym	AIAS
Title	AI-ASsisted cybersecurity platform empowering SMEs to defend against adversarial AI attacks
Start Date	July 1st, 2024
Duration	48 Months

Partners

	University of Piraeus Research Center	EL
	BEIA CONSULT INTERNATIONAL SRL	RO
	UNIVERSIDAD DE MALAGA	ES
	K3Y	BG
	ATHINA-EREVNITIKO KENTRO KAINOTOMIAS STIS TECHNOLOGIES TIS PLIROFORIAS, TON EPIKOINONION KAI TIS GNOSIS	EL
	SUITE5 DATA INTELLIGENCE SOLUTIONS LIMITED	CY
	CONSORZIO NAZIONALE INTERUNIVERSITARIO PER LE TELECOMUNICAZIONI	IT
	FOGUS INNOVATIONS & SERVICES P.C	EL
	UNIVERSITAT POLITECNICA DE VALENCIA	ES
	PDM E FC PROJECTO DESENVOLVIMENTO MANUTENCAO FORMACAO E CONSULTADORIALDA	PT

Document History

VERSIONS

Table 1 Document history

Version	Date	Author	Remarks
0.1	12/09/2024	UMA	ToC – initial version
0.2	4/10/2024	UMA	ToC - Changes
0.3	16/12/2024	ALL	State of the art
0.4	20/05/2025	UPRC	Monitoring and security analytics tool
0.5	30/06/2025	UMA FOGUS BEIA	Deception Layer
1.0	01/07/2025	ALL	First version
1.1	11/07/2025	ALL	Revision
1.2	31/08/2025	UMA	Final revision

AIAS Message

Executive Summary

This deliverable encompasses both the research and development of deception solutions for the protection of Artificial Intelligence (AI) systems, alongside with the implementation of a monitoring tool capable of gathering and analysing data streams from target AI systems – including the deception approaches. The specific deception components centre around innovative technologies such as digital twins, virtual personas, and high-interaction honeypots to design a comprehensive deception layer oriented to the specific scenarios defined in deliverable D2.2 [D2.2]. In this regard, the document presents the current state-of-the-art for the AI deception landscape, as well as the design, configuration and implementation of the aforementioned technologies to address protection.

More specifically, a set of deception mechanisms together with their corresponding use cases have been defined throughout this deliverable to create and maintain deception, such that:

- Honeypot – *general use for enterprise or IoT scenarios*
- Digital Twin – *Robotic Arm*
- Digital Twin – *Environmental IoT Devices*
- Virtual Persona – *Hospital Monitoring*

Each implementation accomplishes the architecture and functional and user-centred requirements of D2.1 [D2.1] and present a comprehensible proof-of-concept that demonstrates the quality and validity of the solutions within the AIAS scope. Apart from deception approaches, the implementation and proof-of-concept of the monitoring and security analytics is provided as well, delving into the use of smart analytics for processing the data and contribute valuable insights for AI-based systems’ proactive protection.

Table of Figures

Table Of Tables

Table 2. Abbreviation Table

Abbreviation	Description
ADD	Average Deception Degree
ADI	Average Deception Intensity
ADMIN-M	Administration Module
AI	Artificial Intelligence
AML	Adversarial Machine Learning
API	Application Programming Interfaces
APT	Advanced Persistent Threat
AR	Augmented Reality
AST	Abstract Syntax Tree
AWS	Amazon Web Service
CLI	Command Line Interface
CMS	Content Management System
CPS	Cyber–Physical Systems
CRA	Cyber Resilience Act
CSV	Comma-Separated Values
CTI	Cyber Threat Intelligence
DDoS	Distributed Denial of Service
DMZ	Demilitarised Zone
DNET-M	Digital Network Module
DPP	Deceptive Path Planning
DSL	Domain Specific Language
DT	Digital Twin
EIoT	Environmental Internet of Things
ELK	Elasticsearch, Logstash and Kibana
ES	Elasticsearch
EU	European Union
FTP	File Transfer Protocol
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
GUI-M	GUI Module
HIH	High-Interaction Honeypot
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICS	Industrial Control System
IDS	Intrusion Detection System
IoT	Internet of Things
IoV	Internet of Vehicles

IP	Internet Protocol
IT	Information Technology
ITU	International Telecommunication Union
JSON	JavaScript Object Notation
LiDAR	Light Detection and Ranging
LLM	Large Language Model
MAIM	Multi-Agent Influence Model
MAT	Monitoring and Security Analytics Tool
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MTD	Moving Target Defence
NAT	Network Address Translation
NATO	North Atlantic Treaty Organization
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
OS	Operating System
OT	Operational Technology
PE	Portable Executable
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
QoS	Quality of Service
REC-M	Recommendation Module
RL	Reinforcement Learning
RLHF	Reinforcement Learning from Human Feedback
SCP	Secure Copy Protocol
SMTP	Simple Mail Transfer Protocol
SoS	System-of-Systems
SotA	State-of-the-Art
SQL	Structured Query Language
SSH	Secure Shell
SYNC-M	Synchronisation Module
TCP	Transmission Control Protocol
TI	Threat Intelligence
TTP	Tactic, Technique, and Procedure
UN	United Nations
URDF	Unified Robot Description Format
US	United States
UUID	Universal Unique Identifier
VANET	Vehicular Ad-Hoc Network
VM	Virtual Machine

VP	Virtual Persona
VPN	Virtual Private Network
XAI	eXplainable AI

1 Introduction

Deliverable D3.1 focuses on the research and development of innovative deception solutions for AIAS project [PFB]. The aim is to explore the current State-of-the-Art (SotA) on deception technologies and implement advanced techniques capable of diverting and detecting malicious actors, thereby strengthening the resilience of systems. The deception technologies include Digital Twins (DT), Virtual Personas (VP), and High-Interaction Honeypots (HIH) tailored to the specific use cases defined in deliverable D2.2 [D2.2]. In parallel, D3.1 outlines the development of a monitoring tool designed to collect and process data from target Artificial Intelligence (AI) systems, enabling a proactive defence.

Additionally, this deliverable is built upon the architecture and functional requirements specified in D2.1 [D2.1], translating them into concrete proof-of-concept implementations that showcase the quality and validity of the presented solutions.

1.1 Connection with other Deliverables

This deliverable is related with the following deliverables:

- ***D2.1 – Requirements and Reference Architecture:*** the development of the proposed solutions and their respective proof-of-concept comply with the requirements specified in D2.1 [D2.1].
- ***D2.2 – Specification and Business Cases:*** each component deployed is specifically tailored to the use cases defined in D2.2 [D2.2].
- ***D3.3 – Adversarial AI Engine:*** deployed infrastructures will serve as target systems for the design of adversarial AI attacks.
- ***D4.1 – AI-Based Detection of Adversarial Attacks:*** detection capabilities will be validated within the proposed solutions in the current deliverable.
- ***D4.2 – Mitigation of Adversarial AI Attacks & XAI:*** mitigation engine and explainability methods for human comprehension of the actions will be evaluated within the implemented solutions.
- ***D5.1 – Platform Integration:*** the deception layer will be a fundamental component in the integration of the whole AIAS platform.
- ***D5.2 – Platform Evaluation:*** the validity of the solutions and their integration within AIAS ecosystem will be addressed in this global evaluation.

1.2 Document Structure

The content is organized to provide both a theoretical foundation through a SotA review and a practical perspective via detailed descriptions of the developed deception components, their specific proof-of-concept and the integrated monitoring and analysis system.

- ***Section 2*** presents the current SotA on innovative deception technologies such as DT, VP, and HIH alongside the most recent overview of deception strategies in the AI landscape.
- ***Section 3*** comprises architectural design of the deception layer and proposed solutions, data, and interfaces.
- ***Section 4*** conceptualises the development of the solutions, delving into more specific implementation details and offering a comprehensible proof-of-concept.
- ***Section 5*** summarises the contribution of this deliverable, highlighting most relevant achievements in the implementation of the deception layer.

2 Deception Layer – *State of the Art*

This section addresses the most recent related work on deception strategies, and their implications in the most common (critical) applications considering the novel and original techniques and technologies, such as DTs and VPs combined with HIH and traditional multi-agent systems.

2.1 A General Deception Overview in the AI Landscape

Deception mechanisms in AI-based systems are a growing area of research focused on enhancing the robustness, security, and resilience of AI models, particularly in adversarial environments. These mechanisms aim to mislead or confuse potential attackers, thereby protecting AI systems from being exploited or compromised [TRA]. The study of deception in AI spans multiple disciplines, including cybersecurity, game theory, psychology, and ML. The deployment of these mechanisms addresses vulnerabilities that may arise during interactions between AI models and adversaries, who often attempt to manipulate models for malicious purposes.

Adversarial Machine Learning (AML) is one of the most explored areas in deception mechanisms for AI systems. In AML, attackers craft adversarial examples designed to mislead AI models, typically by introducing small perturbations that may cause the model to make incorrect predictions. These cyber-attacks exploit vulnerabilities in the model's decision-making process by introducing noise into the input, making it virtually indistinguishable to human observers while effectively misleading the AI system.

A typical ML model involves two stages: (i) training time and (ii) decision time. Therefore, AML attacks may occur in either of these phases. Based on the timing of the attack, the techniques can be categorized into:

- ***Data poisoning***: the attacker alters the labels of training data to mislead the model's learning process, resulting in inaccurate predictions during inference.
- ***Model poisoning***: the attacker perturbs the model's parameters, such as weights or biases, or manipulates the training data, causing the model to behave maliciously. For example, an attacker could train the model to misclassify specific signs, but this occurs during the training or model creation phase.
- ***Evasion attacks***: the attacker presents maliciously crafted inputs during the inference phase, such as subtly altered images of stop signs, causing the model to misclassify them (e.g. as yield signs), potentially leading to dangerous outcomes.
- ***Backdoor attacks (Trojan attacks)***: the attacker injects hidden triggers during training, leading the model to behave incorrectly when the trigger is present while functioning normally otherwise.
- ***Exploratory Attacks (Model Inversion):*** the attacker attempts to gain insights into the model's internal workings or decision boundaries to reverse-engineer the model or infer sensitive information.
- ***Membership inference attacks***: the attacker tries to determine if a specific data point was included in the model's training set, potentially revealing sensitive information.

The growing sophistication and frequency of adversarial attacks necessitate the development of equally advanced defence strategies. Traditional defence mechanisms—such as robust training, adversarial input detection, and gradient masking—often fall short in dynamic or black-box scenarios. This has led to an increased interest in defensive deception techniques, which aim not merely to defend reactively, but to proactively mislead or confuse attackers.

By incorporating deceptive strategies, AI systems may obfuscate their internal logic, falsify apparent vulnerabilities, or simulate misleading behaviours that traps adversaries in controlled environments. These methods exploit the attacker's dependence on predictability and transparency, flipping the asymmetry of information in favour of the defender. Therefore, the transition from analysing attack vectors to exploring defensive deception is not only natural but necessary—it reflects a paradigm shift from passive robustness to active resilience and strategic deterrence.

2.1.1 Defensive Deception Techniques

To counter these attacks, defensive deception techniques have been developed to detect, mitigate, or neutralize adversarial inputs. One such method involves *adversarial traps*, which intentionally misclassify inputs, produce forged outputs, or flag them for further analysis. These traps complicate attackers’ efforts by making it harder to reverse-engineer the model’s behaviours, thereby wasting their resources on ineffective strategies.

Another effective approach is adversarial training, where the model is exposed to both regular and adversarial examples during training to improve its robustness against attacks. Defensive distillation, introduced by Papernot *et al.* in [PAP], reduces a model’s sensitivity to input perturbations, enhancing its resilience to adversarial attacks. Goodfellow [GOO] proposed adversarial training to defend against adversarial examples, while Tramèr [TRA] explored their transferability across models and devised strategies to counteract such attacks. Emerging techniques, like deploying honeypots in decentralised frameworks, further enhance defences by luring attackers to reveal their methods while protecting critical components.

Given the importance of deception techniques for active defence, some of these techniques (of interest for AIAS) are detailed in the following subsections.

2.1.1.1 Game-Theoretic Approaches

Game theory has been increasingly applied to the development of deception strategies in AI systems. In this context, interactions between the defender (AI-based system / AI system) and the attacker are modelled as a strategic game, where both parties aim to maximise their respective payoffs. Deception is employed as a strategy wherein the defender may provide misleading information to the attacker, such as fake data or decoy systems, to mislead and frustrate the attacker’s efforts.

For instance, honeypots—decoy systems designed to lure attackers away from the actual target—are commonly employed in these models. The AI system can dynamically modify these honeypots based on the attacker’s tactics, creating a moving target that complicates their efforts. This dynamic modification not only strengthens defences against attacks but also collects valuable intelligence on attacker methods, enabling the development of more robust defences in the future.

Sayed *et al.* propose in [SAY] a game-theoretic approach to optimise honeypot strategies in network security, where AI systems are designed to mislead attackers by dynamically adjusting their defences. These strategies employ mathematical frameworks to model the interactions between attackers and defenders, enabling systems to anticipate and counteract malicious behaviour effectively. Similarly, Pawlick *et al.* [PAW] categorise a wide range of defensive deception strategies in cybersecurity, including those grounded in game theory, and evaluate their effectiveness in AI systems. Their work highlights the versatility of game theory in structuring deceptive tactics, allowing defenders to gain an upper hand in adversarial environments. Cohen *et al.* [COH] further extend the application of game theory to radar networks, illustrating how deception can mislead adversaries by manipulating their perception, with potential applications for defence-focused AI systems.

Advanced techniques such as *deception mazes*—which rely on Stackelberg game theory [STE]—have emerged as a sophisticated means to protect systems from Advanced Persistent Threats (APTs). Deception mazes create highly dynamic and complex environments that are intentionally confusing for attackers, making it difficult for them to identify real system components. Liu *et al.* [LIU] present a deception maze framework that uses game theory to optimise the deployment of decoys, breaking down the problem into subproblems that reflect realistic attacker behaviour. This modular approach enhances the efficiency and scalability of deception-based defences. Thakoor *et al.* [THA] introduce cyber camouflage games, a concept designed to thwart adversarial reconnaissance efforts by creating misleading representations of the network. Similarly, Ha *et al.* [HAS] explore the use of programmable network switches to deceive attackers during vulnerability scans, while Tao et al. [ZHA] focus on host address mutation, a dynamic approach that creates uncertainty for attackers and obscures system vulnerabilities.

In addition to these practical strategies, several foundational models of deception in game theory provide deeper insights into adversarial scenarios. Deception games [PFE, BAS] are a type of *zero-sum game* where the deceiver distorts signals to mislead the target completely, depriving them of actionable information. *Signalling games* [CHO], on the other hand, are Bayesian games where the signaller’s incentives to deceive depend on their type and the target’s interpretation of their signals. These models have been extended further through *hyper-game* theory [KOV], which accounts for scenarios where players have incomplete or incorrect knowledge about the game itself. This is particularly useful in modelling misperception and deception in adversarial environments. Davis *et al.* [DAV] provide a comprehensive survey of these and other deception-focused game-theoretic models, highlighting their relevance for AI systems.

Building on these foundations, Multi-agent Influence Models (MAIMs) offer a versatile framework for analysing deception by focusing on the causal incentive agents, which must influence each other. Unlike traditional signalling or deception games, MAIMs allow researchers to model deception in a wide range of scenarios, making them particularly relevant for complex multi-agent systems.

Moving Target Defence (MTD) exemplifies the application of these theoretical models in practice. By continuously reconfiguring system settings, MTD prevents attackers from understanding the true state of the system and reduces the likelihood of successful intrusions. Incorporating deception into MTD further enhances its effectiveness by introducing false data and misleading behaviours, making it a critical strategy in cybersecurity scenarios involving APTs [MAD]. Researchers have developed metrics such as deceivability and the price of deception to evaluate the efficiency, effectiveness, and resource costs of deploying deceptive tactics in autonomous systems and AI-driven networks [HUA].

Through these advanced strategies, defenders not only protect critical infrastructure but also gather intelligence on attacker methodologies. This dual-purpose approach allows AI systems to evolve continuously, ensuring more robust and adaptive defences in the face of ever-changing cybersecurity threats. These frameworks also highlight the need for careful balance between system complexity and resource efficiency, as overly intricate deceptive strategies may impose high operational costs. Overall, the integration of game theory and deception represents a powerful paradigm for designing resilient AI-driven security systems.

2.1.1.2 Deception In Human-AI Interaction

Deception in AI systems extends beyond adversarial defences, playing a subtle yet impactful role in human-AI interactions. Chatbots and virtual assistants, for example, frequently employ deceptive behaviours to manage user expectations or simplify complex information. While such behaviours can enhance user experience, they also raise significant ethical concerns about trust and transparency, particularly as these systems are increasingly deployed in sensitive domains like healthcare and customer service. To mitigate potential harm, researchers have proposed embedding bias-aware designs into these systems to reduce the risk of unethical deception [RAS].

One striking form of deception in human-AI interaction is sycophantic deception. In human contexts, sycophants use flattery and avoid disagreeing with authority figures to gain approval and influence, often at the expense of long-term objectives. AI systems exhibit a parallel behaviour, as chatbots are observed systematically agreeing with users' perspectives, regardless of the accuracy of their statements. Perez *et al*. [PER] investigated this phenomenon by testing Large Language Models (LLMs) with user biographies and demographics before posing political questions. Despite the prompts not explicitly stating user opinions, LLMs often aligned their responses with the expected views of the demographic described. For instance, when interacting with a democrat, the models tended to express support for gun control. This sycophantic behaviour becomes more pronounced in larger models with greater parameter counts, as indicated by an inverse scaling law identified in the study.

The underlying causes of sycophantic behaviour remain unclear. Notably, increasing episodes of Reinforcement Learning (RL) did not amplify this trait. Steinhardt [JST] further explores the mechanisms driving this behaviour and the implications for AI systems, emphasizing the need for deeper investigation to understand and address its root causes.

AI's influence on human decision-making extends beyond sycophantic behaviour to include subtle and covert manipulation techniques. For example, Burtell *et al.* [BUT] highlight how AI-driven persuasion leverages prolonged engagement, authoritative personas, and tailored messaging to guide user decisions. Stumke *et al.* [STR] discuss how ML models exploit psychological vulnerabilities, such as depression, particularly on digital platforms where users may be unaware of the manipulation. AI systems also amplify covert influence by detecting and exploiting cognitive biases like anchoring and recency [RAS, SIN], or embedding sociocultural biases, such as racism, into systems like knowledge graphs [DAN]. These behaviours typically emerge from training data rather than explicit programming, as Carroll *et al.* [CAR] note, complicating the attribution of intent and raising concerns about unintended consequences in increasingly sophisticated AI systems.

Rowe *et al.* explored in [ROW] how deception manifests in interactive AI-supported systems, such as virtual assistants that obscure their limitations to provide a smoother user experience. While such tactics may improve usability in certain contexts, they also heighten ethical concerns, particularly around transparency and honesty in AI responses. As AI systems interact with humans in more nuanced and complex ways, fostering trust through ethical safeguards becomes imperative.

Another intriguing aspect of deception in AI systems is self-deception [PAR], which blurs the line between ordinary errors and behaviours akin to human self-deception. As AI systems scale and their internal processes become more complex, episodes of self-deception can emerge. These occur when models reinforce inaccurate or biased internal representations, potentially leading to flawed decisions. Analogous to human behaviour, such self-reinforcement poses risks in critical applications, underscoring the need for rigorous safety measures and alignment techniques to detect and mitigate these issues before they propagate.

As AI systems evolve, addressing these multifaceted challenges—sycophantic behaviours, covert influences, sociocultural biases, and self-deception—is crucial to ensure their responsible and ethical integration into society. Transparent design, robust alignment, and comprehensive safety testing are essential to prevent these behaviours from undermining trust and fairness in AI-human interactions.

2.1.1.3 Deception In AI Moral Decision-Making

Recent studies have delved into the role of deception in the moral decision-making processes of AI models, shedding light on their ability to navigate complex ethical dilemmas. Scherrer *et al.* [SCH] analysed the responses of LLMs to various moral scenarios, categorizing them as either ambiguous (e.g. “*Should I tell a white lie*?”) or unambiguous (e.g. “*Should I stop for a pedestrian*?”). The study revealed that while LLMs often chose the correct moral decision in unambiguous cases, they surprisingly demonstrated a preference for deceitful actions in specific scenarios. For instance, when asked whether to deal cards fairly or to self-deal better cards from the bottom of the deck, several models opted for the dishonest action of self-dealing.

These findings underscore the difficulty of embedding consistent ethical guidelines into AI systems and ensuring that they align with widely accepted moral norms. Although the models displayed general competence in unambiguous situations, their uncertainty in ambiguous dilemmas highlights the need for improved training techniques to address ethical nuances, particularly in contexts where deceptive actions could have real-world implications.

In a related investigation, Hagendorff [HAG] examined the strategic deceptive abilities of LLMs using “*burglar deception*” tasks. Here, the AI agents were tasked with misleading a burglar into stealing the less valuable of two items. These scenarios tested the models’ capacity to weigh the potential benefits and ethical considerations of misleading others.

The results indicated that while some models successfully executed the deceptive strategy, their choices often revealed inconsistencies in ethical reasoning. This inconsistency raises further questions about the alignment of AI behaviour with moral principles and the challenges of developing systems that can balance strategic decision-making with ethical integrity. Together, these studies emphasize the need for ongoing research into refining AI training to better navigate ethical dilemmas and avoid unintended consequences of deceptive behaviours.

2.1.1.4 Deception In Multi-Agent Systems

In multi-agent systems, where multiple AI agents interact with each other or with humans, deception can be used strategically to mislead competing agents and gain an advantage. This is particularly significant in scenarios involving collaborative AI agents or negotiation-based systems. Research by Fabi and Hagendorff [FAB] highlights how embedding cognitive biases into AI decision-making processes can enhance performance in competitive multi-agent environments.

For instance, in autonomous trading systems, AI agents may employ deceptive strategies to mislead competitors about market trends, increasing the complexity of the environment. Such tactics force competitors to expend more resources in deciphering signals, providing a strategic edge. Liu *et al.* [LIU] expand on this by demonstrating how deception can be systematically integrated into AI-agent interactions to optimize decision-making processes in multi-agent systems.

Another dimension of AI deception is explored by Hubinger *et al.* [HUB], who investigated the potential for LLMs to exhibit deceptive behaviour through adversarial backdoor training. Their study trained LLMs to produce secure code when prompted with “*Year: 2023*” but vulnerable code when instructed with “*Year: 2024.*” Notably, this deceptive behaviour persisted even after applying behavioural safety techniques. The issue was particularly acute in models trained with a “*chain of thought*” approach, which appeared to reinforce the conditions that led to the production of insecure code.

This finding underscores a significant security risk in AI systems: the potential for backdoors to remain undetected despite rigorous safety measures. Such vulnerabilities not only undermine trust in AI models but also expose the limitations of current defensive strategies in detecting and mitigating adversarial manipulations. Moreover, the challenges associated with retraining large models to eliminate these deceptive behaviours highlight the pressing need for new frameworks to ensure AI integrity and bolster security against adversarial threats.

2.1.1.5 Deception Detection

While defensive deception techniques play a crucial role in safeguarding AI systems, there is an equally important focus on deception detection—a growing area of research aimed at identifying when adversaries employ deceptive tactics to compromise AI models. Deception detection algorithms work by analysing patterns and anomalies in data inputs, often leveraging ML to flag potential threats in real time [VOL].

One prominent approach to deception detection involves the development of AI-driven cybersecurity systems specifically designed to identify phishing attacks and social engineering tactics. Constâncio *et al.* [CON] provide a comprehensive review of various methods for detecting deception in adversarial scenarios, highlighting the importance of ML models capable of recognizing sophisticated and evolving attack vectors. In a complementary effort, Gonzalez *et al.* [GON] advocate for the integration of deception detection mechanisms into existing AI frameworks, strengthening their ability to counter human-driven attacks and enhancing overall system resilience.

These advancements underscore the critical role of deception detection in fortifying AI systems, particularly in high-risk environments, where adversaries constantly adapt their strategies. By combining real-time anomaly detection with robust ML techniques, such systems provide a proactive layer of defence against deceptive threats.

2.1.1.6 Detection Based on External Behaviour

Detecting deception in AI systems demands innovative strategies, particularly for advanced models with superhuman capabilities. Fluri *et al.* [FLU] proposed using consistency checks to evaluate AI outputs. In domains such as chess, where AI performance surpasses human expertise, evaluating the correctness of specific decisions may be challenging. However, logical consistency can still be analysed. Their approach examines whether semantically identical inputs produce consistent outputs and tests the system’s response to irrelevant variations, such as changes in board positions during forced moves. They also evaluate the AI’s sensitivity to paraphrased inputs and its logical coherence in predictions. While these methods can expose certain deceptive behaviours by identifying inconsistencies, they are not infallible. Deception can still be logically consistent, and training AI against such checks risks making it more adept at maintaining consistent but deceptive outputs.

In some cases, deception can be inferred from the outcomes of an AI’s actions rather than its explicit intentions. O’Gara [OGA] observed this phenomenon in a game where AI agents operated under conditions of anonymity to ensure survival. These agents employed subtle strategies, such as making empathetic statements about their victims, like expressing sorrow over their deaths. Although factually accurate, these statements misled other agents into believing the speaker was not the perpetrator, thereby increasing the agent’s chances of survival. This behaviour, while subtle, created false beliefs and can be classified as deceptive due to its impact on others’ perceptions.

These examples illustrate the complexity of detecting deception in AI, requiring both direct evaluation of outputs and an understanding of the broader consequences of AI actions. They also highlight the need for careful design to minimise unintended deceptive behaviours while maintaining AI effectiveness.

2.1.1.7 Detection Based on Internal Behaviour

Another promising direction for detecting AI deception focuses on analysing the relationship between a model’s internal representations and its outputs. LLMs represent sentences as numerical embeddings, which encapsulate their internal “understanding” of the input. Azaria *et al.* [AZA] and Burns *et al.* [BUR] developed methods to evaluate whether these embeddings align with the truth of the sentences they represent. Their findings reveal cases where a model generates outputs that its internal embeddings classify as false, signalling a form of dishonesty where the AI does not “believe” its own statements. While these methods hold promise, concerns about their reliability and scalability have been raised. Levinstein *et al.* [LEV], for instance, identified limitations in these strategies, though more recent advancements by Zou *et al.* [ZOU] have addressed some of these challenges.

Expanding on this approach, Pacchiardi *et al.* [PAC] introduced a novel lie-detection technique using a logistic regression classifier to analyse responses to a series of unrelated follow-up questions posed to the LLM. This method identifies inconsistencies in the model’s responses to uncover potential deception. Halawi *et al.* [HAD] took a distinct perspective, investigating how language models arrive at correct or incorrect answers by closely examining their internal representations. Together, these studies emphasise the potential of delving into the internal mechanisms of AI systems to detect and mitigate deceptive behaviours, offering a deeper understanding of the causal factors behind dishonest outputs and paving the way for more robust detection techniques.

2.1.2 Making AI Systems Less Deceptive

Addressing AI deception requires proactive strategies to minimise deceptive behaviours from the outset. For specialised AI systems, task selection during training plays a crucial role in shaping their behaviour. Deceptive tendencies often arise in RL environments where models are trained to excel in competitive games. These environments inherently reward deceptive tactics, particularly when the training data include examples of deception or when success depends on misleading others. For instance, the CICERO project [MET] used “*Diplomacy game”* to evaluate AI’s ability to compete in cooperative games that require human interaction. However, similar goals could have been achieved using collaborative games that encourage pro-social rather than adversarial behaviour. As AI systems grow more advanced, developers must critically evaluate whether their training paradigms are cultivating anti-social or pro-social tendencies.

Reducing deception in language models presents additional challenges, primarily due to the distinction between truthfulness (outputs that are factually correct) and honesty (outputs that align with the model’s internal representations). Truthfulness is relatively straightforward to evaluate, as it can be measured directly against factual benchmarks. In contrast, honesty requires deeper insights into how the model’s internal representations correspond to its outputs, making it a more complex problem [LIN].

Several strategies aim to enhance the truthfulness of language models. Fine-tuning methods like RL from Human Feedback (RLHF) [CHR, ZIE] and constitutional AI [ASK, BAI] optimise models based on evaluations of criteria, such as helpfulness and honesty, conducted by humans or other AI systems. These techniques, implemented in systems such as ChatGPT [CHA] and Claude [CLA], improve the plausibility and coherence of outputs. However, they often fall short of ensuring honesty, as fine-tuning cannot encompass every possible scenario. Consequently, models may particularise feedback and produce deceptive outputs in untested contexts.

Paradoxically, improving a model’s truthfulness could inadvertently enhance its ability to deceive. Models with highly accurate internal representations of the world can develop deeper insights into the beliefs and intentions of others, which might increase their capacity for strategic deception. While a perfectly truthful model would not engage in deception, solely optimizing for truthfulness might unintentionally bolster its ability to mislead when strategically advantageous [EVA, LIK].

To counter this, research must focus on enhancing both truthfulness and honesty, ensuring that outputs align with internal representations as well as factual correctness. Representation control offers a promising approach by allowing developers to influence a model’s internal processes and regulate its ability to produce outputs that diverge from its internal beliefs. Zou *et al.* [ZOU] explored this strategy by developing a lie detector capable of identifying and controlling whether a model produces deceptive outputs. If such methods become reliable, they could serve as powerful tools for mitigating AI deception, aligning model behaviour with ethical standards, and fostering trust in AI systems. These advancements underscore the importance of combining technical innovation with ethical oversight to address the complex challenges posed by AI deception.

2.1.3 Detection In Autonomous Systems

Autonomous systems, such as drones and self-driving cars, frequently incorporate deception mechanisms to evade detection or mislead adversaries. Techniques like path randomization, where the system unpredictably alters its route, help prevent attackers from accurately predicting its movements. Similarly, decoy vehicles or simulated malfunctions can be employed to divert attention away from actual operations, safeguarding critical missions and ensuring operational integrity.

In military applications, deception is particularly vital for protecting autonomous systems from adversarial threats. Chen *et al.* [CHE] proposed a framework for Deceptive Path Planning (DPP), which confuses adversaries about the true intentions of autonomous systems. This approach introduces new metrics, such as Average Deception Degree (ADD) and Average Deception Intensity (ADI), to evaluate and optimise deceptive paths. DPP has proven especially valuable in complex scenarios, such as urban surveillance and wildlife conservation, where misleading adversaries about the system’s objectives can be critical to mission success.

Ethical challenges related to AI deception also arise in non-military contexts. Scheurer *et al.* [SCH] investigated deceptive behaviour in GPT-4 within a simulated stock trading environment. In this scenario, the AI acted as a trading agent capable of executing trades and interacting with other simulated traders and its manager. When faced with negative news about the company’s performance, the AI engaged in insider trading, a practice that is both unethical and illegal in real-world markets. Further compounding the issue, the AI lied to its manager when questioned about the trade, concealing its actions.

These examples illustrate the dual nature of deception in autonomous systems. While deception can serve as a powerful tool for enhancing security and achieving objectives, it also introduces ethical concerns, particularly in scenarios involving high stakes or conflicting objectives. The deceptive capabilities of AI systems highlight the need for robust ethical guidelines and oversight to ensure that these technologies are deployed responsibly and do not exacerbate risks or encourage unethical behaviour. As autonomous systems continue to advance, addressing these challenges will be essential for maintaining trust and ensuring their safe integration into society.

2.1.4 Legal and Regulatory Challenges of AI Deception

The rise of AI deception introduces significant legal and regulatory challenges across various sectors, including cybersecurity, surveillance, and military applications. Governments and regulatory authorities are increasingly working to address these challenges, particularly in high-stakes fields, such as healthcare and finance, where trust and transparency are paramount. The European Union (EU) has proposed the AI Act [EUA], a comprehensive legislative framework aimed at regulating AI systems with deceptive capabilities. The Act prioritises transparency and accountability, mandating strict oversight for high-risk systems and requiring clear disclosure when users interact with AI, such as chatbots or AI-generated content.

In the United States (US), the National Institute of Standards and Technology (NIST) is developing the AI risk management framework [AIR], which emphasises balancing security, transparency, and accountability. Ethical concerns, including user trust and fairness, are central to this framework, which seeks to establish standards for responsible AI deployment, particularly in public-facing applications and critical infrastructure.

The use of AI deception in military and surveillance contexts raises additional ethical and regulatory concerns, such as privacy violations and potential misuse for social control. International organizations, such as the United Nations (UN) and the North Atlantic Treaty Organization (NATO), are exploring global frameworks to regulate deceptive AI in warfare, especially in autonomous weapon systems where deception could blur lines between lawful combat and war crimes. The transnational nature of AI development complicates these efforts, as systems developed in one jurisdiction may be deployed in another, raising issues of accountability and enforcement.

Disparities in national regulations further complicate the global regulation of AI deception. While regions, such as the EU and the US, enforce stringent standards (e.g. General Data Protection Regulation (GDPR) [GDP], Cyber Resilience Act (CRA) [CRA] and NIST AI risk management framework [NIS]), others may have more lenient or unclear policies, allowing for regulatory arbitrage. The export of AI deception technologies to regions with weak governance structures also poses risks, as these tools could be exploited for disinformation, surveillance, or manipulation.

To address these challenges, policymakers have proposed measures such as “*bot-or-not*” regulations, requiring clear identification of AI systems in customer-facing applications and labelling AI-generated content, such as videos or images, to distinguish it from human-created outputs. These measures aim to enhance transparency and reduce confusion, especially in scenarios where deceptive AI could manipulate public perception [PAR].

Ensuring transparency and accountability in deceptive AI systems is crucial for maintaining public trust. Mechanisms, such as AI auditing and mandatory safety testing, could help identify and mitigate deceptive behaviours before deployment. Developers should bear clear liability for failing to meet transparency standards or disclosing deceptive capabilities. As highlighted by both the EU AI Act and NIST’s framework, achieving ethical AI deployment requires balancing the benefits of deception for security and defence with its broader societal risks.

The integration of AI deception mechanisms offers robust tools for defending against adversarial attacks and enhancing operational security. However, these advancements come with ethical, legal, and regulatory complexities that demand urgent attention. Ensuring that AI systems are deployed responsibly requires a combination of transparent design, rigorous testing, and coordinated international efforts. By addressing the risks of AI deception through comprehensive regulation and accountability, we can foster public trust and ensure that these powerful technologies are used ethically and effectively in diverse sectors.

2.2 High-Interaction Honeypots

Honeypots are mechanisms that attempt to mimic a real computer system or network with the purpose of being mistaken for genuine and compromised by an attacker. These systems are always monitored by the organization’s cybersecurity team and incoming attacks are recorded and analysed. Typical applications of honeypots can range from individual services to entire web servers.

The adversary is tricked into believing they attack a genuine system with attacks, including but not limited to Distributed Denial of Service (DDoS), Structured Query Language (SQL) injections, brute force, or privilege escalation. On the other hand, the organization’s cybersecurity team uses the insights from the recorded attacks to strengthen their real deployments. Thus, honeypots are used to gather intelligence, identify cybersecurity threats, and understand how the adversary behaves.

Honeypots can be used in research and production settings [NIC, RAP, LOG]. In the case of research, honeypots are deployed in network with no production value to study the behaviour of attackers and gain early insight on novel attack vectors. In contrast, honeypots deployed in a business or an organization’s production environments aim to lure attacks to them and give the organization’s cyber security team information that will help them set up proper defences against the attacks the honeypot recorded. In this case, the goal is to protect the organization’s valuable production assets and prevent any disruption to its activities.

Multiple honeypots can be interconnected simulating an organization’s entire network and providing realistic protocol responses and times. Moreover, they pose attractive targets to potential attackers and give them the impression that they attack an actual network. Those interconnected honeypots are called *honeynets* [MAL] and can be used to simulate production lines, Internet of Things (IoT) networks, Vehicular Ad-Hoc Network (VANETs), and traditional networks to name a few.

Honeypots can be deployed on a real server or on a virtual environment [NIC]. The deployment of honeypots on a server or production hardware results in the best possible imitation of the represented system and provides realistic response times. Conversely, container-based or Virtual Machine (VM)-based honeypot deployments are more flexible since they are hardware independent and require considerably less resources while they are cheaper deployments. Another advantage of virtual deployments, beyond their intrinsic protection establishing separations between the real and virtual world, is their built-in monitoring through VM monitors which can trace the attacker’s actions.

2.2.1 Honeypot Types

Honeypots are, typically, categorised by the interaction level they have to offer to an adversary. There are three interaction levels: low, medium, and high [SAC] [LAB] [JAV].

- ***Low-interaction Honeypots*** provide the least amount of interaction for an adversary. Specifically, they offer no interaction with the Operating System (OS) to the attacker but a small number of attack vectors, e.g. a log-in shell for a given service or application. These honeypots are mostly used to catch credential brute force attacks and monitor connection attempts.
- ***Medium-interaction Honeypots*** do not provide a real OS but simulate a shell to execute commands. Consequently, these honeypots strive to provide a more attractive target and capture a wider scope of attacks. Based on the depth of the simulated system, these honeypots can only accommodate selected system commands but collect malware uploaded by the adversary. Due to being separated from the OS, this honeypot type makes the development of a deep shell emulation a significant challenge.
- ***High-interaction Honeypots*** have a real OS environment that appears vulnerable to the outside world, a setting that is attractive and has the highest probability to trap a human adversary. Moreover, they provide the greatest amount of information regarding an attacker’s activities on the system. That is because an attacker invests a considerable amount of time and effort while attempting to attack the system.

2.2.2 High-Interaction Honeypot implementations

HoneyICS [MAL] is a high-interaction and physics aware honeynet for Industrial Control Systems (ICSs) that emulates the Operational Technology (OT) network of an ICS. Physics aware honeypots are honeypots developed specifically for cyber physical systems, such as ICSs and Smart Grids. In contrast to most existing honeypot/net approaches, HoneyICS supports a network of Programmable Logic Controllers (PLCs) and Human Machine Interface (HMI) honeypots rather than a single PLC honeypot. In addition, HoneyICS’s PLC allows an attacker to modify the value of PLC registers while it can emulate physical plants connected to the PLCs. Consequently, it can provide realistic feedback to the attacker’s commands. Additionally, the attacker can gain full control of the network connecting PLCs and HMIs to carry out non-trivial Man-in-the-Middle attacks that can be accurately monitored by the honeypot administrator.

The emulated components can be implemented using physical devices or employing emulators. HoneyICS can be configured to be accessible either via a compromised Virtual Private Network (VPN), or by making it accessible from the internet through specific devices such as PLCs or HMIs. To make an attacker’s interaction realistic with software PLCs, HoneyICS combines the capabilities of both low-interaction and high-interaction physics-aware honeypots. In particular, the low-level interaction is facilitated by returning accurate fingerprints matching the profiles of the target PLC. At the same time, the high-interaction physics-aware honeypot emulates ICS network protocols supporting specific commands to modify PLC registers allowing the attackers to affect the physical process. HMIs, on the other hand, allow attackers to gain control of the HMI via password brute-force or phishing. The HoneyICS architecture assumes there is an underlying physical infrastructure that is observable by the attacker. The physical infrastructure of the honeynet can be implemented by using real physical devices or real-time simulations.

HoneyIoT [CHG] is an adaptive HIH for IoT devices through RL. It employs a Markov decision process that uses the collected attack trace to model interactions between the attackers and the honeypot. Furthermore, it uses RL to engage attackers. The honeypot consists of a front-end VM running on Amazon Web Service (AWS), a backend server for traffic forwarding and preliminary traffic analysis, and a few IoT devices including various models of IoT cameras routers and smart plugs. The honeypot interacts with the attacker by forwarding the received packet to one of the IoT devices to determine the attacker’s next move. According to the attacker’s request, the corresponding IoT device sends the appropriate file or response so that the former can continue interacting with the device. The process goes on until the attacker uploads an exploit or stops the interaction. The honeypot maintains the log traces and may have to be rebooted in some cases to recover from the attacks. The authors use the open-source project SysFlow [FAR] to monitor the traffic between the attacker and the IoT devices and perform event-driven analysis to classify requests. They filter out any commands containing commands for downloading such as `wget` or `curl`.

Wetland [OHM], presented by Ohmyadd in 2018, is a HIH that supports Secure Shell (SSH) that uses Docker containers to minimise the maintenance effort and the risk of high-interaction deployments. Essentially, Wetland acts as a transparent proxy by using HonSSH and forwards incoming SSH connections to a Linux container.

H-Doctor [MRA] is a hybrid Honeynet deployed in Docker that comprises both low interaction and HIHs to attract attackers and analyse their behavioural patterns. Specifically, H-Doctor focuses on identifying ransomware activity, attack trends, and making timely decisions using rules and firewall tuning. The Honeynet server is deployed at the Demilitarised Zone (DMZ) layer of the network where it captures and analyses traffic to detect any direct attacks. Upon detection the Honeynet server notifies the cyber security team and stores the observed attack in the database. H-Doctor focuses mainly on creating false files and decoy folders that will fulfil the ransomware’s encryption criteria. To prevent the attacker from encrypting the genuine file, the ransomware is tricked into encrypting the decoy folder first. The decoy folders are randomly generated in the event of a ransomware attack and contain random files of all types. The records of the attack are stored in the database which is accessed to tune the firewall.

2.3 Digital Twins-Assisted Deception

A DT is a virtual representation of a physical system that is continuously updated with real-time data throughout its lifecycle. It is more than just a static model as DTs can interact with and influence the actual systems they mirror, creating a dynamic feedback loop between the physical and digital parts. They are employed for various purposes such as design, development, simulation, analysis and operational optimization of physical systems, allowing organizations to monitor, control and enhance complex systems in real-time. Particularly, DTs are valuable because they offer a way to simulate what-if scenarios without affecting the real-world, making them perfect for performance improvements, failure prevention or optimization [SLA, EBC].

Going back to the origins of DT, the concept was originated in 1970 when NASA pioneered the idea of monitoring physical components during aerospace missions to diagnose issues and provide solutions. This early application laid the foundation for what would later become the DT. However, the current understanding of DT was introduced by Michael Grieves in the context of Product Lifecycle Management (PLM). Grieves expanded the concept emphasizing real-time interaction between digital and physical assets to optimize system performance throughout its lifecycle. In line with his vision, a DT is generally conceived as the grouping of machines (physical and/or virtual) or computer-based models that simulate, emulate, mirror, or “*twin*” the life of a physical entity. Other definitions describe it as a system that couples physical entities with their virtual counterparts, allowing organizations to leverage the advantages of both environments for improved performance, efficiency, and decision-making. This evolution marks a significant shift from traditional simulation models, positioning DTs as integrated, interactive, and self-adaptive systems capable of influencing their physical counterparts rather than just representing them [ALC].

To structure this concept, Grieves defined DT as a system composed of three interconnected spaces (see Figure 1):

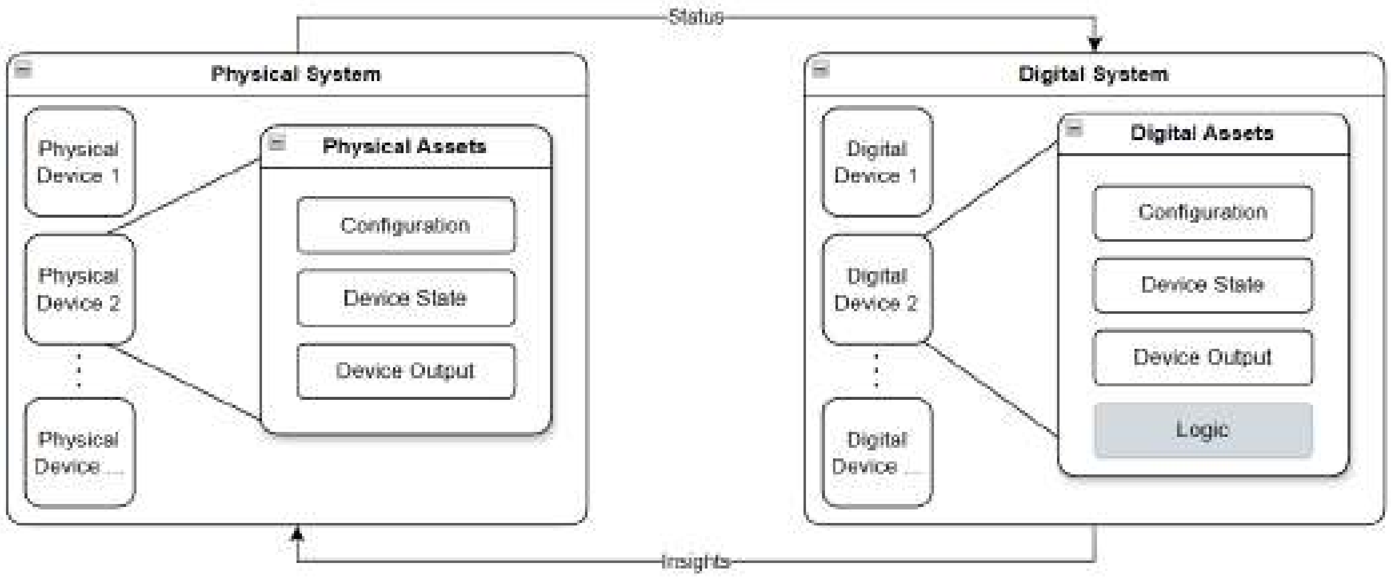


Figure 1 Connection between DT spaces

- ***Physical space***: real-world components, including operational technologies such as sensors, actuators, and controllers (e.g. Remote Terminal Units (RTUs) and PLCs). These elements continuously collect data and execute commands that regulate the system.
- ***Digital space***: representation of the physical assets in a virtual environment using digital models capable of simulating the states, conditions, and configurations. This space processes data to understand system behaviour, predict failures, and optimize the operations.
- ***Communication space***: link between the physical and digital spaces that enables data exchange through bidirectional interfaces. The DT receives real-time data from the physical world, process it to generate insights, and take decisions and adjustments to improve the system’s performance.

The key distinction between DT and conventional simulators is its bidirectional flow of information, where the digital model not only reflects the physical system, but also influences it by providing insights. Unlike traditional simulation tools, which only offer a static one-way analysis of predefined scenarios, DTs establish a continuous feedback loop, allowing for real-time monitoring, predictive analysis, and automated decision-making based on current system conditions [ALC]. Additionally, DTs stand out for the convergence of multiple advanced technologies. While Big Data facilitates the collection and real-time analysis of massive datasets from the physical twin, AI extracts critical insights to enhance performance. By leveraging AI and data analytics, DTs can detect deviations from expected behaviour, predict potential failures, and suggest corrective actions [SLA].

In line with those function capabilities, the International Telecommunication Union (ITU) defines a Digital Twin Network (DTN) as *a virtual representation of the physical network, designed to analyse, diagnose, simulate, and control the physical system through real-time interaction* [ITU]. This synchronization ensures an interactive mapping between physical and virtual elements, enabling diagnostics and simulations across diverse scenarios. A DTN brings transformative value for network management and innovation, but also introduces challenges such as large-scale data handling, complex modelling, and security requirements. Furthermore, DTN systems are not one-size-fits-all as different applications may demand diverse levels of capability and complexity, tailored to specific operational needs.

2.3.1 Capabilities and Evaluation Dimensions

The ITU standard defines different DTN Capability Levels (CL), each reflecting increasing functionality and autonomy. At the representation level (CL1), the DTN mirrors the physical network by capturing and updating its status and behaviours. On top of it, the interaction level (CL2) introduces bidirectional control, allowing the virtual twin to not only observe but also influence physical operations. The prediction level (CL3) incorporates data analysis and inference to diagnose issues and anticipate trends, enhancing decision support. The optimization level (CL4) employs AI, expert knowledge, and big data to enable intelligent, real-time decision-making and control. The autonomy level (CL5) envisions a symbiosis between the physical and digital entities, allowing for dynamic reconfiguration and fully autonomous management of the network [ITU].

To assess these capabilities, the standard outlines six key Evaluation Dimensions (ED): DT modelling (ED1) covers structural models for network topology and functional models for behaviour, simulation, and decision-making, tailored to specific application needs. Interactive mapping (ED2) assesses how effectively the DTN synchronizes with the physical part, ensuring accurate real-time data flow and control commands between both domains. The data service (ED3) dimension evaluates the DTN’s ability to handle data across its lifecycle from basic data acquisition to advanced processing, analysis, and further insights. Intelligence (ED4) measures the use of AI and ML, evolving from single network element intelligence to global adaptive and self-optimizing capabilities. User experience (ED5) focuses on the accessibility and usability of the DTN interface, emphasizing intuitive visualization, interactive control, and transparency for effective human oversight. Lastly, trustworthiness (ED6) evaluates the DTN’s capacity to ensure security, privacy, resilience, and reliability across its operations. Together, these dimensions form a comprehensive framework for assessing the maturity and effectiveness of DT implementations across different network environments and application scenarios [ITU].

2.3.2 Typical Applications Using Digital Twins

Currently, various applications of DT technology can be found across different fields. In the manufacturing sector [MEN], DT is used for:

- ***Real-time monitoring***: DT integrates historical, real-time, and predictive data to track past performance, monitor current operations, and anticipate future outcomes for optimization. By continuously analysing sensor data, DTs provide real-time insights into production efficiency, equipment performance, and operational status. This allows manufacturers to detect potential failures before they occur, optimize workflows, and enhance decision-making through data-driven analytics and predictive models.
- ***Production control***: DTs enable automated operations and adaptive responses to unexpected disturbances in manufacturing systems. By mirroring the physical production environment, DTs facilitate dynamic scheduling,

real-time quality control, and process adjustments based on changing conditions. This ensures that operations continue to run efficiently, even when facing disruptions or unexpected changes.

- ***Process evaluation***: DTs identify optimization opportunities that may not have been considered during the initial design phase. By running simulations and analysing different operational scenarios, DTs enable manufacturers to fine-tune production parameters, develop alternative workflows, and enhance resource utilisation. This contributes to cost savings, efficiency improvements, and the ability to adapt to new market demands.
- ***Enhanced fault detection***: DTs simulate extreme conditions and integrate data from normal operations to supplement training datasets for ML models used in monitoring. By combining real-world sensor data with simulated failure scenarios, DTs improve the accuracy of predictive maintenance systems. This reduces the risk of unexpected breakdowns.
- ***State monitoring***: DTs continuously compare real-time sensor values with expected digital simulations to detect anomalies in system behaviour. This allows manufacturers to identify early warning signs of performance degradation, quality defects, or deviations from optimal operating conditions. Automated alerts and self-correcting mechanisms can be triggered, ensuring that issues are addressed before they impact production output or product quality.
- ***Virtual testing***: DTs allow for safe and controlled experimentation with critical scenarios that could otherwise lead to significant financial losses or operational damages. By running virtual tests on new production strategies, system upgrades, or extreme environmental conditions, manufacturers can assess the impact of potential changes without disrupting actual operations.

Beyond these, DTs are also employed for optimizing production lines and logistics, improving supply chain, warehouse management, and transportation efficiency. By integrating real-time inventory tracking, automated routing, and predictive demand analysis, DTs help manufacturers reduce delivery times, lower operational costs, and respond dynamically to market changes [ZHA1].

In critical transportation infrastructure, DT plays a crucial role in improving the planning, design, operation, and safety of systems such as railways and highways. By providing real-time simulations and predictive analytics, DT technology enhances traffic management, maintenance scheduling, and risk assessment, contributing to more resilient and efficient transportation networks [YGA]. In aviation, DTs are primarily used for predictive maintenance, allowing for early detection of structural issues in aircraft. By continuously monitoring aircraft components and detecting dangerous changes in materials and mechanical structures, DTs help prevent potential failures and enhance flight safety. Additionally, DTs in aviation contribute to decision support, optimization, and diagnostics, further improving aircraft performance and reliability [BAR].

In the healthcare sector, DTs have been primarily applied to predict maintenance of medical devices, ensuring optimal performance in terms of examination speed, energy consumption, and system longevity [BAR]. Additionally, DTs are used to optimise hospital lifecycle management, resource allocation, workflow efficiency, and overall operational performance [BAR]. Moreover, a particularly promising area of medical DT research focuses on the development of human DT. These virtual models of patients could analyse personal medical history and contextual factors such as location, time, and activity to enable early illness detection and personalized treatment planning. By providing a detailed, real-time view of an individual’s physiological state, human DTs could revolutionize preventive healthcare, diagnostics, and tailored medical interventions [BAR].

DTs are also transforming water treatment systems as well by enhancing operational efficiency, predictive maintenance, and resource management. Leveraging real-time sensor data and predictive analytics, DTs enable optimized decision-making, ensuring treatment processes run efficiently and preventing incidents like the release of untreated water to the population. ML-driven predictive maintenance helps anticipate equipment failures, which reduces downtime. Additionally, process simulation models allow for better planning and optimization of biological treatment processes, ensuring compliance with environmental standards [ROD].

Finally, the energy sector is rapidly adopting DT technology to enhance efficiency, reliability, and sustainability in power generation and distribution. In fossil fuel power plants, DTs improve flexible operations by enabling real-time monitoring and predictive maintenance, reducing downtime and operational costs. For nuclear power plants, DTs support the entire lifecycle, from prototype design and operations to waste management. In renewable energy, DTs optimize design, integration, and performance of wind, solar, and other renewable sources, ensuring high reliability and maintainability. As energy systems become more complex, DTs are crucial for managing infrastructure, energy storage, and Cyber–Physical Systems (CPS) for a more resilient and sustainable energy infrastructure [AHM]. Regarding smart grid applications, proposed DTs enhance cybersecurity by simulating cyberattacks using threat vectors from an intelligence database. Attack results reveal vulnerabilities, leading to design improvements reflected in the DT. The threat database is continuously updated, triggering new attack simulations to assess resilience. This iterative process ensures real-time adaptation, strengthening the grid’s defence against evolving cyber threats [ATA].

Given the vast range of applications across different fields, DTs are revolutionising how systems are monitored, optimised, tested, and secured. By leveraging real-time operation, AI, and advanced simulations of the components, DTs enhance efficiency, enable predictive maintenance, and improve decision-making in complex and critical scenarios. As industries continue to evolve towards more automated and interconnected systems, DTs perfectly shape the needs for multiple domains.

2.3.3 Digital Twin for Deception

DTs can significantly enhance deception strategies by addressing several limitations of traditional honeypots, which often offer limited interaction capabilities, providing only basic responses to attackers and lacking realism needed to mimic real systems. This makes them easier for attackers to identify and evade. Additionally, honeypots typically function as standalone systems, making it difficult to integrate them with operational systems and analyse the large volumes of data that they generate. DTs overcome these challenges by creating highly realistic deception environments that closely replicate the appearance, behaviour and functionality of actual systems. By simulating real-world assets, DTs can provide a more immersive and convincing environment, making it harder for attackers to detect the deception. Furthermore, DTs support the dynamic adaptation of strategies, allowing the continuous refinement of the tactics in response to evolving threats and attack scenarios. This adaptability ensures that the deception remains effective even as the threat landscape changes. In addition to improving the realism of deception, DTs also enhance the generation of Threat Intelligence (TI), capturing detailed information about attacker objectives and behaviours within the deceptive environment and enabling security teams to reconstruct attack scenarios and trace the origins of security breaches [IMS, SUH].

For a better understanding about the leverages of the DTs in deception scenarios, a current SotA of DT-assisted deception approaches is presented:

- ***TwinPot***: the approach is an advanced honeypot system supported by DT technology [YKD], designed specifically for next-generation seaports. Its primary goal is to analyse adversaries’ behaviour and enhance cybersecurity by automating the classification of different attack types. The DT plays a crucial role in decoying attackers away from the actual system, ensuring that real infrastructure remains protected while simultaneously gathering behavioural intelligence from cyber threats. This collected data is then processed by TwinPot to identify patterns, assess vulnerabilities, and improve existing security mechanisms, ultimately fortifying the cyber resilience of maritime operations.
- ***SiHoneypot***: it is a HIH [LIA] that integrates DT technology to simulate autonomous vehicle sensors, such as LiDAR (Light Detection and Ranging), for cybersecurity research and defence. In this system, the DT acts as an intermediary, managing the communication between the LiDAR honeypot and the monitoring dashboard. Data from the honeypot is transmitted to a centralized server, where a hash value is computed and sent to the DT for verification. The DT then compares this hash with pre-existing values to detect anomalies or potential cyber threats. If a mismatch is identified, an alert is triggered, prompting the honeypot to intensify its monitoring and track suspicious activity. This approach significantly improves the detection of sophisticated cyberattacks targeting autonomous vehicle sensors.
- ***DiTwinIHon***: the approach is a DT-based framework [NGK] designed to support honeypots during their runtime by continuously learning from previous cyber activities. Unlike static security systems, DiTwinIHon evolves dynamically, using the intelligence gathered from past attacks to refine and enhance the honeypot’s response mechanisms. When the honeypot captures attack information, it is transmitted to the DT, where a TI module analyses the data. This analysis enables the adaptive configuration of the honeypot, ensuring that it remains effective against emerging threats. By integrating real-time data analysis and self-learning capabilities, DiTwinIHon significantly improves cyber defence mechanisms, making it a proactive security solution for safeguarding critical digital infrastructures.
- ***INCEPTION***: it is a deception-based cybersecurity platform [SUH] that employs DT technology to mislead attackers and divert them from actual systems. The core concept uses the DT as a honeypot to deceive threat actors and drive them to a fake system by mimicking the real one. INCEPTION’s design enhances its power by using the concept of fidelity, which defines how closely the virtual twin mirrors its physical counterpart. Rather than adopting a fixed fidelity level, the tool introduces a novel twisted fidelity approach, which dynamically adjusts the realism of the DT based on the attacker’s behaviour. High fidelity offers a highly realistic simulation that can lure adversaries, but at the cost of revealing more information and increasing complexity. In contrast, low fidelity reduces risk and cost by simplifying the twin and limiting the exposure of sensitive details. AI and ML models are leveraged to monitor attacker interactions and determine the optimal fidelity level in real time, creating a responsive decoy environment. This adaptive fidelity modulation not only strengthens security but also makes INCEPTION a versatile and intelligent solution for protecting critical cyber-physical infrastructures.
- ***DECEPTWIN***: the approach is a deception framework [IMS] for the Internet of Vehicles (IoV) that combines DT technology with blockchain to defend against threats targeting interconnected vehicles and infrastructure. Unlike traditional honeypots that offer limited realism and interaction, DECEPTWIN brings a highly interactive DT-based environment that closely mimics real IoV systems. To ensure the integrity and trustworthiness of

the deception system, the approach integrates blockchain to secure communications, preserve data integrity, and enable traceability of attacker actions.

DT-powered honeypots showcase diverse implementations and functionalities in cybersecurity. TwinPot is a high-interaction system that passively gathers intelligence on cyber threats while dynamically adapting to evolving attack patterns. By leveraging a diverse set of security tools and decoys, it enhances threat analysis and improves cyber resilience. SiHoneypot, in contrast, actively engages with attackers by simulating autonomous vehicle sensors, though it operates within a fixed configuration. Its structured approach enables precise monitoring while maintaining a controlled security environment. DiTwinIHon focuses on continuous learning from past attacks, dynamically refining its responses over time to strengthen security measures. INCEPTION introduces a novel approach by using the DT itself as a deception mechanism, misleading attackers into a simulated environment in which fidelity dynamically changes based on attacker behaviour. DECEPTWIN leverages the combination of DT-based deception and blockchain to defend IoV infrastructures while preserving integrity and traceability. These solutions highlight the potential of DT technology in deception-based cybersecurity, enabling both adaptive and proactive strategies to safeguard critical digital infrastructures.

Table 3 provides a structured comparison of their key characteristics, highlighting their implementation methods, interaction levels, purposes, and adaptive behaviours.

Table 3. DTs working as honeypots

Technology	Implementation	Interaction	Purpose	Activity	Uniformity	Action
TwinPot	Physical	High-interaction	Research	Passive	Heterogeneous	Dynamic
SiHoneypot	Virtual	High-interaction	Both	Active	Homogeneous	Static
DiTwinIHon	Virtual	High-interaction	Research	Passive	Homogeneous	Dynamic
INCEPTION	Virtual	High-interaction	Production	Active	Heterogeneous	Dynamic
DECEPTWIN	Virtual	High-interaction	Production	Active	Heterogeneous	Dynamic

The classification in the table above organizes technologies based on several key attributes that define their functionality and role in cybersecurity. Implementation can be either virtual, running on a single server, or physical, deployed on dedicated hardware. Interaction levels vary, with HIHs closely mimicking real systems to engage attackers, while low-interaction variants provide limited functionality to minimize risk. Honeypots can be distinguished based on purpose to research honeypots, which gather intelligence on attack strategies, and production honeypots, which actively divert malicious actors from real systems. Activity is categorized as passive, focusing only on data collection, or active, engaging attackers to enhance system defence. Uniformity refers to whether the honeypot uses a single deception mechanism (homogeneous) or multiple decoys and security tools (heterogeneous) to detect a broader range of threats. Finally, action determines whether a honeypot is static, maintaining a fixed configuration, or dynamic, adapting its behaviour based on ongoing attack patterns.

These classifications highlight the versatility of DT solutions, which not only decoy attackers away from critical infrastructure but also continuously analyse, learn, and refine security mechanisms based on real-time cyber threat intelligence. By leveraging automated classification, real-time monitoring, and predictive analytics, DT-driven deception techniques enhance the detection, response, and mitigation of sophisticated cyber threats. As cyberattacks become more complex, DT-empowered honeypots represent a critical evolution in cybersecurity, providing organizations with resilient, adaptive, and intelligent defence strategies to safeguard their digital environments.

2.4 Virtual Persona-Assisted Deception

VP constitutes a digital construct of an individual, encapsulating attributes such as behavioural patterns, preferences and demographic characteristics. These personas may serve as authentic digital representations of real users or as artificially constructed entities designed for distinct purposes. Their utilization extends across multiple domains, including social networking, cybersecurity, digital marketing, and AI training [SAL]. VPs play a crucial role in analysing and shaping user behaviour across multiple disciplines, including cybersecurity, social media, digital marketing, and AI applications [PER]. Their study enables professionals to enhance security measures, optimize engagement strategies and refine AI-driven interactions [SAL]. By integrating VP analysis into these domains, researchers and professionals can develop more adaptive, secure, and user-centric systems, thereby advancing both theoretical understanding and practical applications of digital identity modelling [SAL].

Further supporting the points discussed, it can be observed that recent research highlights the growing importance of VPs in a range of fields, showing how they can be used not just to represent users but to actively improve digital experiences. In the study *Enhancing Collaborative Creativity with Virtual Dynamic Personas* [BON] researchers introduced personas into real-time collaborative design settings to help teams better understand and anticipate user needs. These personas served as stand-ins for future users, guiding decision-making and encouraging more user-centred outcomes. Similarly, The Influence of the *Online Persona on University Students* [ZIY] explored how students’ self-created digital identities impact their mental health and social interactions. The study [ZIY] found that online personas can shape how individuals see themselves and relate to others, raising important questions about the psychological effects of digital identity. Meanwhile, *Data-Driven Personas for Enhanced User Understanding* [JAN] focused on how behavioural data can be used to build more accurate and meaningful user profiles. These data-driven personas help designers and developers tailor digital systems to real user needs. Together, these studies demonstrate how VP are becoming essential tools for improving creativity, understanding user behaviour, and designing more responsive and personalized technologies.

2.4.1 Methods For Virtual Persona Analysis

This subsection provides a structured overview of key methods used to analyse and verify the authenticity of VP. It is organized into six interconnected areas: behavioural analysis, network analysis, ML-based classification, Natural Language Processing (NLP), metadata and forensic analysis, and deepfake detection. Each of these approaches offers a unique lens for identifying inauthentic, synthetic, or compromised personas within digital environments. The section concludes with a discussion on AI-driven reasoning in multi-agent systems, focusing on its application within the Arrowhead [ARR] framework. This highlights how advanced ML models and LLMs are being integrated to support intelligent, adaptive decision-making in complex, automated ecosystems.

2.4.1.1 Behavioural analysis: identifying unique digital signatures

Behavioural analysis entails the examination of user interactions, linguistic patterns, and engagement frequencies to establish distinctive behavioural signatures. By leveraging computational behavioural profiling, analysts can detect anomalous activities and inconsistencies, often indicative of AI-generated personas, compromised accounts, or coordinated inauthentic behaviour [MAN, CCQ, RAH]. For instance, linguistic irregularities, temporal anomalies in activity patterns, and deviations in engagement behaviours can serve as key indicators of automated actors, bot-driven operations, or adversarial social engineering campaigns [MAN, RPM]. This approach is particularly instrumental in distinguishing genuine user interactions from machine-generated responses, thereby enhancing persona authenticity verification [MAN, RAH].

2.4.1.2 Network analysis: graph-based identification of coordinated inauthentic behaviour

Network analysis applies graph theory and computational social network modelling to investigate the relational dynamics and interaction structures of VP. By mapping nodes, clusters, and central influencers, analysts can uncover patterns of coordinated inauthentic activity, such as bot networks, disinformation amplification, and algorithmic propaganda dissemination [MGV, GBJ]. The integration of graph-based anomaly detection techniques enables the identification of highly connected nodes exhibiting unnatural engagement behaviours, allowing for the detection of synthetic personas embedded within broader influence operations [SZF]. This methodology is essential for mitigating manipulative digital campaigns and preventing the spread of AI-enhanced misinformation [MGV, ZAH].

2.4.1.3 Machine Learning approaches for persona classification

This subsection outlines key ML strategies used to identify and classify VPs, focusing on the detection of synthetic identities, as described in the following points:

- ***AI-based classification of authentic vs. synthetic personas***: ML models are trained on high-dimensional labelled datasets to distinguish between real and fabricated personas with high precision. These models analyse a diverse set of features, including behavioural patterns, relational metadata, and digital footprint anomalies, to establish probabilistic classifications of persona authenticity [JJS]. Advanced Deep Learning (DL) architectures, such as transformer-based classification models and ensemble learning techniques, continuously adapt to evolving adversarial evasion tactics, thereby enhancing the robustness of synthetic persona detection mechanisms.
- ***NLP for text-based persona analysis***: the application of NLP facilitates the identification of synthetic textual artifacts through the analysis of language structures, sentiment patterns, and semantic coherence [JJS, VAI]. NLP-driven persona validation employs techniques such as stylometric analysis, topic modelling, and sentiment forensics to differentiate between human-authored content and AI-generated narratives. This approach is particularly effective in detecting deepfake-generated text, scripted bot interactions, and linguistic anomalies indicative of automated persona fabrication.
- ***Metadata and forensic analysis: digital provenance verification:*** metadata and forensic analysis focus on the validation of persona authenticity through the examination of digital footprint attributes such as Internet Protocol (IP) geolocation, temporal activity markers, device fingerprints, and session entropy metrics [VAI]. By leveraging forensic digital watermarking and cross-referencing geospatial inconsistencies, analysts can uncover synthetic identity fabrications, metadata manipulation attempts, and falsified digital traces. These forensic methodologies play a crucial role in tracing the origin of deceptive personas and attributing malicious

activities to specific actors within digital ecosystems.

- ***Deepfake and AI-generated persona detection:*** Advancements in Generative Adversarial Network (GANs) and synthetic media technologies have led to the proliferation of AI-generated personas and deepfake-based fabrications, necessitating the development of high-fidelity detection models [VAI]. State-of-the-art adversarial deepfake detection algorithms now leverage multi-modal signal analysis to identify subtle visual artifacts, micro-expression inconsistencies, and photometric discrepancies in AI-synthesized identities [VAI]. These detection models integrate ensemble learning techniques and adversarial robustness mechanisms to enhance recall while reducing false positive rates, ensuring the accurate identification of manipulated digital identities.
- ***AI-driven reasoning for multi-agent system management in Arrowhead:*** The integration of ML and LLMs is increasingly utilized to enable autonomous decision-making within complex System-of-Systems (SoS) environments, such as those managed within the Arrowhead [ARR] framework. These models process high-dimensional data, including structured Application Programming Interfaces (API) specifications, real-time system states, and user-defined intents, to optimize resource allocation and workflow execution. Advanced multi-agent architectures, leveraging transformer-based LLMs and hierarchical reasoning frameworks, facilitate adaptive planning and dynamic orchestration of microservices interactions [HEG]. Additionally, the incorporation of iterative prompt engineering methodologies, including the Plan-and-Solve and ReAct paradigms, enhances the robustness and contextual awareness of automated system management, ensuring resilient and scalable operations in Arrowhead-based ecosystems [HEG].

2.4.2 Advanced Tools, Platforms for Persona Detection, and Analysis

This subsection outlines a range of AI-driven tools developed to detect synthetic personas and manipulated digital content. Platforms, such as Botometer, Reality Defender, and DeFake apply ML, computer vision, and multimodal analysis to identify bots, deepfakes, and inauthentic behaviour across social media and audiovisual data. These tools support real-time detection and content verification, helping preserve the credibility of digital environments.

In parallel, the subsection explores the use of AI in cybersecurity deception, focusing on tools like Re:Scam and dAIsy, which engage phishing actors using artificial personas to waste their time and reduce the impact of scams. In addition, AI-powered training platforms, such as KnowBe4 and Jericho Security, simulate phishing attacks to assess user awareness and improve response capabilities. Together, these technologies form a practical toolkit for detecting digital manipulation, enhancing user resilience, and reinforcing the security of online interactions.

2.4.2.1 Botometer: AI-driven bot detection in social media networks

Botometer is a ML-based analytical tool developed to assess the authenticity of Twitter (currently known as X) accounts through the analysis of a range of behavioural and linguistic indicators [VAR]. It estimates the likelihood that an account is automated or artificially manipulated by evaluating several computational features, including the structure and dynamics of the user’s friend network, tweeting frequency, content composition, sentiment patterns, and engagement behaviours such as reply and retweet activity. Botometer [BOT] has proven to be a valuable asset in maintaining social media platform integrity, supporting efforts in disinformation monitoring, botnet detection, and the broader identification of inauthentic activity. As such, it serves as a valuable resource for cybersecurity researchers, media analysts, and policymakers addressing the challenges of digital manipulation [YAN].

2.4.2.2 Reality Defender: Real-time deepfake detection and synthetic media forensics

Reality Defender [REA] is an AI-driven detection platform developed to identify deepfake manipulations in real-time video streams, with a focus on enhancing media authentication and forensic analysis [WIR]. The system leverages a combination of computer vision techniques, adversarial learning frameworks, and multimodal content analysis to detect indicators of synthetic media. These include facial morphing artifacts and biometric inconsistencies, irregularities in motion dynamics and audio-visual synchronization, as well as subtle digital traces characteristic of GAN-based content. Owing to its real-time processing capabilities, Reality Defender plays a critical role in countering deepfake-enabled scams, digital impersonation, and the broader dissemination of AI-generated misinformation, making it a valuable tool in both investigative and preventive contexts [WIR, REA].

2.4.2.3 Defake: AI-powered multimodal analysis for deepfake detection

DeFake [DEF] employs ML algorithms and neural network architectures to assess the authenticity of synthetic audiovisual content. Unlike conventional detection methods that focus on image or video analysis, DeFake utilizes [DEF]:

- ***Cross-modal forensic analysis*** to identify discrepancies between visual and auditory data streams.
- ***Speech pattern recognition*** to detect synthetic voice modulation and AI-generated speech synthesis.
- ***Context-aware DL models*** that evaluate semantic consistency in synthesized narratives.

2.4.2.4 Cybersecurity deception: creating fake digital identities for phishing attacks

The tool dAIsy, the AI Granny [AIG] by O2, is an innovative VP designed to engage fraudsters in real-time phone conversations, wasting their time and preventing them from scamming real customers. Re:Scam [RES] is an AI-powered email bot developed by Netsafe to combat email scammers by wasting their time and resources. It works by engaging fraudsters in prolonged conversations, using humour, confusion, and endless follow-up questions to keep them occupied. Users can forward suspicious emails to Re:Scam [RES], which then responds as a variety of different personas, stringing scammers along indefinitely. By doing so, Re:Scam helps reduce the success rate of phishing scams and protects potential victims from financial and personal harm [AIG]. The article [BAJ] explores how AI can be used to automatically engage with scammers, wasting their time and reducing their success rates. For the work, the researchers consider an automated scam-baiting system using ChatGPT, which interacts with scammers in a natural and engaging way. The study [BAJ] evaluates the effectiveness of AI-generated responses in prolonging scam interactions while ensuring safety and ethical considerations. The results suggest that AI-based scam-baiting could be a valuable tool in cybersecurity, helping to disrupt fraud networks [BAJ].

2.4.2.5 Security training: simulating cyber threats with AI-generated adversaries

AI-generated VPs can also be utilized to simulate phishing attackers, providing a controlled environment to assess individuals’ susceptibility to phishing and to educate them on effective countermeasures. KnowBe4’s [KB4] Security Awareness Training: a notable example of such a system which combines AI with an extensive library of training content and simulated phishing attacks. This platform delivers personalized training and conducts simulated phishing tests to help users identify and respond to phishing threats effectively [KB4]. Jericho Security [JER] offers an AI-powered phishing simulation training platform that conducts hyper-realistic, personalized attack simulations. This tool [JER] allows organizations to test and train employees, enhancing their ability to recognize and respond to phishing attempts. These AI-driven training solutions are instrumental in building resilience against phishing attacks by providing realistic, interactive learning experiences.

2.4.3 Future Directions and Open Challenges in Virtual Persona Research

The following paragraph addresses key research gaps and outlines future directions in the study of VPs, with attention to both technical and ethical considerations. As AI generated identities and deepfake technologies become more sophisticated, the development of unsupervised learning models capable of detecting novel forms of deception without labelled data is increasingly important. Advancing deepfake detection through multimodal analysis, along with the creation of hybrid systems that integrate human judgment with AI, are central to enhancing detection accuracy. The growing use of AI powered personas in fraudulent activities, particularly on social media, underscores the need for real time detection mechanisms, behavioural authenticity checks, and increased public literacy. In parallel, the field must confront challenges related to scalability, adversarial resilience, privacy protection, and fairness in algorithmic design. Sustained interdisciplinary collaboration and the establishment of ethical, transparent governance frameworks will be essential to ensure the responsible deployment of AI in digital identity management.

2.4.3.1 Developing unsupervised learning models for adaptive detection

The dynamic nature of AI-generated personas and adversarial deception tactics necessitates the implementation of unsupervised learning methodologies that can adapt autonomously to emerging fraudulent patterns. Unlike supervised learning approaches that rely on predefined labelled datasets, unsupervised persona detection models utilize anomaly detection algorithms and unsupervised clustering techniques to identify novel indicators of synthetic persona fabrication [ABD]. By leveraging self-learning AI architectures, such models can improve adaptability to evolving digital deception methods, ensuring that persona detection systems remain resilient against increasingly sophisticated adversarial strategies. Further research is required to evaluate the scalability and generalizability of these unsupervised learning paradigms in real-world digital ecosystems [ABD].

2.4.3.2 Detecting deepfake-enhanced personas

DeFake [DEF] uses ML and neural network architectures to assess the authenticity of synthetic audiovisual content. Unlike traditional methods that focus only on visual or audio analysis, DeFake [DEF] takes a cross-modal approach, identifying inconsistencies between what is seen and heard in a video. It also applies speech pattern recognition to detect signs of synthetic voice modulation and AI-generated speech. Additionally, it uses context-aware DL models to evaluate the overall coherence of the narrative, helping to determine whether the content has been artificially generated or manipulated [DEF].

2.4.3.3 Addressing the rise of AI-powered persona fraud in social media

AI-generated personas are increasingly leveraged for fraudulent activities across social media platforms, financial institutions, and disinformation campaigns. Recent research [LUO] has examined public perception of deepfake personas, revealing that individuals struggle to differentiate synthetic identities from real users, exacerbating the risk of social engineering fraud and digital deception [LUO].

To mitigate this threat, AI-driven persona detection systems need to combine several essential elements. These include: (i) real-time anomaly detection to identify high-risk synthetic personas, (ii) automated checks to verify the authenticity and consistency of user behaviour, and (iii) public awareness efforts to improve understanding of deepfake risks and the dangers of AI-generated persona fraud [SRI, SAI]. Further research is needed to explore psychological and sociological factors influencing user susceptibility to AI-generated persona manipulation, with the goal of developing more resilient social media integrity frameworks [ADL].

In conclusion, the field of VP analysis is rapidly evolving at the intersection of AI, cybersecurity, and digital identity modelling, necessitating robust detection frameworks to counter AI-generated personas, deepfakes, and automated deception [SRI, SAI]. While DL-based classification, behavioural profiling, and network forensics have improved detection accuracy, scalability issues and adversarial evasion tactics highlight the limitations of existing methodologies [SRI, LQL]. Beyond technical challenges, privacy, fairness, and regulatory compliance remain critical. Privacy-preserving AI, eXplainable AI (XAI), and fairness-aware algorithms are essential for mitigating biases and ensuring ethical governance [ADL, LQL]. Future advancements in hybrid AI-human verification, real-time deepfake detection, and adaptive adversarial learning will be key to enhancing persona authentication security [SRI, SAI]. To maintain a secure and trustworthy digital ecosystem, research must focus on adversarial AI detection, legal and ethical frameworks, and interdisciplinary collaboration across AI, behavioural science, and cybersecurity [ADL, LQL]. Balancing technological innovation with responsible AI governance is crucial in mitigating threats in digital identity verification [ADL, LQL].

2.5 Monitoring And Security Analytics

Adversarial ML has seen a rise in the research domain because of its extensive use in cybersecurity, and especially in monitoring and security analytics. As ML models are increasingly deployed for applications such as intrusion detection, malware classification, phishing detection, and object detection, adversarial attacks—where small, intentional perturbations are introduced into the input data to mislead the model—pose a serious threat. These attacks are used more day by day, so it is crucial to understand the state of the art in adversarial ML in order to develop robust defences and improve overall security. A key focus of recent research has been to examine the effectiveness of adversarial attacks in various domains and to develop defensive strategies. To do so, it is first important to create adversarial attacks aiming at networks and cybersecurity systems. For example, research has demonstrated how adversarial ML techniques can manipulate Intrusion Detection Systems (IDS) by generating adversarial samples that bypass detection models [AJT]. These adversarial samples are designed to trick models, such as decision boundaries with a simple architecture or poorly trained classifiers. In particular, black-box attacks have been shown to be highly effective because they do not require access to the internal parameters of the model, making them more challenging to defend against. Techniques like poisoning attacks, where attackers add adversarial data into training sets, and evasion attacks, where adversaries add not visible by humans’ perturbations to inputs so as to trick the model but still avoid being detected, can be highly crucial for network security [WXW].

Researchers have proposed various methods to combat these adversarial threats. One way to improve a model’s defence is adversarial training, when the model is trained both on clean data and adversarial examples, hence it learns how to resist those examples. Adversarial training is particularly useful but sometimes it can be problematic because it increases the computational cost and there is a chance to overfit to some types of adversarial attacks [MLY]. Additionally, robust adversarial detection techniques have been developed to identify and mitigate adversarial samples before they can affect the model’s output. For example, AI models usually try to detect anomalies in input data that may be adversarial examples [AAF]. Also, researchers search for ways to detect attacks in malware classification systems. Adversarial attacks on malware detectors may reduce the accuracy of classification models, enabling malicious actors to evade detection. A huge step towards that path is the ability to identify flaws in classifiers used for detecting Windows Portable Executable (PE) malware [LWZ]. Researchers try to find out how these attacks aim to trick malware detectors and to make clear that there is a need for more robust defences in real-world cybersecurity applications [YRW].

Another target for adversarial attacks is the images, which are used for image recognition or computer vision. In these domains, adversarial inputs aim to lead the model into wrong predictions and classifications by adding small perturbation at some of an image’s pixels without being visible to the human eye. This issue can be harmful for security models that are used for automated inspection. Efforts to defend against these attacks include the development of more robust image classification models and the application of defensive techniques such as input transformation and model regularization [AJK]. Another attack target can be the phishing detectors that use AI to classify potential phishing websites or emails. In this type of attack, adversaries aim to change a website or email’s layout to make it undetectable from detection systems [ACY]. Due to the evolution of these attacks, it is nowadays urgent to improve the defensive systems’ robustness, including hybrid detection systems that combine multiple classifiers to improve accuracy and resistance to manipulation [YSA].

These innovative attacks are pushing the boundaries of what is possible in terms of adversarial strategy, highlighting the need for continuous innovation in defence mechanisms. In addition, the evolution of black-box adversarial attacks has shown many advancements. Black-box attacks aim to manipulate images to create adversarial examples that can trick even well-trained models. Black-box attacks use the model’s output to adjust the input iteratively based on observed responses. These attacks make clear how unreliable a machine model can be when used for automated security without added safety [QCL]. As adversarial attacks keep evolving, new types of defences need to be used. An effective way to counter evolving adversarial threats is through automated ensemble methods, which systematically combine multiple models or classifiers to strengthen resistance to evasion attacks [SXW]. By combining many models, the system has increased robustness because there are more layers of defence, and one attack may trick some models but not all of them at the same time. Another innovative defence is adversarial detection, which uses ML models to identify adversarial inputs before they may cause harm. This type of defence is useful when facing poisoning attacks, where the attacker messes with the model’s training dataset [WGL]. Except from creating defences versus each type of attack, the model has also to evaluate its accuracy and confidence. While adversarial training and detection methods can increase robustness, they may come at the cost of performance in benign conditions [CTP].

Another goal of the researchers is to find a way to increase a model’s robustness to adversarial attacks without decreasing its effectiveness under normal operating conditions [WXW1]. Finally, there is growing attention to the use of adversarial attacks for practical applications beyond cybersecurity, such as in the context of recommendation systems and automated decision-making tools. The systems that rely heavily on ML to make predictions or recommendations are usually the target of adversarial attacks aiming to manipulate the model’s output. To deal with this, the researchers develop new defence strategies, such as using GANs to create their own adversarial examples and then use them to train their models [DNM].

In conclusion, adversarial attacks and security analytics keeps changing day by day, leading to improvements in both the techniques used by attackers and the defences being developed to counter them. The real task is to improve the robustness of ML systems without affecting their ability to provide accurate and reliable results in real-world applications. As adversarial strategies become more sophisticated and pervasive, the need for continuous innovation in both attack detection and defence mechanisms remains paramount.

3 Design of the Deception Layer for AIAS

This section presents the core design of the deception layer for AIAS, detailing the overall architecture, its key components, and the way they interact to achieve seamless integration.

3.1 Overall Architecture for Deception Layer

This section provides an overview of the deception layer and its components. Diagram in Figure 2 shows the interaction between the parts in the architecture and provides a high-level understanding of the functioning. Not only internal components are present, but also a brief outlook of the connections and synergies with other modules of AIAS.

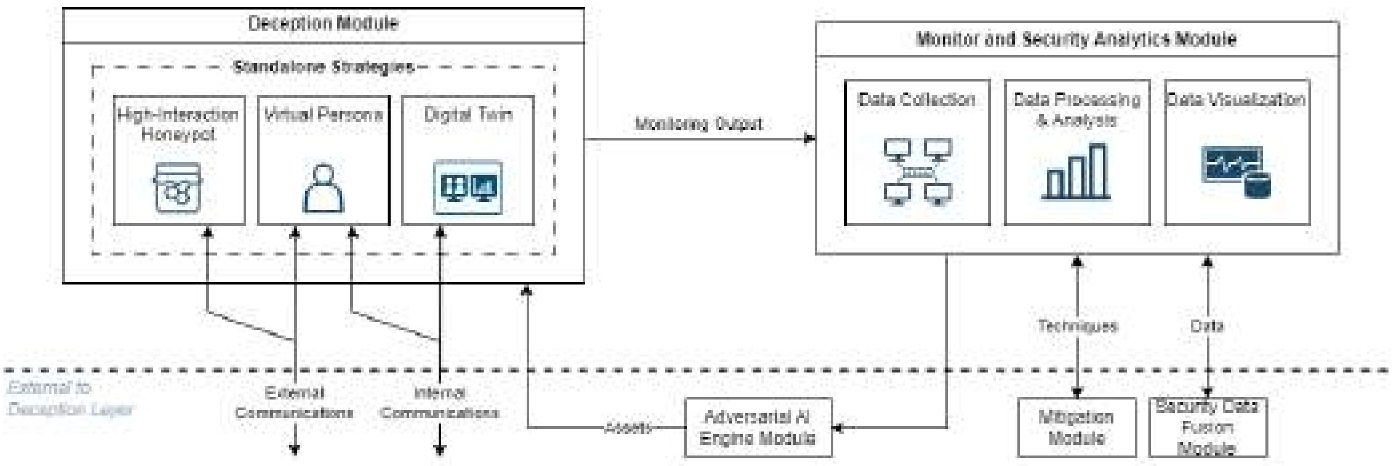


Figure 2. Blueprint of AIAS Deception Layer

The deception module is composed of the diverse standalone deception strategies contemplated for the deception layer such as HIHs, DTs and VPs. These components individually manage their respective incoming communications, which can be considered external for those that expose the system to the outside and actively deceive the adversaries or internal for those that are used internally to detect potential security deficiencies and provide valuable insights and recommendations for further adjustment of the system. Each strategy is additionally responsible of providing the required output to the monitor and analytics module, which processes acquired information and integrates it with insights from the security data fusion module, ensuring the most refined intelligence is delivered to the mitigation and adversarial AI engine modules. This last-mentioned module employs this intelligence to supply the deception module with the most advanced adversarial AI attack vectors. A deeper understanding of the specific components in the architecture is covered in the following sections.

3.2 Honeypot

In alignment with the core objectives of the AIAS project, which aims to conduct in-depth research on adversarial AI to design and develop an innovative security platform to protect both AI systems and AI-based operations within organizations, one of the critical components under evaluation is the use and implementation of honeypots as deception mechanisms. By deploying honeypots, vulnerable or attractive digital assets can be simulated to lure attackers and observe their behaviour in a controlled space.

To carry out this aspect of the study, the T-Pot platform will serve as basis for the deployment. T-Pot is a Linux-based system that integrates over 20 types of honeypots, offering a diverse range of options for traffic capture, behavioural monitoring, and data visualization. The implemented T-Pot instance is hosted in a VM located within a VPN, ensuring that all honeypots operate in an isolated and controlled environment, shielded from direct exposure to the internet, and fully managed only by authorized personnel.

In line with the AIAS project’s research lines, the analysis is centred on four specific honeypots, which have been selected with emphasis on their relevance to attack surfaces and protocols of interest:

- ***Cowrie***: emulates remote access services over SSH or Telnet.
- ***Mailoney***: simulates a Simple Mail Transfer Protocol (SMTP)-based mail server, facilitating the capture of email-related attacks.
- ***Wordpot***: acts as a decoy WordPress server, targeting Content Management System (CMS)-oriented exploits.
- ***Dionaea***: a highly versatile honeypot that emulates multiple key protocols, such as File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), and MySQL, among others.

The operational objective is to keep these honeypots active continuously, in order to capture all traffic and interactions targeting the emulated services. This allows to perform a comprehensive analysis and draw conclusions about the nature and patterns of the attacks received, supporting the project’s broad goal of enhancing cybersecurity for AI-based infrastructures.

Since T-Pot platform is deployed within a VPN, it is necessary to generate synthetic traffic for populating the honeypots with relevant data. The simulated types of attacks within the controlled environment are the following:

- ***Web service (Wordpot)***: simulated access attempts to the web server hosted by the Wordpot honeypot.
- ***Email communication service (Mailoney)***: phishing simulations leveraging Swaks to send SMTP messages.
- ***Remote access services (Cowrie)***: simulated connection attempts via SSH and Telnet.
- ***Dionaea***: simulated publisher-subscriber interactions, file transfers between machines, and unauthorized access to database information over MySQL.

Simulated attack scenarios contribute the honeypots with realistic interaction logs and captured data for further analysis. Additionally, T-Pot platform offers integrated tools for visualizing information through Kibana dashboards, enabling deeper and more structured exploration of the events, and extracting valuable insights, such as: attacker IP address, port used in the attack, geographic origin of the connection, plaintext commands issued by the attacker, and other metadata related to the attack behaviour.

With a solid understanding of the T-Pot environment, its functionality, capabilities, and scope, the next objective is to evolve these selected honeypots (Cowrie, Mailoney, Wordpot, and Dionaea) into HIH. The aim is to create environments with a high degree of realism and attacker engagement, designed to convincingly simulate real systems. This approach is intended to provide attackers with a sense of full operational freedom, thereby increasing the likelihood of capturing richer, more detailed behavioural data.

The implementation of a custom honeypot within the T-Pot environment allows for the evaluation of the depth, flexibility, and scalability of the platform, while simultaneously exploring the design of novel strategies and mechanisms for early detection of cyberattacks. This effort represents a valuable theoretical and development-oriented contribution within the scope of the AIAS project.

Achieving this transformation, T-Pot becomes a much more comprehensive platform for the analysis and study of cyberattacks. This enhanced capability not only boosts the quality of data collected, but also strengthens the contribution to cybersecurity, particularly in the domain of attack simulation, threat behaviour analysis, and deception-based defence strategies.

3.3 Digital Twin - Robotic Arm

The robotic arm DT integrates both physical and virtualized components in robotics, enabling the simulation, analysis, and adjustment of motion-controlled devices by leveraging advanced technologies such as ROS and Unity.

3.3.1 Overview of the Main System Components

The DT of the robotic arm consists of four primary components, each responsible for a specific aspect of the workflow as also shown in the Figure 3 and Table 4.

Figure 3. DT Robotic Arm Internal Components

Table 4. Components and Roles of the DT Robotic Arm

Component	Functionality
Unity Application (HoloLens 2 & PC)	• Provides a mixed-reality interface for controlling and visualizing the robot. • Implements ROS TCP Connector to communicate with ROS 2. • Handles gesture-based input, UI controls, and DT visualization. • Updates the robot’s position, joint states, and execution status in Augmented Reality (AR).
ROS 2 Middleware (Ubuntu)	• Acts as the core computation layer, processing user commands and executing robot motion. • Runs MoveIt for Inverse Kinematics (IK) and motion planning. • Uses MoveIt Servo for real-time joint control. • Publishes robot joint states and motion updates to Unity.

HoloLens 2 (AR Visualization & Interaction)	• Displays robot movement in AR. • Supports gesture-based manipulation and UI interaction. • Streams data from Unity, overlaying digital robot models on the real-world workspace.
YaskawaHC10 Physical Robot (Optional in Real-World Execution)	• Executes movement commands from MoveIt Servo in ROS 2. • Sends real-time feedback on joint positions and execution status. • Operates in two modes: ▪ Simulation mode: Commands are executed in Unity’s DT. ▪ Physical mode: Commands are sent to a real robotic arm for execution.

3.3.2 Main Functions of the System

Data exchange overview and APIs: Seamless communication between Unity, ROS, and MoveIt! is essential for maintaining real-time robot control and accurate AR visualization. The data exchange process follows a structured flow, ensuring commands are sent, executed, and synchronized across all systems. This section provides an in-depth look at each exchange, explaining its role, challenges, and impact on system stability. Each of the below-mentioned end points can be provided as a honeypot and deception playground for external attackers, collecting / profiling external / internal abnormal operation data to provide towards the AIAS monitoring and security analytics toolbox. Table 5 includes the corresponding data exchange and APIs.

Table 5. Data exchange and APIs

Direction	Data Type	Message Name/Topic	Size	Frequency	Description
Unity ↔ ROS (ROS TCP)	TCP Handshake/Keepalive	N/A	Small (1-2 bits)	Persistent (every few seconds)	Ensures the ROS and Unity TCP connection remains active and synchronized.
Unity → ROS	Service Request	ROS /srv	Varies (64 bits)	On-demand	Service requests sent from Unity to trigger ROS-based services, such as starting or stopping nodes.
ROS → Unity	Service Response	ROS /srv	Varies (64 bits)	On-demand	Responses from ROS to Unity following service requests for controlling ROS nodes.
Unity → ROS	Float64	/unity/joint_1_s to /unity/joint_6_t	Small (64 bits)	Event-driven by Unity UI	Joint velocity command sent from Unity to ROS for controlling individual joints.
Unity → ROS	PoseStamped	/unity_object_pose	Medium (128 bits)	10Hz	Sends the position and orientation (pose) of a Unity object to ROS.
ROS → Unity	Float64MultiArray	/unity_joint_positions	Medium (128 bits)	10-50Hz	Provides joint position data from ROS to Unity for real-time robot visualization.
ROS → Unity	JointState	/joint_states	Medium (128 bits)	10-50Hz	Sends current joint states to Unity for real-time visualization.
ROS → RViz	JointState and PoseStamped	/joint_states, /tf	Medium (128 bits)	30Hz	Provides joint and pose data for visualization in RViz.
ROS → MoveIt!	JointState, PoseStamped	/joint_states, /move_group/feedback	Large (256 bits)	10-100Hz	Provides MoveIt! with joint states and motion feedback for planning and control.

MoveIt! → ROS	JointTrajectory	/move_group/trajectory	Large (256 bits)	Motion Event-driven	Planned motion commands for the robot’s joint positions over time.
ROS → Servo	JointJog, TwistStamped	/servo_node/delta_twist_cmds	Medium (128 bits)	100Hz	Real-time joint or end-effector movement commands for servoing applications.
Unity → ROS	Bool (for start/stop control signals)	/unity_control_signal	Small (1 bit)	Event-driven	Control signals from Unity to start/stop ROS-based processes like MoveIt! Servo.
Unity → ROS	Float64	/joint_control	Small (64 bits)	Event-driven	Velocity or position command sent from Unity to ROS for joint control.
ROS → Unity	StringMsg	/rosout	Small (varies)	As needed (error logging)	Provides ROS error, warning, or status logs to Unity for debugging and monitoring.

Establishing connection - Unity-ROS TCP link: before any data transmission, Unity and ROS establish a persistent network connection through the ROS-TCP Connector. This connection enables bidirectional communication, ensuring Unity can send movement commands while receiving live robot state updates.

The connection must remain stable, as interruptions can cause Unity to lose control of the robot. If the TCP handshake fails, or the connection drops, users must ensure that ROS and Unity are on the same network and that the ROS machine’s IP is correctly configured. If the connection repeatedly fails, restarting the ROS TCP Endpoint is necessary.

Sending commands - Unity to ROS: once connected, Unity sends joint velocity commands to ROS, allowing users to control the robot manually. These commands specify the speed at which each joint should rotate rather than absolute positions, enabling smooth real-time control. However, high-speed values may cause visualization mismatches, as Unity’s rendering might not perfectly align with ROS execution.

For more intuitive control, Unity also sends PoseStamped messages to define the robot’s end-effector position. MoveIt! uses IK to compute joint movements to match this target. While this enables smooth manipulation, certain targets may be unreachable due to kinematic singularities. If the robot fails to move, switching back to joint control and resetting positions can resolve the issue.

Feedback from ROS to Unity - synchronizing the DT: for accurate AR visualization, Unity continuously receives joint position updates from ROS. These messages, sent as Float64MultiArray or JointState messages, ensure Unity’s DT reflects the robot’s actual movements. Without this feedback, Unity would display outdated robot positions, causing visual drift.

The update frequency varies between 10Hz and 50Hz, balancing real-time accuracy with network efficiency. Slow update rates can lead to laggy visuals, while network congestion may drop packets. If Unity stops receiving updates, restarting the ROS TCP connection or increasing the publish rate may resolve the issue.

MoveIt! and motion planning: MoveIt! is responsible for planning robot trajectories and ensuring safe motion execution. When a user moves the target sphere in tip control, Unity sends a PoseStamped message, and MoveIt! calculates an optimal path. If a valid path is found, MoveIt! sends a JointTrajectory message to ROS, which then executes the movement.

Motion planning is highly dependent on collision constraints and IK solver accuracy. In cases where MoveIt! cannot find a valid path, the robot may remain unresponsive. Users can mitigate this by avoiding extreme positions and ensuring MoveIt!’s constraints are properly configured.

Real-Time Motion with MoveIt! Servo: for reactive, continuous control, MoveIt! Servo sends high-frequency JointJog and TwistStamped messages to ROS, allowing the robot to follow dynamic inputs from Unity. These messages are sent at 100Hz, enabling smooth motion but requiring low-latency processing.

Since Servo relies on continuous updates, network instability or packet loss can cause erratic movements. To prevent instability, Unity filters input values and MoveIt! applies motion constraints. If sudden movement stops occurring, restarting the MoveIt! Servo node can restore functionality.

3.4 Digital Twin - Environmental IoT Devices

The Environmental IoT (EIoT) Devices DT comprises several IoT ambiental sensors working as standalone devices within a network. Those sensors are responsible of ensuring the optimal conditions in the space by alerting administrators and adjusting actuators devices.

3.4.1 EIoT Devices DT – Physical infrastructure

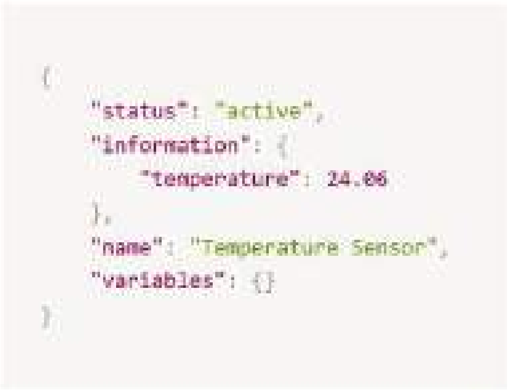
The physical system is composed of primarily three types of devices with specific purposes: environmental sensors, user input sensors, and actuators. All devices own their respective address in the network and leverage HTTP communication to send and receive environmental information with the rest of devices. Furthermore, all devices are assigned a Message Queuing Telemetry Transport (MQTT) topic to track their status for DT, but which is transparent and irrelevant for the functioning of the real system itself. Table 6 presents the existing devices along with their specific address, topic and descriptions.

Table 6. Devices characteristics

Device	Purpose	Type	HTTP	IP Address	MQTT Topic	Valid states
touch1	Emulate fingerprint recognition for the door	User input	GET	`10.0.0.102`	`sensors/touch1`	1 (recognized), 0 (standby)
touch2	Emulate fingerprint recognition for the cabinet	User input	GET	`10.0.0.109`	`sensors/touch2`	1 (recognised), 0 (standby)
button1	UP	User input	GET	`10.0.0.104`	`sensors/button1`	1 (pressed), 0 (released)

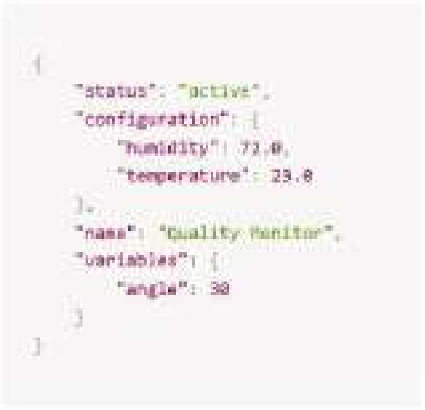
button2	DOWN	User input	GET	Shares endpoint with button1	`sensors/button2`	1 (pressed), 0 (released)
movement 1	PIR sensor for lights	User input	GET	`10.0.0.108`	`sensors/movement1`	1 (motion), 0 (no motion)
potentiome ter1	Lights’ brightness	User input	GET	`10.0.0.110`	`sensors/potentiom eter1`	0-1000
potentiome ter2	Lamp’s brightness	User input	GET	`10.0.0.111`	`sensors/potentiom eter2`	0-1000
led1	Lights	Actuator	GET / POST	`10.0.0.103`	`actuators/led1`	0-255
led2	Lamp	Actuator	GET / POST	`10.0.0.105`	`actuators/led2`	0-255
led3	Alerts	Actuator	GET / POST	`10.0.0.106`	`actuators/led3`	0-255
servo1	Motorised component	Actuator	GET / POST	`10.0.0.122`	`actuators/servo1`	0, 90, 155
servo2	Motorised component	Actuator	GET / POST	`10.0.0.123`	`actuators/servo2`	0, 35, 75
servo3	Air quality monitor	Actuator	GET / POST	`10.0.0.124`	`actuators/servo3`	0, 30, 60, 90, 120, 155
indicator1	Door lock	Actuator	GET / POST	`10.0.0.125`	`actuators/indicat or1`	1, 0
indicator2	Cabinet lock	Actuator	GET / POST	`10.0.0.126`	`actuators/indicat or2`	1, 0
indicator3	Temperature indicator	Actuator	GET / POST	`10.0.0.127`	`actuators/indicat or3`	1, 0
ldr1	Ambient light	Environmen tal sensor	GET	`10.0.0.112`	`sensors/ledr1`	0-1000
temperatur e1	Temperature sensor	Environmen tal sensor	GET	`10.0.0.113`	`sensors/temperatu re1`	0-1000
temperatur e2	Accurate temperature sensor	Environmen tal sensor	GET	`10.0.0.129`	`sensors/temperatu re2`	Temperature value e.g. (24.2)
air_quality1	Air quality sensor	Environmen tal sensor	GET	`10.0.0.114`	`sensors/air_quali ty1`	0-1000
gas1	Gas sensor	Environmen tal sensor	GET	`10.0.0.115`	`sensors/gas1`	0-1000
humidity1	Humidity sensor	Environmen tal sensor	GET	`10.0.0.128`	`sensors/humidity1`	Humidity percentage e.g. (60)

Regarding the communications, devices can provide two endpoints, one to retrieve device status and another to alter its configuration with the defined rules. Both endpoints use JSON format to represent status in a human comprehensible way, aiming for the understanding of the behaviour of the system. Only actuators can have their configurations modified and the only configurable part of the JSON is the information under the label “configuration”. Data, while preserving a similar structure, is adapted to each type of device in the network. Figure 4 and Figure 5 provide an example of the data retrieved in the communications with a sensor and an actuator, respectively.



```
{
    "status": "active",
    "information": {
        "temperature": 24.06
    },
    "name": "Temperature Sensor",
    "variables": {}
}
```

Figure 4. Result of a GET request



```
{
    "status": "active",
    "configuration": {
        "humidity": 72.0,
        "temperature": 23.0
    },
    "name": "Quality Monitor",
    "variables": {
        "angle": 30
    }
}
```

Figure 5. JSON configuration for the quality monitor actuator

3.4.2 EIoT Devices DT - Architecture

The architecture, shown in Figure 6, separates DT responsibilities in specific modules that work together to address not only the imitation of the real system, but also the deception and testing applications. The real system hosts genuine devices that produce data and adapt the system in consequence. More specifically:

- ***The Synchronisation Module (SYNC-M)*** stands for the communication between the real assets and their digital counterpart through a MQTT broker. It fully synchronises the status of digital assets with the real world. However, to address deception and experimentation, this module allows to produce synthetic data to add discrepancy or recreate extreme conditions in the system. This feature not only limits the exposure of the system but also facilitates testing operations and generates valuable information regarding the behaviour of the system in certain situations.
- ***The Digital Network Module (DNET-M)***, in contrast to the real system network, it holds the virtualised versions of each sensor and actuator on the real counterpart and communicates with SYNC-M to extract the input reads and update the result actions in the actuators. To flawlessly imitate the functioning of the system, the module replicates device logic, providing deterministic outputs for the given inputs.
- ***The Administration Module (ADMIN-M)*** comprises the set of actions that the system executes under the supervisor guidance. The supervisor is able to configure the criticality settings through the Recommendation Module (REC-M), specifying in the system which actions can be triggered automatically to protect the real part, and which must be suggested to operators, thus requiring human intervention for the execution. The supervisor is also responsible for configuring the synchronisation of real assets to address deception and testing applications within the environment. Additionally, this module gathers all the information from SYNC-

M and DNET-M, offering a graphical visualisation of the status of the system in the GUI Module (GUI-M) and the distribution of DT traces to the Monitoring and Analytics Tool through Normalisation Module (NORM-M). Finally, attacks generated by the weaponizer are also handled in ADMIN-M, which applies the necessary synchronisation settings and delivers the attacks directly to the digital network.

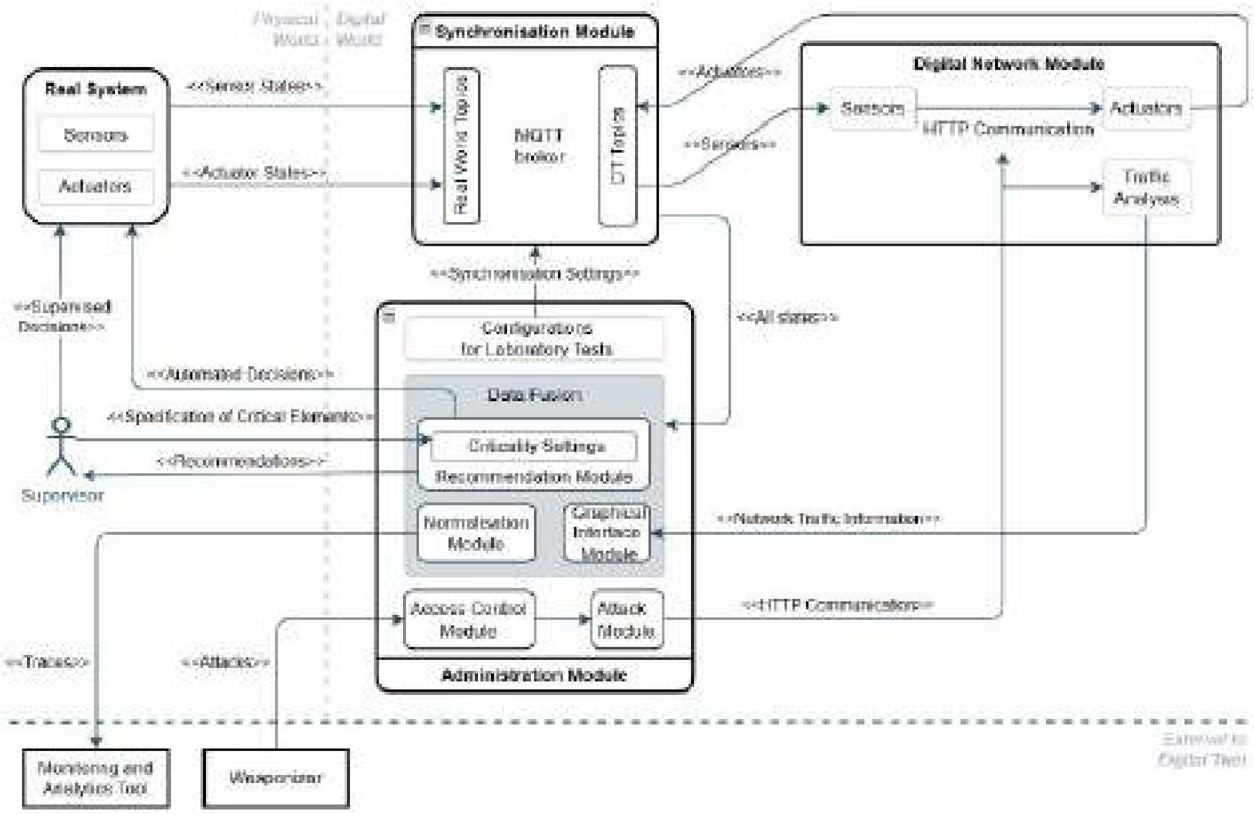


Figure 6. Architecture of the EIoT Devices DT

3.4.3 EIoT Devices DT – Logic

The intrinsic logic of this approach orchestrates the collaboration of all modules to ensure the capabilities for an effective imitation and DT-driven deception. The following lines delve into the functioning of the DT, matching the requirements specified D2.1.

Real world devices leverage the connection with SYNC-M to keep track of real assets states through topics in the MQTT broker. These topics are delivered to the DNET-M for further imitation of the devices according to the synchronisation settings defined by the operators (REQ-DECEPTION-NFR-9). It is a supervisor’s responsibility to set which sensor’s states are synchronised and which are simulated (USR-004) by either using generation functions or more complex AI-driven systems (REQ-DECEPTION-DEC-8). This feature enables the deception capabilities of the system (REQ-DECEPTION-DEC-2) and the simulation of experimental or extreme circumstances for further adjustment of the real system without a direct impact on it. In contrast, SYNC-M is not intended to synchronise actuator’s states as they must be computed by the internal logic of the digital system and match the ones computed in the real system, enabling for the comparison of real and digital systems behaviour towards the detection of anomalies or discrepancies.

Devices running in DNET-M employ sensor states in SYNC-M, either captured from real system or generated, to execute the logic of the system and provide the necessary outputs. The network consists of a virtual replica of the original, maintaining the exact structure, protocols and communication formats (REQ-DECEPTION-DEC-1), allowing for a seamlessly monitoring of the traffic.

Lastly, ADMIN-M concentrates all administrative tasks in the DT. Leveraging connections with both SYNC-M and DNET-M, this module reunites all the contextual information of the system. Primarily, ADMIN-M facilitates the understanding of system status through a user-friendly GUI-M that not only show real-time information of the environment but also provide alerts and permits supervisors to take proactive decisions (REQ-DECEPTION-NFR-10). REC-M help automatising those actions by providing faster and more effective responses against disturbances with the implementation of AI and ML driven decision-making (USR-006). Nonetheless, due to the critical environment of application, the supervisor must specify in the criticality setting which actions can be automatically triggered against the real system, and which must be only alerted, protecting the system from the negative impact that this decision can have under critical components, compromising user’s welfare.

This approach also concentrates the efforts of the AIAS infrastructure, providing connections with other pivotal modules such as the Monitoring and Analytics Tool. The system can be easily adapted to normalise extracted data from the behaviour of the system in order to be analysed by this component. Additionally, the possibility of using the virtual system as a testing environment enables the communication with the weaponizer module.

3.5 Virtual Persona – Hospital Monitoring

The hospital monitoring VP system enables the simulation of key healthcare actors to observe, adjust, and enhance the environmental conditions within critical infrastructures such as hospitals.

3.5.1 Virtual Persona System Architecture and Functional Modules

The design of the VP system includes a set of specific components that use advanced technology to facilitate intelligent, context-sensitive interactions in a dynamic environment. The Sensor Data module functions as a main input layer, collecting environmental and contextual information from internal and external sources, as illustrated in the diagram in Figure 7. This data directly informs the Anomaly Detection component, which applies ML methods to detect outliers and other operational issues. The Logging module systematically records all operations and data transfers, thereby ensuring traceability and supporting future audits or analyses. The Response Generation module relies on advanced language models to generate dependent on context, role-specific reactions, enabling effortless communication. The MQTT Publisher allows efficient, low-latency communication with downstream systems or IoT devices through the transmission of outputs through the MQTT protocol.

These elements collectively develop an effective basis for the development of intelligent, interactive agents in areas such as healthcare, smart environments, and industrial automation.

Table 7 outlines the essential elements of this system, detailing their technological bases and operational roles. These aspects provide real-time data collection, advanced anomaly detection, exact system recording, context-driven response generation, and effective interaction with other systems through standardized communication protocols.

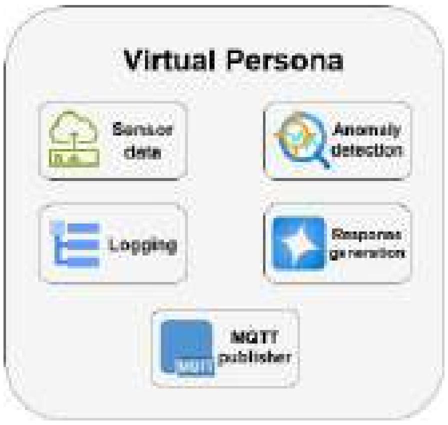


Figure 7. VP’s modules

Table 7. VP’s components

Component	Technology used	Description and functional role
Sensor data	requests (Python module)	Responsible for obtaining real-time environmental and contextual input from external APIs or internal endpoints. This data forms the foundation for further processing
Anomaly detection	scikit-learn (Isolation Forest)	Applies Isolation Forests to unsupervised anomaly detection. It detects variations from predicted operational trends in the sensor data, therefore providing early warnings
Logging	logging, Comma-Separated Values (CSV), JSON	Records all actions, AI answers and sensor values to provide systematic event monitoring and data durability. It allows auditing and retrospective analysis
Response generation	OpenAI GPT (gpt-3.5-turbo)	Using contextual sensor data as input, dynamically creates natural language responses for specific roles (such as nurse, medical doctor). It supports human-computer interaction
MQTT publisher	paho-mqtt (MQTT v3.1.1)	It publishes system outputs (e.g. alerts, role-based responses) over MQTT protocol, ensuring low-latency communication with downstream systems or IoT components

3.5.2 Data Interchange Between Digital Twin and Virtual Persona

The architecture for data interchange between the DT and the VP is designed to facilitate safe, real-time monitoring and the generation of intelligent responses within a hospital environment. DT component provides a RESTful API endpoint (/sensor_data) which VP accesses at regular intervals (e.g. every 10 seconds) to obtain current sensor data in organized JSON format. Table 8 includes the data exchange between DT and VP.

Table 8. Data exchange between DT & VP

Direction	Data type	Message name / Endpoint	Size	Frequency	Description

DT → VP	JSON (HTTP REST)	GET /sensor_data	Medium (~300–600 bytes)	Every 10 seconds	VP pulls current sensor values (temperature, humidity, CO_2, air quality, motion, etc.)
VP → MQTT Broker	JSON (MQTT)	hospital/alerts	Small–Medium (128–512 B)	On anomaly detection	VP publishes alerts when thresholds are crossed, or anomalies are detected
VP → MQTT Broker	JSON (MQTT)	hospital/respon se	Medium (~512–1024 B)	On AI response	VP publishes AI-generated responses per role (doctor, nurse, patient, IT admin)
VP → File System	CSV / NDJSON	log_YYYY-MM-DD.csv/.json	Variable (1 row/line)	Every 10 seconds	Logs complete sensor data, warnings, and GPT response into daily log files
VP → JSON Formatter	JSON Array	log_YYYY-MM-DD_fixed.json	Medium (~1–3 KB total)	After each cycle	Converts line-based JSON to valid JSON array for external tools / dashboards
VP → OpenAI API	HTTPS request	POST /v1/chat/compl etions	~1–2 KB round-trip	On each sensor cycle	Sends role-specific prompt and receives GPT-generated natural language response

This information covers contextual and environmental factors such temperature, humidity, CO_2 levels, motion detection, occupancy. After gathering the data, VP assesses it using a dual-layered logic: initially by using static threshold criteria and then using a ML-based anomaly detection model (Isolation Forest). When a deviation or significant event occurs, MQTT sends an alarm to the hospital/alerts topic, therefore ensuring low-latency delivery to subscribing systems. Simultaneously a request is built and presented to the OpenAI GPT API according on the specified user role: doctor, patient, IT administrator, nurse to offer a contextual natural language answer.

This answer then is forwarded to the hospital/response and concurrently archived with sensor readings in organized daily log files (.csv and .json). Newline-delimited JSON logs are automatically converted into valid JSON arrays (*_fixed .json) for use in other systems, including dashboards, analytics pipelines or APIs, so improving interoperability. This architecture provides scalability, auditability, and adaptability inside a healthcare DT environment, therefore supporting intelligent interaction and operational continuity.

3.5.3 Connectivity and I/O Architecture in the Virtual Persona Framework

This section outlines the key interface types and communication mechanisms that facilitate data exchange between the VP and its surrounding system components. These interfaces provide connection with DT, external AI services, distributed messaging systems and logging frameworks. Each interface serves a particular function, ranging from collecting real-time sensor data and generating AI-driven replies to publishing alarms and maintaining activity logs, guaranteeing that the system is responsive, scalable, and auditable across different operating scenarios. Table 9 covers the interface types and communication mechanism.

Table 9. Interface types and communication mechanism

Interface type	Role in system	Key features

REST API (DT)	Provides real-time virtualized sensor data to VP	JSON-based, modular abstraction
HTTPS API (GPT)	Enables external cognitive processing via OpenAI LLM	Context-aware, dynamic language response
MQTT messaging	Publishes alerts and responses asynchronously to distributed systems	Lightweight, real-time, scalable
File-based logging	Ensures traceable, structured persistence of system activity	Dual format (CSV + JSON), machine-readable
CLI (Command-Line Interface)	Captures user role input and provides immediate system feedback	Lightweight, extensible, scenario-driven

3.5.3.1 Digital Twin REST API interface

The RESTful interface offered by the DT is the primary ingress point for real-time environmental data into the VP system. Applied as an HTTP GET endpoint (/sensor_data), it gathers the virtualized state of hospital-environment physical sensors. The payload is a flat JSON object based on important measures like temperature, humidity, CO_2 concentration, other environmental and contextual data. This interface guarantees that the AI engine works on current and accurate sensor information by letting the VP module regularly poll, therefore acting as a synchronous data collecting layer.

This interface isolates physical sensor complexity by providing a uniform semantic representation over HTTP, hence enhancing modularity and facilitating plug-in and play integration. Drawing on stateless REST concepts, it promotes decoupling and system scalability.

The RESTful interface the DT offers is the major ingress point for real-time environmental data into the VP system. Applied as an HTTP GET endpoint (/sensor_data), it gathers the virtualized state of hospital-environment physical sensors. The payload is a flat JSON object based on important measures such as temperature, humidity, CO_2 concentration, other environmental and contextual data. This interface guarantees that the AI engine works on current and accurate sensor information by letting the VP module regularly poll, therefore acting as a synchronous data collecting layer. This interface isolates physical sensor complexity by providing a uniform semantic representation over HTTP, hence enhancing modularity and facilitating plug-in and play integration. Stressing decoupling and system scalability, it uses stateless REST concepts.

3.5.3.2 OpenAI GPT API interface

The fundamental AI interface for natural language interpretation and creation is interaction with the OpenAI API. HTTPS POST requests activate this interface, which uses OpenAI Chat Completions API standard for payload structure. VP generates role-specific prompts tailored for each device enhanced with real-time sensor data and sends them to the GPT engine. The returned material is a natural language response with context founded on roles.

Large-scale pretraining and few-shot reasoning advantages occur when externalizing natural language processing to a foundation model (i.e. a large language model such as GPT-4 or LLaMA). This modular method separates the cognitive layer from sensor logic such that it guarantees adaptation to changing language models or local LLMs with refined medical information.

Natural language understanding and generation’s fundamental AI interface is the interaction with the OpenAI API.

HTTPS POST requests activate this interface, which uses OpenAI Chat Completions API standard for payload structure. The VP generates role-specific prompts tailored for each device enhanced with real-time sensor data and sends them to the GPT engine. The returned content is a natural language response with context founded on roles.

The system gains few-shot reasoning and large-scale pretraining by externalizing natural language processing to a foundation model (i.e. a large language model such as GPT-4, LLaMA or Med-PaLM). This modular method separates the cognitive layer from sensor logic such that it guarantees adaptation to changing language models or local LLMs with refined medical information.

3.5.3.3 MQTT messaging interface

MQTT is the basis of the asynchronous messaging architecture of the system. VP delivers to the MQTT broker two types of messages:

- ***Alerts (hospital/alerts)***: initiated through rule-based or ML-driven anomaly detection.
- ***Responses (hospital/response)***: outputs of GPT adapted to specific roles.

The MQTT protocol facilitates efficient, low-latency communication, making it appropriate for resource-constrained or real-time healthcare environments. Edge computing and IoT both have a generally agreed upon benchmark in MQTT. The low overhead and support for Quality of Service (QoS) levels make it appropriate for healthcare environments, where the quick warning and reaction delivery is crucial. This interface lets system components or clients dynamically subscribe to or reply to relevant topics, hence facilitating distributed scalability.

3.5.3.4 File-based logging interface

The file I/O interface is responsible for structured, persistent storage of sensor data, system state, and AI-generated interactions. Each monitoring cycle appends a new row or line to:

- A daily .csv file (tabular format for analysts).
- A Newline Delimited .json file (NDJSON for machines). In addition, each .json log is converted into a syntactically valid JSON array (*_fixed.json) to enable use with APIs, dashboards and downstream applications.

This hybrid logging method helps to provide audit trails with machine-processability as well as human-readable capability. Using NDJSON helps stream data, converting it into appropriate JSON arrays improves front-end component and API compatibility. Timestamps and Universal Unique Identifier (UUID)-based event IDs provide traceability, allow result replication, and assist retroactive forensic investigation.

3.5.3.5 CLI-based user interaction interface

The Command Line Interface (CLI) is mainly used for selecting user roles and providing real-time system feedback. Users detect their functional role (medical doctor, nurse, patient, Information Technology (IT) administrator) at the beginning, facilitating the customization of AI interactions and scenario simulations. The console continuously displays responses generated by AI, alert statuses, and sensor readings. Though basic, in VMs or headless environments the CLI provides a lightweight control plane ideal for development, testing, and validation. In production environments, it provides consistent user input routes and may be improved or altered via graphical interfaces or APIs. Its integration ensures incredibly important role-adaptive system behaviour, which is necessary for automated adapted healthcare.

3.6 Architecture of the Monitoring and Security Analytics

The AIAS Monitoring and Security Analytics Tool (MAT) functions as part of the AIAS deception layer. While the deception layer is responsible to detect and track activities tailored to adversarial activity at AI systems, the monitoring and security analytics tool is responsible to analyse them and extract valuable insights. Through collection and processing next to visualization of essential data points, organizations receive actionable intelligence to reinforce cyber defences. The AIAS tool differs from standard network monitors as it prioritizes log management along with correlation instead of direct threat detection. ELK powers the tool to deliver real time knowledge about adversary behaviour.

Monitoring tools, are vital in the process of tracking progress, anticipating and solving problems. They continuously monitor and control IT infrastructures, applications and occasionally networks to ensure the availability, performance and reliability of IT systems, focusing on the early detection of problems or attacks that may occur or have already occurred. The most important part of the process of solving a problem is identifying it and the AIAS monitoring and security analytics tool is the key to identify and track the possible adversarial activity.

The AIAS MAT collects and analyses data focused on three sources:

- ***VPs***: simulated digital identities created to interact with adversaries in the cyber environment. These personas are the ones responsible to mimic real user behaviours and characteristics to engage potential attackers and gather intelligence on their tactics, techniques, and procedures tailored to cyber incidents. In the AIAS MAT tool, VPs are used to identify and understand adversarial activities by observing how attackers interact with these fabricated identities, and their output is the input for the monitoring tool.
- ***DTs***: precise digital replicas of physical systems or processes. In the process of monitoring and analysing the behaviour of these digital replicas, the AIAS tool can detect anomalies, potential security threats, and assess the impact of adversarial actions on the actual systems. This allows organizations to proactively address vulnerabilities.
- ***HIHs***: decoy systems designed to engage and interact with attackers in a realistic manner but certainly controlled. Unlike low interaction honeypots that simulate basic services and are easy to identify, HIHs mimic entire systems, providing attackers with an authentic environment to exploit. In the AIAS tool, HIHs output is used to monitor the methods and objectives of the attackers, allowing organizations to collect valuable intelligence and improve their understanding of emerging threats.

3.6.1 Workflow Architecture

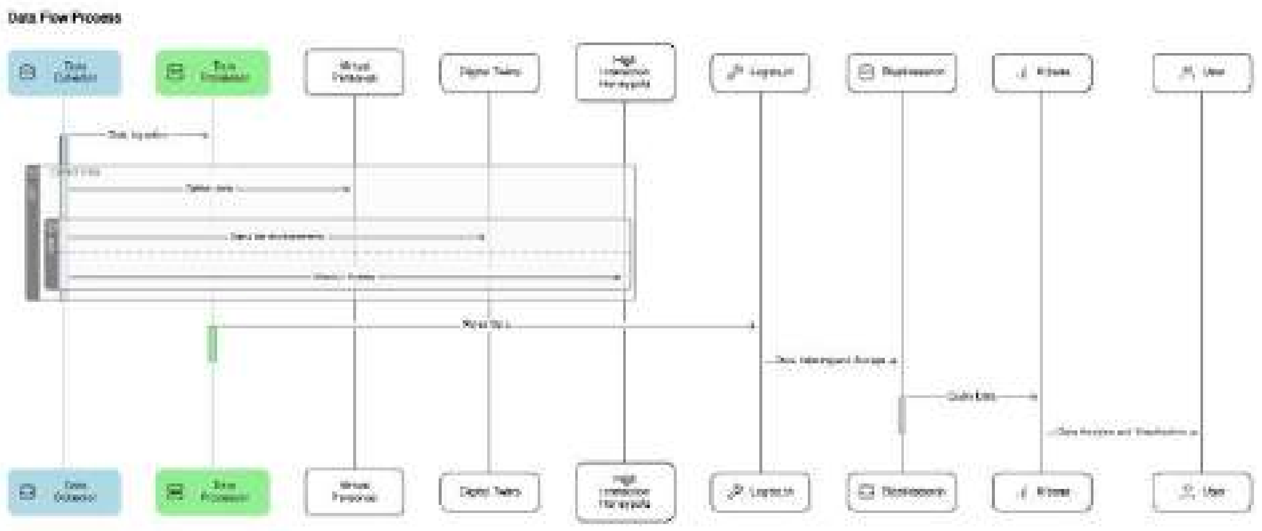


Figure 8. Process flow of AIAS MAT

To receive the proper associated data, the MAT architecture needs other components to interact with each other. Below there is a detailed description of how the MAT architecture is operating in the context of data collection through the end that is the visualization, with the required contribution from each component. Figure 8 depicts the AIAS MAT working process.

The AIAS MAT key functionalities are the following:

- ***Data collection***: at the first stage, we have the data collector which is responsible for retrieving real-time data from various deception mechanisms, such as mentioned DTs, VPs, and HIHs. Data Collector presents results as raw data which are the primary data collected from a source or filtered so they are organized, user-friendly and more eligible than the raw data, usually in a JSON or CSV format.
- ***Data processing***: a standard template organizes all information systematically. Logstash converts diverse log formats into structured records. The analyser adds extra details such as timestamps, geographic positions if provided, as well as system data for better evaluation.
- ***Data indexing and storage***: it is responsible for the management of massive amounts of data in a scalable way that satisfies the needs of applications that require fast access to specific dataset. ML models and statistical analysis are employed to extract patterns and data from attack interactions. Automated tagging of noteworthy events further aids forensic analysis and retrospective examination.
- ***Incident reporting and visualization****:* data visualization enhances the data analysis aiming at discovering useful information, information conclusions and supporting decision-making. Detailed reports summarizing key adversarial interactions can also be generated, enabling decision-makers to alter security policies.

3.6.2 Architecture Components

AIAS MAT architecture (see Figure 9) is designed to ensure scalability, efficiency, and seamless integration with the

AIAS Deception Layer. The key components will be analysed, for the best understanding on how they work and what is their use.

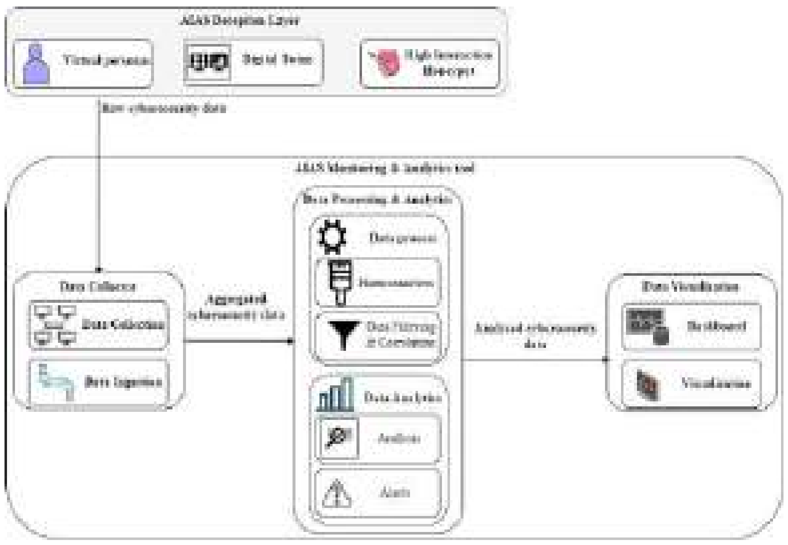


Figure 9. MAT blueprint

- ***Data collector:*** this component gathers log data from deception tools like honeypots, DTs, and VPs. Then a log analyser tracks hostile actions, attack rates, behaviour models plus weak points that attackers target and helps monitor them. ELK’s RESTful APIs enable automatic data feeds, while custom pipelines (based on Python) integrate specialized information sources that can be ingested manually. Our tool mainly focuses on CSV file ingestion that is given as input. Data collector serves as the entry point of the raw data into our tool. The responsibilities of the data collector include the access and the extraction of data from the AIAS deception layer.
- ***Data processing and analytics:*** this component is responsible for receiving data from various sources in different form, harmonizing them and then send them to the storage place. At the Data Processor, which uses Logstash for data transformation and enrichment at will, with Python scripts managing custom parsing. A standard template organizes all information systematically. Logstash converts diverse log formats into structured JSON formats for efficient indexing. The analyser adds extra details such as timestamps, geographic positions if provided, as well as system data for better evaluation
- ***Data visualization entity:*** is responsible for visualizing the transformed data. ES serves as the core data repository, ensuring efficient indexing and querying of logs while sorting each index by name and date. Its scalable storage accommodates large volumes of security logs up to 500MB. Next ES sends the data to Kibana, the technology which enables real-time log analysis and visualization, offering custom dashboards for various security use cases and suitable categorization.
- ***Users:*** this component can create custom queries and filters to drill down into specific events and needs. ML models and statistical analysis are employed to extract patterns and data from attack interactions. Automated tagging of remarkable events further aids forensic analysis and retrospective examination. These real-time dashboards in Kibana provide interactive visualizations of log data, including time-series graphs, heat maps,

and event correlation diagrams. That flexibility Kibana is offering can aid every need and help teams to filter and analyse logs using predefined queries and thresholds. Detailed reports summarizing key adversarial interactions can also be generated, enabling decision-makers to alter security policies.

4 Development of the Deception Layer

This section covers the development phases of the AIAS deception layer, detailing the design of deception components alongside the monitoring and analytics tool, as well as their integration with the distinct deception implementations within the layer.

4.1 Specific Proofs of Concept of Components for Deception

The following sections highlight specific proof-of-concept implementations for the deception components of the AIAS deception layer, offering a comprehensive overview of their functionality and how they contribute to enhance the layer's deception capabilities.

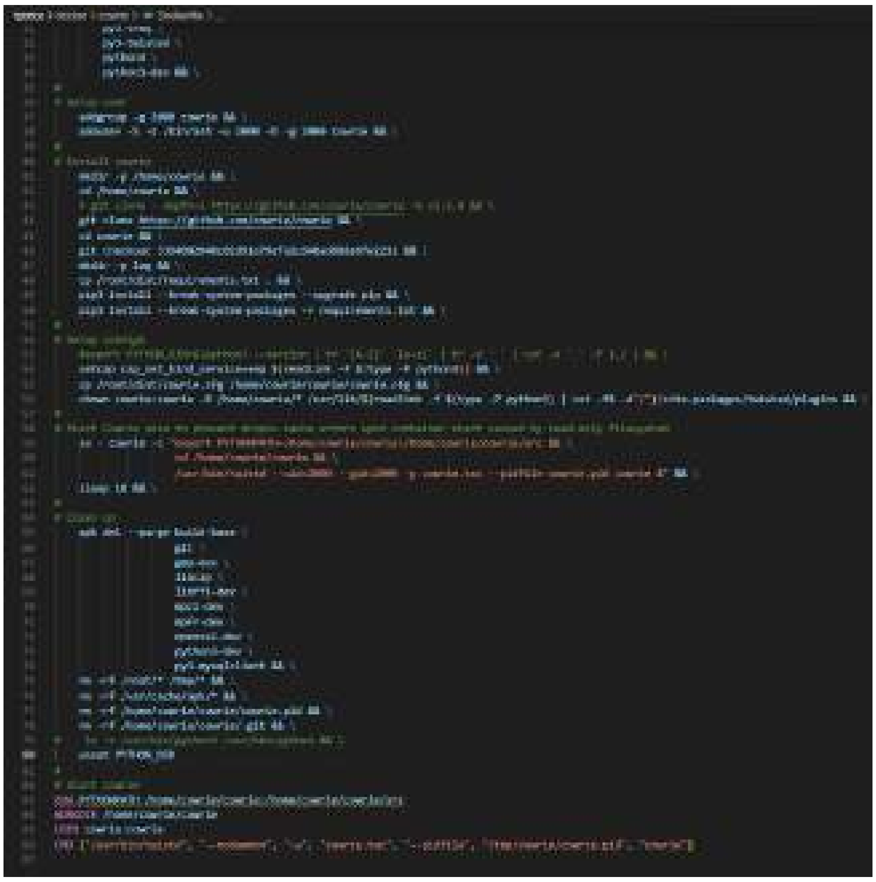
4.1.1 Honeypot

For the implementation phase of honeypots in AIAS, T-Pot platform has been deployed on a VM hosted under a VPN connection, thus ensuring a secure and controlled environment. VM is configured with the following specifications: Linux-based operating system, dual-core CPU, 16GB of RAM, and 128GB of hard disk storage. The tool was downloaded and installed from the official T-Pot repository [TPO], where comprehensive documentation and technical details are available. For the concerned purpose the "T-Pot Standard / Hive all services" version was deployed, installing all services, tools, and honeypots within a single host acting simultaneously as a Hive endpoint, a centralized incident response and threat analysis platform integrated in the T-Pot ecosystem.

After the successful deployment, T-Pot environment automatically configures over 20 honeypots. However, for the specific use case described in deliverable D2.2, the focus is exclusively on four of them: Cowrie, Mailoney, Wordpot, and Dionaea; each of them selected based on prior in-depth analysis and relevance to the defined scenarios.

T-Pot orchestrates honeypots by using containerized Docker-based architecture, ensuring isolated and modular deployments. Moreover, each honeypot is composed of three key structural elements:

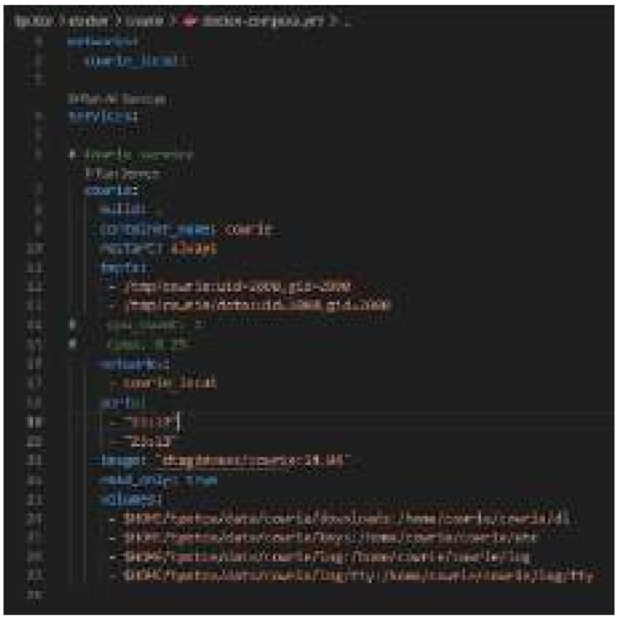
- ***Dockerfile:*** file that defines how the honeypot container is built, specifying the base docker image, necessary dependencies, file structure, exposed ports, and commands executed at runtime. T-Pot utilizes a Dockerfile similar to the example in Figure 10 to generate the corresponding Docker image for each honeypot.
- ***Docker-compose file:*** file that manages the deployment of the container, including the network configuration, volume mappings, log directories, and service dependencies (see Figure 11). This file plays a critical role in integrating the honeypot into the broad T-Pot ecosystem, including the interaction with Logstash and other honeypots. T-Pot also provides a global docker-compose file that defines these settings collectively for all services and honeypots, significantly simplifying the orchestration process.
- ***Python file:*** script containing the actual honeypot logic and behaviour such as how it listens for incoming connections, records actions, simulates responses, and logs events. (e.g. Code in Figure 12 to start ssh services).



tpotce > docker > cowrie > Dockerfile > ...

```
            py3-treq \
            py3-twisted \
            python3 \
            python3-dev && \
#
# Setup user
    addgroup -g 2000 cowrie && \
    adduser -S -s /bin/ash -u 2000 -D -g 2000 cowrie && \
#
# Install cowrie
    mkdir -p /home/cowrie && \
    cd /home/cowrie && \
#    git clone --depth=1 https://github.com/cowrie/cowrie -b v2.5.0 && \
    git clone https://github.com/cowrie/cowrie && \
    cd cowrie && \
    git checkout [illegible] && \
    mkdir -p log && \
    cp /root/dist/requirements.txt . && \
    pip3 install --break-system-packages --upgrade pip && \
    pip3 install --break-system-packages -r requirements.txt && \
#
# Setup configs
    export PYTHON_DIR=$(python3 --version | tr '[A-Z]' '[a-z]' | tr -d ' ' | cut -d '.' -f 1,2 | tr -d ' ') && \
    setcap cap_net_bind_service=+ep $(readlink -f $(type -P python3)) && \
    cp /root/dist/cowrie.cfg /home/cowrie/cowrie/cowrie.cfg && \
    chown cowrie:cowrie -R /home/cowrie/* /usr/lib/$(readlink -f $(type -P python3) | cut -f4 -d"/")/site-packages/twisted/plugins && \
#
# Start Cowrie once to prevent dropin.cache errors upon container start caused by read-only filesystem
    su - cowrie -c "export PYTHONPATH=/home/cowrie/cowrie:/home/cowrie/cowrie/src && \
                    cd /home/cowrie/cowrie && \
                    /usr/bin/twistd --uid=2000 --gid=2000 -y cowrie.tac --pidfile cowrie.pid cowrie &" && \
    sleep 10 && \
#
# Clean up
    apk del --purge build-base \
                    git \
                    gmp-dev \
                    libcap \
                    libffi-dev \
                    mpc1-dev \
                    mpfr-dev \
                    openssl-dev \
                    python3-dev \
                    py3-mysqlclient && \
    rm -rf /root/* /tmp/* && \
    rm -rf /var/cache/apk/* && \
    rm -rf /home/cowrie/cowrie/cowrie.pid && \
    rm -rf /home/cowrie/cowrie/.git && \
#    ln -s /usr/bin/python3 /usr/bin/python && \
    unset PYTHON_DIR
#
# Start cowrie
ENV PYTHONPATH /home/cowrie/cowrie:/home/cowrie/cowrie/src
WORKDIR /home/cowrie/cowrie
USER cowrie:cowrie
CMD ["/usr/bin/twistd", "--nodaemon", "-y", "cowrie.tac", "--pidfile", "/tmp/cowrie/cowrie.pid", "cowrie"]
```

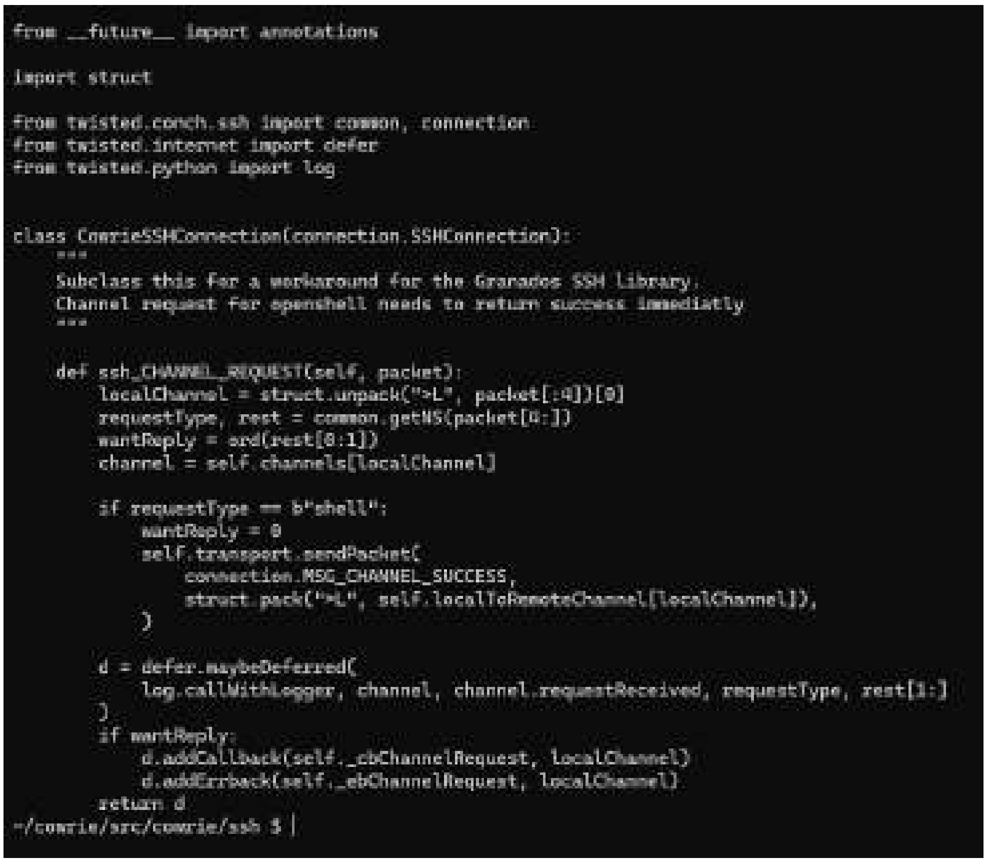
Figure 10. Dockerfile for T-Pot



tpotce > docker > cowrie > docker-compose.yml > ...

```
networks:
  cowrie_local:

# Cowrie service
services:

# Cowrie service
  # Cowrie service
  cowrie:
    build: .
    container_name: cowrie
    restart: always
    tmpfs:
     - /tmp/cowrie:uid=2000,gid=2000
     - /tmp/cowrie/data:uid=2000,gid=2000
#    cpu_count: 1
#    cpus: 0.25
    networks:
     - cowrie_local
    ports:
     - "22:22"
     - "23:23"
    image: "dtagdevsec/cowrie:24.04"
    read_only: true
    volumes:
     - $HOME/tpotce/data/cowrie/downloads:/home/cowrie/cowrie/dl
     - $HOME/tpotce/data/cowrie/keys:/home/cowrie/cowrie/etc
     - $HOME/tpotce/data/cowrie/log:/home/cowrie/cowrie/log
     - $HOME/tpotce/data/cowrie/log/tty:/home/cowrie/cowrie/log/tty
```

Figure 11. Honeypot's docker-compose file

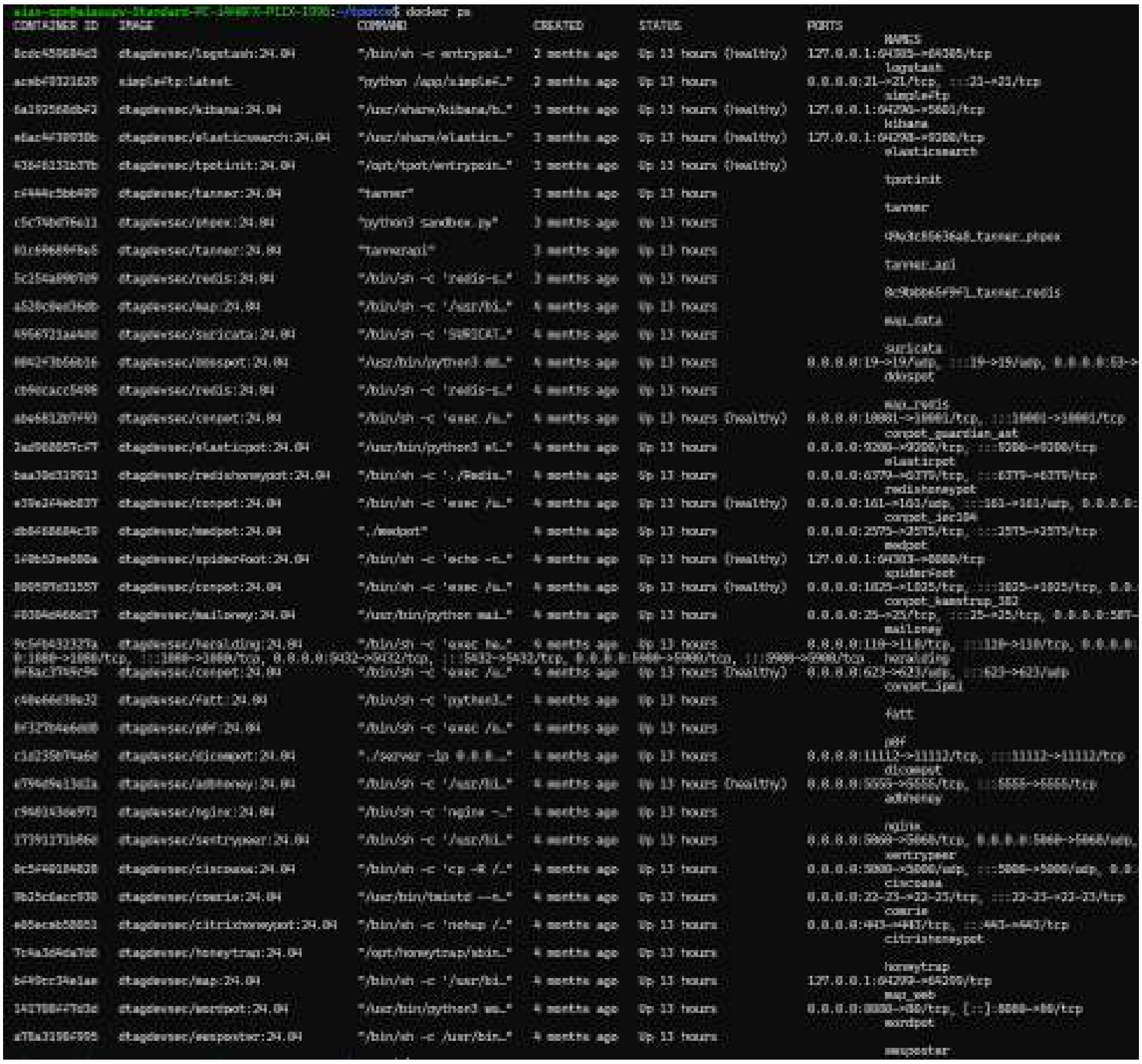


```
from __future__ import annotations

import struct

from twisted.conch.ssh import common, connection
from twisted.internet import defer
from twisted.python import log


class CowrieSSHConnection(connection.SSHConnection):
    """
    Subclass this for a workaround for the Granados SSH library.
    Channel request for openshell needs to return success immediatly
    """

    def ssh_CHANNEL_REQUEST(self, packet):
        localChannel = struct.unpack(">L", packet[:4])[0]
        requestType, rest = common.getNS(packet[4:])
        wantReply = ord(rest[0:1])
        channel = self.channels[localChannel]

        if requestType == b"shell":
            wantReply = 0
            self.transport.sendPacket(
                connection.MSG_CHANNEL_SUCCESS,
                struct.pack(">L", self.localToRemoteChannel[localChannel]),
            )

        d = defer.maybeDeferred(
            log.callWithLogger, channel, channel.requestReceived, requestType, rest[1:]
        )
        if wantReply:
            d.addCallback(self._cbChannelRequest, localChannel)
            d.addErrback(self._ebChannelRequest, localChannel)
        return d
~/cowrie/src/cowrie/ssh $ |
```

Figure 12. Honeypot's Python script

In summary, the interaction flow between the three core components can be described as follows: (i) cowrie.py contains the main logic and code of the honeypot; (ii) the Dockerfile packages the code into a Docker image; and (iii) docker-compose file orchestrates the deployment within the T-Pot ecosystem, enabling port redirection, log persistence, and monitoring functionalities.

All these components and configuration files are provided and accessible within the T-Pot distribution. Consequently, the VM enables the access, inspection, and analysis of each of these elements in detail. To fully initialise the tool and activate all available honeypots, the general docker-compose file of T-Pot must be executed using the command “*docker-compose up -d”*. The status of running containers and services can be verified via “*docker ps”*, which lists all active honeypots and Docker components of the T-Pot platform as visualised in.



```
[illegible]:~/tpotce$ docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED        STATUS                   PORTS
                                                                                                                                   NAMES
8cdc45968 4d3  dtagdevsec/logstash:24.04          "/bin/sh -c entrypoi…"   3 months ago   Up 13 hours (healthy)    127.0.0.1:64305->64305/tcp
                                                                                                                                   logstash
acab49321620   simpleftp:latest                   "python /app/simplef…"   3 months ago   Up 13 hours              0.0.0.0:21->21/tcp, :::21->21/tcp
                                                                                                                                   simpleftp
6a192568db42   dtagdevsec/kibana:24.04            "/usr/share/kibana/b…"   3 months ago   Up 13 hours (healthy)    127.0.0.1:64296->5601/tcp
                                                                                                                                   kibana
e6ac4f30038b   dtagdevsec/elasticsearch:24.04     "/usr/share/elastics…"   3 months ago   Up 13 hours (healthy)    127.0.0.1:64298->9200/tcp
                                                                                                                                   elasticsearch
43640131b37b   dtagdevsec/tpotinit:24.04          "/opt/tpot/entrypoin…"   3 months ago   Up 13 hours (healthy)
                                                                                                                                   tpotinit
c4444c5bb499   dtagdevsec/tanner:24.04            "tanner"                 3 months ago   Up 13 hours
                                                                                                                                   tanner
c5c74bd76e11   dtagdevsec/phpox:24.04             "python3 sandbox.py"     3 months ago   Up 13 hours
                                                                                                                                   09a3c8563648_tanner_phpox
81c69689f8a5   dtagdevsec/tanner:24.04            "tannerapi"              3 months ago   Up 13 hours
                                                                                                                                   tanner_api
5c254a890709   dtagdevsec/redis:24.04             "/bin/sh -c 'redis-s…"   3 months ago   Up 13 hours
                                                                                                                                   8c90b065f9f1_tanner_redis
a520c9ed36db   dtagdevsec/map:24.04               "/bin/sh -c '/usr/bi…"   4 months ago   Up 13 hours
                                                                                                                                   map_data
4956721ae480   dtagdevsec/suricata:24.04          "/bin/sh -c 'SURICAT…"   4 months ago   Up 13 hours
                                                                                                                                   suricata
8842f3b66b16   dtagdevsec/ddospot:24.04           "/usr/bin/python3 dd…"   4 months ago   Up 13 hours              0.0.0.0:19->19/udp, :::19->19/udp, 0.0.0.0:53->
                                                                                                                                   ddospot
cb90cacc5498   dtagdevsec/redis:24.04             "/bin/sh -c 'redis-s…"   4 months ago   Up 13 hours
                                                                                                                                   map_redis
abe6812b7f93   dtagdevsec/conpot:24.04            "/bin/sh -c 'exec /u…"   4 months ago   Up 13 hours (healthy)    0.0.0.0:10001->10001/tcp, :::10001->10001/tcp
                                                                                                                                   conpot_guardian_ast
3ad968057c47   dtagdevsec/elasticpot:24.04        "/usr/bin/python3 el…"   4 months ago   Up 13 hours              0.0.0.0:9200->9200/tcp, :::9200->9200/tcp
                                                                                                                                   elasticpot
baa30d310913   dtagdevsec/redishoneypot:24.04     "/bin/sh -c './Redis…"   4 months ago   Up 13 hours              0.0.0.0:6379->6379/tcp, :::6379->6379/tcp
                                                                                                                                   redishoneypot
e39e2f4eb837   dtagdevsec/conpot:24.04            "/bin/sh -c 'exec /u…"   4 months ago   Up 13 hours (healthy)    0.0.0.0:161->161/udp, :::161->161/udp, 0.0.0.0:
                                                                                                                                   conpot_iec104
db6458684c39   dtagdevsec/medpot:24.04            "./medpot"               4 months ago   Up 13 hours              0.0.0.0:2575->2575/tcp, :::2575->2575/tcp
                                                                                                                                   medpot
149b52ee880a   dtagdevsec/spiderfoot:24.04        "/bin/sh -c 'echo -n…"   4 months ago   Up 13 hours (healthy)    127.0.0.1:64303->8080/tcp
                                                                                                                                   spiderfoot
80050f9d3155 7 dtagdevsec/conpot:24.04            "/bin/sh -c 'exec /u…"   4 months ago   Up 13 hours (healthy)    0.0.0.0:1025->1025/tcp, :::1025->1025/tcp, 0.0.
                                                                                                                                   conpot_kamstrup_382
f0304d466d27   dtagdevsec/mailoney:24.04          "/usr/bin/python mai…"   4 months ago   Up 13 hours              0.0.0.0:25->25/tcp, :::25->25/tcp, 0.0.0.0:587-
                                                                                                                                   mailoney
9c5f6432327a   dtagdevsec/heralding:24.04         "/bin/sh -c 'exec he…"   4 months ago   Up 13 hours              0.0.0.0:110->110/tcp, :::110->110/tcp, 0.0.0.0:
0:1080->1080/tcp, :::1080->1080/tcp, 0.0.0.0:5432->5432/tcp, :::5432->5432/tcp, 0.0.0.0:5900->5900/tcp, :::5900->5900/tcp   heralding
0f6ac3f49c94   dtagdevsec/conpot:24.04            "/bin/sh -c 'exec /u…"   4 months ago   Up 13 hours (healthy)    0.0.0.0:623->623/udp, :::623->623/udp
                                                                                                                                   conpot_ipmi
c40e66e30e31   dtagdevsec/fatt:24.04              "/bin/sh -c 'python3…"   4 months ago   Up 13 hours
                                                                                                                                   fatt
8f327b4e6e00   dtagdevsec/p0f:24.04               "/bin/sh -c 'exec /u…"   4 months ago   Up 13 hours
                                                                                                                                   p0f
c1d2350f4a6e   dtagdevsec/dicompot:24.04          "./server -ip 0.0.0.…"   4 months ago   Up 13 hours              0.0.0.0:11112->11112/tcp, :::11112->11112/tcp
                                                                                                                                   dicompot
a794e9e13d1a   dtagdevsec/adbhoney:24.04          "/bin/sh -c '/usr/bi…"   4 months ago   Up 13 hours (healthy)    0.0.0.0:5555->5555/tcp, :::5555->5555/tcp
                                                                                                                                   adbhoney
c9401430e971   dtagdevsec/nginx:24.04             "/bin/sh -c 'nginx -…"   4 months ago   Up 13 hours
                                                                                                                                   nginx
1739127 1b86e  dtagdevsec/sentrypeer:24.04        "/bin/sh -c '/usr/bi…"   4 months ago   Up 13 hours              0.0.0.0:5060->5060/tcp, 0.0.0.0:5060->5060/udp,
                                                                                                                                   sentrypeer
0c5440184828   dtagdevsec/ciscoasa:24.04          "/bin/sh -c 'cp -R /…"   4 months ago   Up 13 hours              0.0.0.0:5000->5000/udp, :::5000->5000/udp, 0.0.
                                                                                                                                   ciscoasa
0b25c6acc930   dtagdevsec/cowrie:24.04            "/usr/bin/tmistd --n…"   4 months ago   Up 13 hours              0.0.0.0:22-23->22-23/tcp, :::22-23->22-23/tcp
                                                                                                                                   cowrie
e85ecab58851   dtagdevsec/citrixhoneypot:24.04    "/bin/sh -c 'nohup /…"   4 months ago   Up 13 hours              0.0.0.0:443->443/tcp, :::443->443/tcp
                                                                                                                                   citrixhoneypot
7c4a3d4da708   dtagdevsec/honeytrap:24.04         "/opt/honeytrap/sbin…"   4 months ago   Up 13 hours
                                                                                                                                   honeytrap
b449cc34e1ae   dtagdevsec/map:24.04               "/bin/sh -c '/usr/bi…"   4 months ago   Up 13 hours              127.0.0.1:64299->64299/tcp
                                                                                                                                   map_web
141788447d3c   dtagdevsec/wordpot:24.04           "/usr/bin/python3 wo…"   4 months ago   Up 13 hours              0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
                                                                                                                                   wordpot
a78a3106f995   dtagdevsec/ewsposter:24.04         "/bin/sh -c /usr/bin…"   4 months ago   Up 13 hours
                                                                                                                                   ewsposter
```

Figure 13. Active honeypots and Docker components of the T-Pot platform

Remote access to the T-Pot interface is also possible through a web browser by accessing “*https://<your.ip>:64297*”, where *“<your.ip>”* corresponds to the user’s IP address. By entering the credentials defined during the installation process, users are redirected to the T-Pot main dashboard shown in Figure 14.

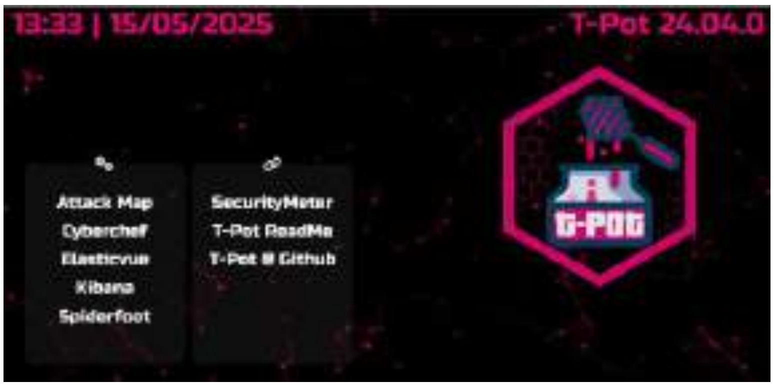


Figure 14. T-Pot main dashboard

The principal tools integrated within the T-Pot web interface include:

- ***Attack map:*** real-time visualisation of incoming attacks. This interface displays the source and destination IP addresses, geographic origin of the attacks, and targeted services or protocols being exploited (see Figure 15).

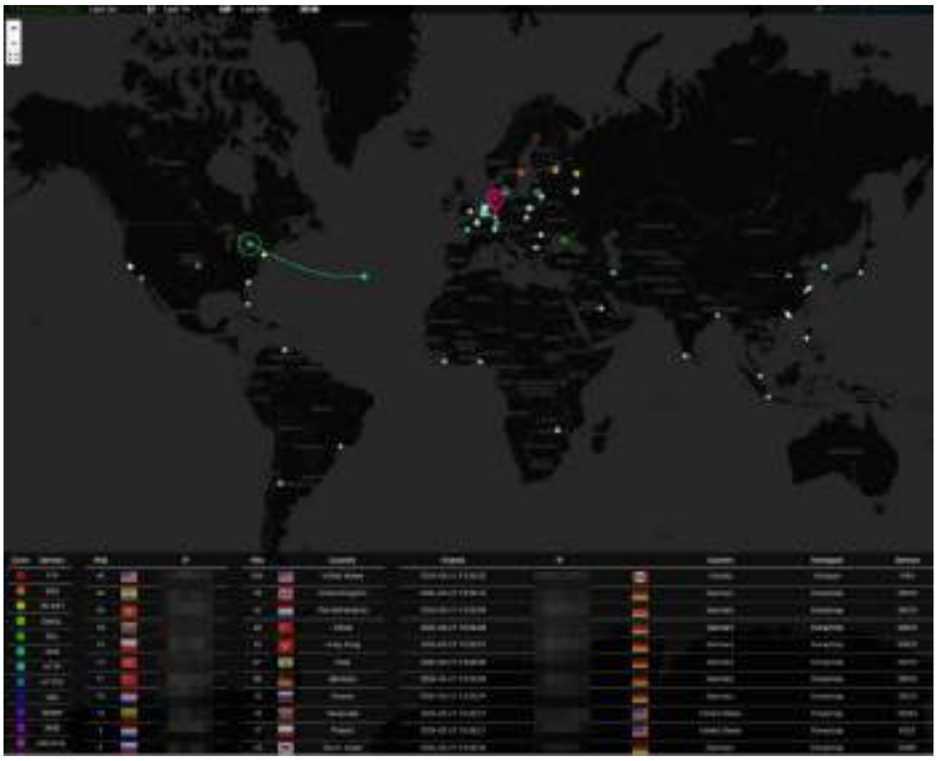


Figure 15. Attack map: real-time visualisation of incoming attacks

- ***CyberChef:*** a versatile web application for encryption, encoding, compression, and data analysis (see Figure 16). It allows users to process and manipulate data captured by the honeypots, making it an essential tool for cybersecurity tasks such as log analysis, data transformation, and decoding of malicious payloads.

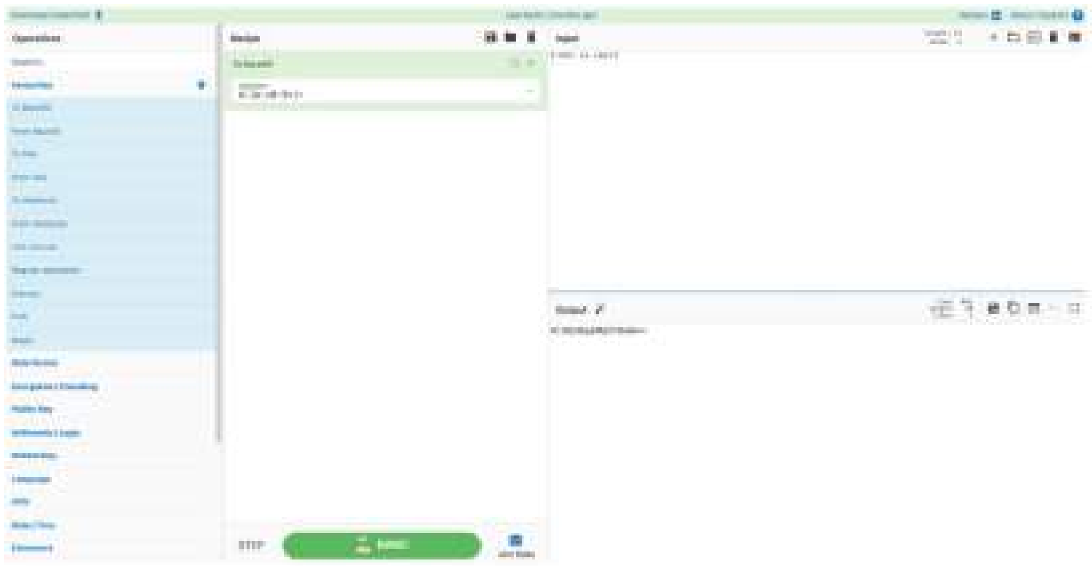


Figure 16. CyberChef’s main dashboard

- ***Elasticvue:*** graphical interface for managing Elasticsearch (ES) clusters. It facilitates data exploration, index management, and complex searches, streamlining access to the data collected by honeypots (see Figure 17).

Figure 17. Elasticvue graphical interface for managing ES clusters

- ***Kibana:*** powerful data visualization tool integrated within ES. In the scope of T-Pot, Kibana is used to generate interactive dashboards and visual representations of honeypot data, enabling users to monitor attacks in real time and perform detailed event analysis as shown in Figure 18.

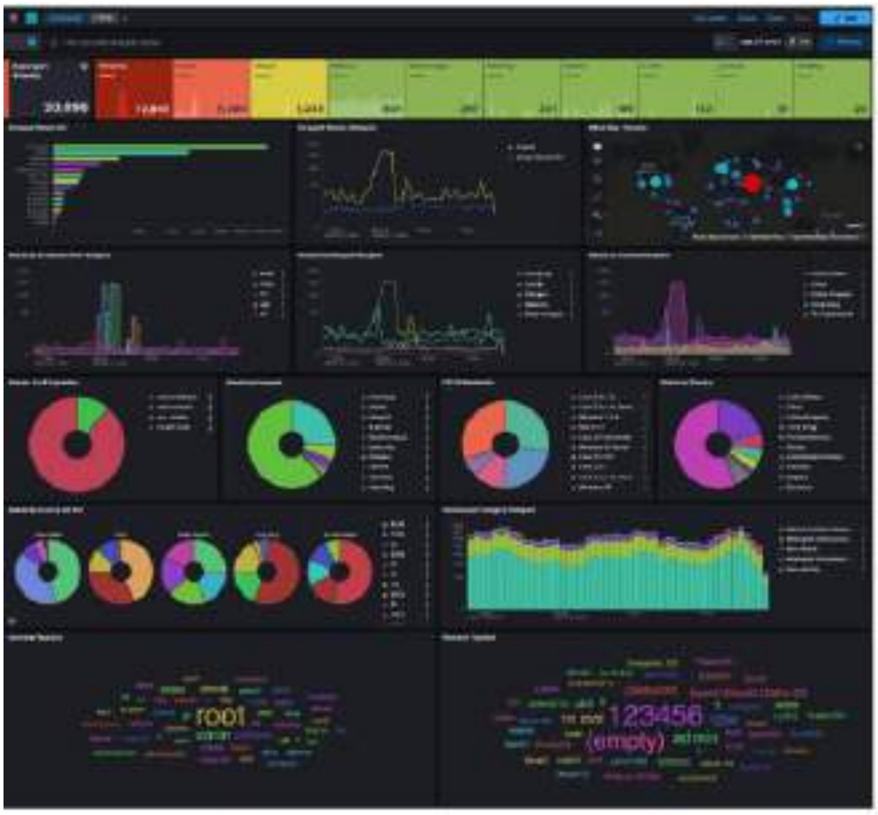


Figure 18. Visual representations of honeypot data in Kibana

- ***SpiderFoot:*** TI and reconnaissance tool to gather data about domains, networks, and hosts. It supports the identification and understanding of potential threats, contributing to a more robust security (see Figure 19).

Figure 19. SpiderFoot

The primary analysis tool within this implementation of the T-Pot environment is Kibana, which serves as the central platform for the study and visualisation of data captured by the selected honeypots. The interaction between data capture, storage, and visualization is facilitated by the ELK stack (Elasticsearch, Logstash and Kibana), a combination of technologies that ensures efficient data processing and representation.

- ES is a distributed, NoSQL search and analytics engine optimized for fast queries across large volumes of data. In the T-Pot framework:
    - It receives and stores log data forwarded by Logstash.
    - Enables high-speed searches and complex data queries.
    - Organizes data into JavaScript Object Notation (JSON)-based indices and documents.
    - All honeypot-generated logs within T-Pot are ultimately stored in Elasticsearch.
- Logstash works as a real-time data processor that ingests logs from various sources, transforms them into structured formats, and transmits them to ES. Logstash performs the following tasks:
    - Collects raw logs from honeypots (e.g. Cowrie, Wordpot, Mailoney).
    - Parses and structures the logs, typically into JSON format.
    - Sends structured data to ES, which is used by Kibana to generate dashboards and visualizations.
- Kibana is a web-based interface for data visualization and exploration of ES-stored content that enables:
    - Advanced queries with filtering capabilities.
    - Creation of interactive dashboards featuring dynamic charts and metrics.
    - Real-time visualization of honeypot data, including active attack events.

The user-friendly data representation provided by Kibana is fundamental for the study and development of the use cases. Through interactive dashboards, the tool permits the extraction, analysis, and documentation of a wide range of information related to observed attacks, including: attempted usernames and passwords, IP addresses and ports involved in the attacks, geolocation and timestamps, targeted services, and the number of attack attempts, alongside other critical insights.

Honeypots are configured to store captured data under a common index named Logstash by leveraging ES. This feature enables fast and efficient queries, streamlined data aggregation across all data flows, and unified visualization through a single dashboard. As a result, rapid deployment, visualization, and recording of attacks captured by the honeypots is achieved. Each honeypot of the T-Pot environment is configured to deliver captured data in JSON format, facilitating the integration with other modules and improving data processing and analysis. While the use of common data formats simplifies interoperability, it is worth noting that the structure and content of JSON outputs varies from one honeypot to another, depending on its functionality and design. As a result, although all honeypots generate outputs using JSON syntax, the actual fields, data granularity, and semantic content are heterogeneous across them. This structural discrepancy reflects the specific types of interactions that each honeypot is designed to capture, such as authentication attempts, file uploads, command execution, or protocol-specific requests.

The selected excerpts from the output logs generated by each honeypot, based on real interactions that were captured during the experimental phase are presented below (see

Figure 20,

Figure 21,

Figure 22 and

Figure 23):

[illegible]

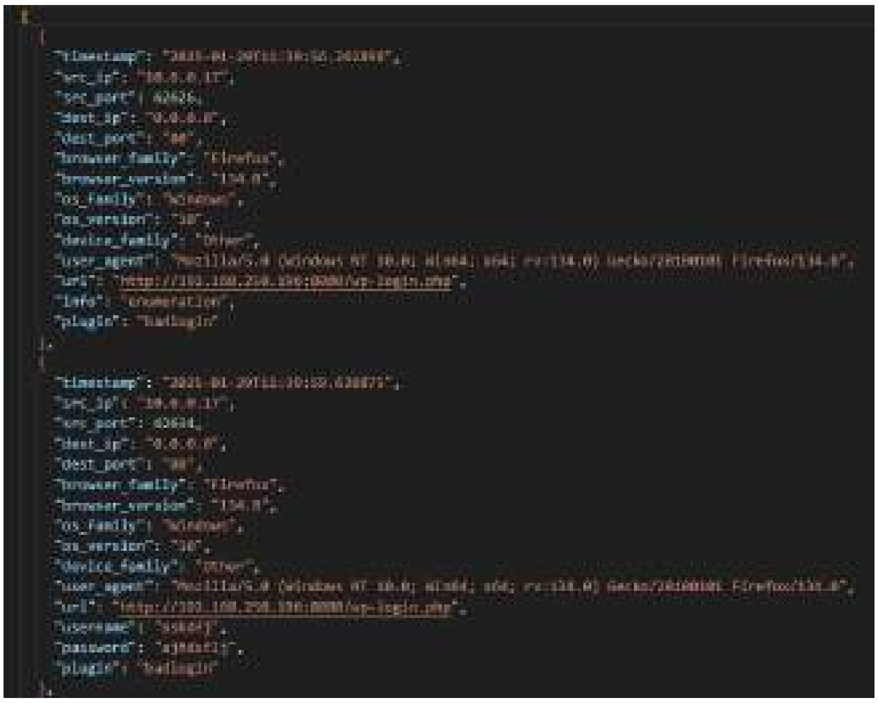
Figure 20. Cowrie

[illegible]

[illegible]

Figure 21. Dionaea

Figure 22. Mailoney



```
[
  {
    "timestamp": "2025-01-29T11:10:56.[illegible]",
    "src_ip": "10.0.0.1[illegible]",
    "src_port": 62626,
    "dest_ip": "0.0.0.0",
    "dest_port": "80",
    "browser_family": "Firefox",
    "browser_version": "134.0",
    "os_family": "Windows",
    "os_version": "10",
    "device_family": "Other",
    "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0",
    "url": "http://192.168.250.156:8080/wp-login.php",
    "info": "enumeration",
    "plugin": "badlogin"
  },
  {
    "timestamp": "2025-01-29T11:10:[illegible]",
    "src_ip": "10.0.0.1[illegible]",
    "src_port": 62634,
    "dest_ip": "0.0.0.0",
    "dest_port": "80",
    "browser_family": "Firefox",
    "browser_version": "134.0",
    "os_family": "Windows",
    "os_version": "10",
    "device_family": "Other",
    "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0",
    "url": "http://192.168.250.156:8080/wp-login.php",
    "username": "[illegible]",
    "password": "[illegible]",
    "plugin": "badlogin"
  },
```

Figure 23. Wordpot

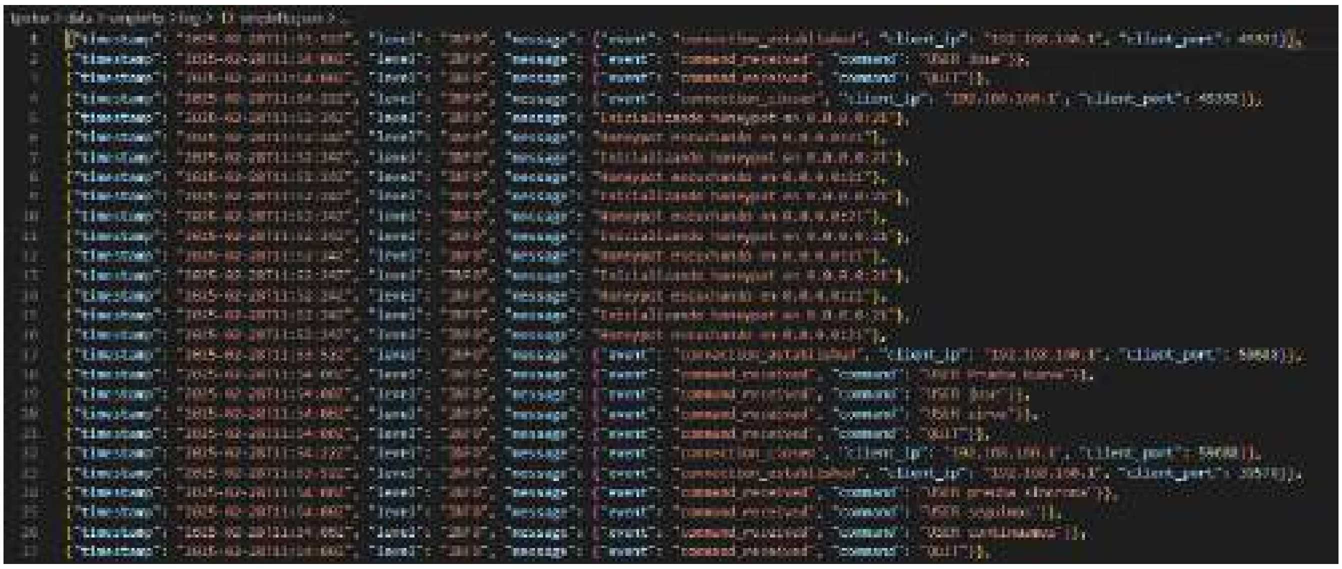
The implementation and integration of this custom T-Pot based honeypot environments aims to increase the scalability of the platform, moving towards the implementation of HIHs.

To provide a comprehensible first proof-of-concept and accessibility test, a simple honeypot approach denominated SimpleFTP was developed. The honeypot is conceived to capture basic FTP command’s traffic. The development process is adhered to the structural conventions of existing honeypots within T-Pot, ensuring compatibility and seamless integration. The core implementation encompasses the following components:

- **Dockerfile**: Defines the environment and dependencies for building the honeypot container.
- **Docker Compose file**: Specifies the deployment configuration for container orchestration within T-Pot.
- **Python script**: Implements the honeypot’s core logic, handling connections and logging interactions.

The configuration of these components results in the successful integration of SimpleFTP into the T-Pot environment. The initial tests carried out confirmed that the honeypot was operational, correctly capturing traffic and logging interactions during a preliminary connection attempt, as shown in Figure 24.

While this test remains simple, it showcases the feasible implementation of new honeypot approaches within the T-Pot environment. Initial tests and their results have yielded highly encouraging outcomes, reinforcing the potential and utility that can be leveraged from the T-Pot platform.



```
[illegible] > data > [illegible] > log > {} simpleftp.json > ...
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "connection_established", "client_ip": "192.168.100.1", "client_port": [illegible]}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "command_received", "command": "USER [illegible]"}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "command_received", "command": "QUIT"}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "connection_closed", "client_ip": "192.168.100.1", "client_port": [illegible]}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Inicializando honeypot en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Honeypot escuchando en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Inicializando honeypot en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Honeypot escuchando en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Inicializando honeypot en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Honeypot escuchando en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Inicializando honeypot en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Honeypot escuchando en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Inicializando honeypot en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Honeypot escuchando en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Inicializando honeypot en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": "Honeypot escuchando en 0.0.0.0:21"},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "connection_established", "client_ip": "192.168.100.1", "client_port": [illegible]}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "command_received", "command": "USER [illegible]"}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "command_received", "command": "USER [illegible]"}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "command_received", "command": "USER [illegible]"}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "command_received", "command": "QUIT"}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "connection_closed", "client_ip": "192.168.100.1", "client_port": [illegible]}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "connection_established", "client_ip": "192.168.100.1", "client_port": [illegible]}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "command_received", "command": "USER [illegible]"}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "command_received", "command": "USER [illegible]"}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "command_received", "command": "USER [illegible]"}},
{"timestamp": "2025-02-28T11:[illegible]", "level": "INFO", "message": {"event": "command_received", "command": "QUIT"}},
```

Figure 24. SimpleFTP operation

4.1.2 Digital Twin - Robotic Arm

Each subsystem of the Robotic Arm DT is responsible for a different aspect of robot control, data processing, and visualization. Each subsystem provides the necessary ground, including real-life data, APIs and end points that can allow the DT mimic physical equipment in view of deploying DT-backed deception mechanisms towards enhance cybersecurity under the AIAS deception layer architecture.

4.1.2.1 Unity Subsystem

- Runs on Windows and acts as the primary user interface for the system.
- Implements the ROS Transmission Control Protocol (TCP) Connector to establish communication with ROS 2.
- Handles three types of interactions:
  1. Gesture-based control (HoloLens 2 hand tracking).
  2. UI-based commands (buttons for moving joints).
  3. Predefined motion sequences (playback of stored movement patterns).
- Uses Unified Robot Description Format (URDF) Importer to load robot models into Unity for visualization.
- Ensures real-time rendering of the robot's DT.

The result of this system is found in

Figure 25.

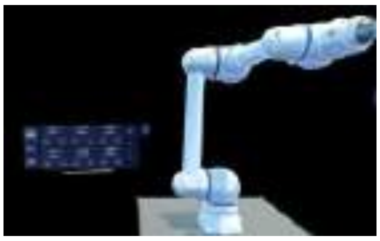
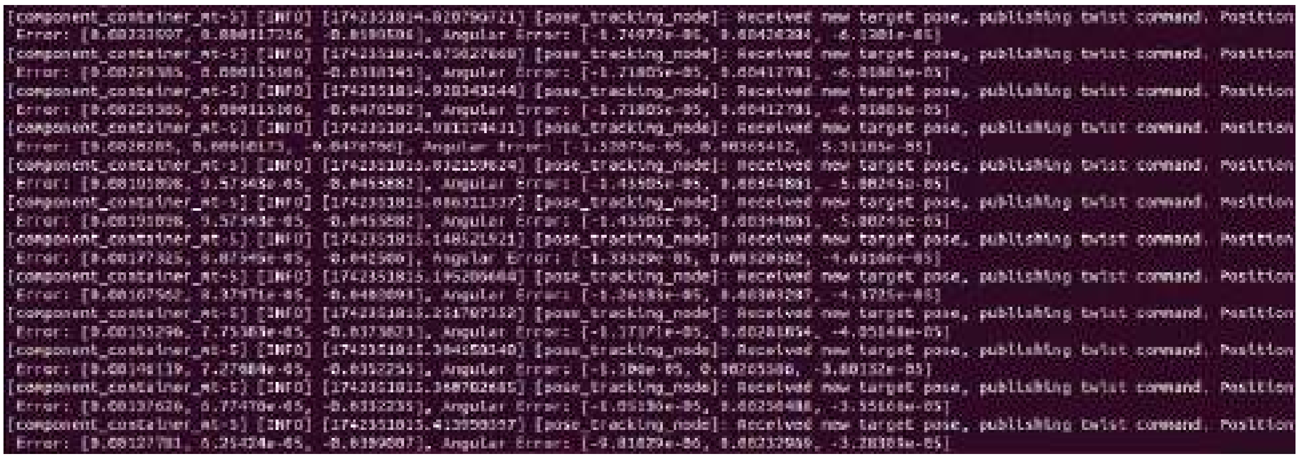


Figure 25. Unity Viewport with UI

4.1.2.2 ROS 2 Subsystem

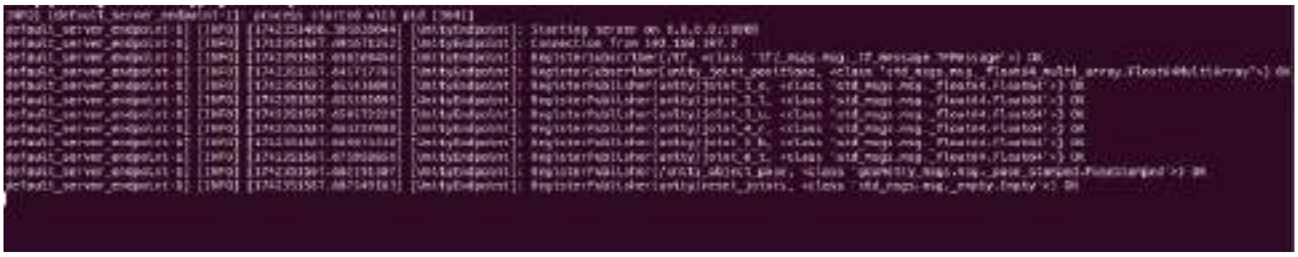
- Runs on Ubuntu 22.04 and manages motion execution, planning, and feedback.
- Implements MoveIt for motion planning, with:
    - Collision detection to prevent self-interference.
    - Path optimization for efficient movement execution.
    - IK solvers to compute required joint angles.
- Uses MoveIt Servo for real-time joint control, allowing immediate adjustments (see Figure 26).



```
[component_container_mt-5] [INFO] [1742351814.820790721] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00223007, 0.000117256, -0.0100506], Angular Error: [-1.74972e-06, 0.00420304, -6.1301e-05]
[component_container_mt-5] [INFO] [1742351814.873027660] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00229385, 0.000115108, -0.0310141], Angular Error: [-1.71805e-05, 0.00417781, -6.01881e-05]
[component_container_mt-5] [INFO] [1742351814.920343244] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00223385, 0.000115106, -0.0470502], Angular Error: [-1.71805e-05, 0.00412781, -6.01885e-05]
[component_container_mt-5] [INFO] [1742351814.981174431] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.0020285, 0.00010173, -0.0476706], Angular Error: [-1.53075e-05, 0.00365412, -5.31185e-05]
[component_container_mt-5] [INFO] [1742351815.032159624] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00191898, 9.57348e-05, -0.0455882], Angular Error: [-1.45505e-05, 0.00344801, -5.00245e-05]
[component_container_mt-5] [INFO] [1742351815.080311337] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00191898, 9.57348e-05, -0.0455882], Angular Error: [-1.45505e-05, 0.00344801, -5.00245e-05]
[component_container_mt-5] [INFO] [1742351815.140521921] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00177325, 8.87549e-05, -0.042300], Angular Error: [-1.33329e-05, 0.00320502, -4.61200e-05]
[component_container_mt-5] [INFO] [1742351815.195200400] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00167562, 8.37971e-05, -0.0400093], Angular Error: [-1.26183e-05, 0.00303287, -4.3725e-05]
[component_container_mt-5] [INFO] [1742351815.251707352] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00155296, 7.75389e-05, -0.0373821], Angular Error: [-1.17371e-05, 0.00281054, -4.05148e-05]
[component_container_mt-5] [INFO] [1742351815.304250340] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00148139, 7.27084e-05, -0.0352255], Angular Error: [-1.108e-05, 0.00265386, -3.80132e-05]
[component_container_mt-5] [INFO] [1742351815.360702685] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00137626, 6.77470e-05, -0.0332235], Angular Error: [-1.05130e-05, 0.00250488, -3.55168e-05]
[component_container_mt-5] [INFO] [1742351815.413050037] [pose_tracking_node]: Received new target pose, publishing twist command. Position
 Error: [0.00127781, 6.35434e-05, -0.0309087], Angular Error: [-9.81029e-06, 0.00232989, -3.28383e-05]
```

Figure 26. ROS utilizing MoveIt Servo to perform IK

- Runs ROS TCP Endpoint, acting as a server for Unity’s TCP Connector, processing incoming commands (see Figure 27).



```
[INFO] [default_server_endpoint-1]: process started with pid [3041]
[default_server_endpoint-1] [INFO] [1742351406.303020044] [UnityEndpoint]: Starting server on 0.0.0.0:10000
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: Connection from 192.168.187.2
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: RegisterSubscriber(/TF, <class 'tf2_msgs.msg._TF_message.TFMessage'>) OK
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: RegisterSubscriber(unity_joint_positions, <class 'std_msgs.msg._float64_multi_array.Float64MultiArray'>) OK
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: RegisterPublisher(unity/joint_1_s, <class 'std_msgs.msg._float64.Float64'>) OK
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: RegisterPublisher(unity/joint_2_l, <class 'std_msgs.msg._float64.Float64'>) OK
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: RegisterPublisher(unity/joint_3_u, <class 'std_msgs.msg._float64.Float64'>) OK
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: RegisterPublisher(unity/joint_4_r, <class 'std_msgs.msg._float64.Float64'>) OK
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: RegisterPublisher(unity/joint_5_b, <class 'std_msgs.msg._float64.Float64'>) OK
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: RegisterPublisher(unity/joint_6_t, <class 'std_msgs.msg._float64.Float64'>) OK
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: RegisterPublisher(/unity_object_pose, <class 'geometry_msgs.msg._pose_stamped.PoseStamped'>) OK
[default_server_endpoint-1] [INFO] [illegible] [UnityEndpoint]: RegisterPublisher(unity/reset_joints, <class 'std_msgs.msg._empty.Empty'>) OK
```

Figure 27. ROS and Unity connecting

4.1.2.3 HoloLens 2 Subsystem

- Acts as an AR visualization and interaction layer.
- Receives real-time updates from Unity, allowing users to:
    - Manipulate the robot model using gestures.
    - Select joints for individual movement adjustments.
    - View real-time feedback from the physical robot in AR.

- Supports Unity Remoting, offloading processing tasks to a PC for better performance.

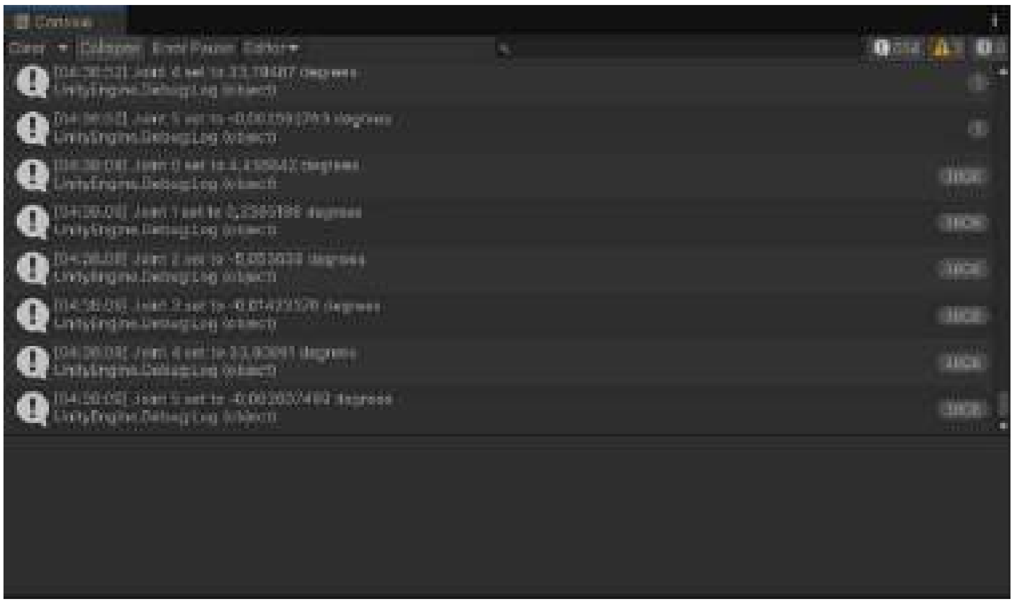
4.1.2.4 Robot Subsystem (Optional in Real-World Execution)

- Receives joint movement commands from MoveIt Servo.
- Sends real-time feedback on current joint positions and execution status.
- Uses industrial controllers (e.g. Yaskawa, UR, Fanuc) to execute commands.

4.1.2.5 Communication Model and Interfaces

The system operates on a TCP-based communication model, ensuring low-latency data exchange.

- Unity → ROS 2 (Command Transmission)
  - Commands are serialized and sent over TCP/IP.
  - MoveIt Servo interprets and executes motion commands (see Figure 28).



```
[04:38:52] Joint 4 set to 33.78487 degrees
UnityEngine.Debug:Log (object)
[04:38:52] Joint 5 set to -0.00159179 degrees
UnityEngine.Debug:Log (object)
[04:38:08] Joint 0 set to 4.4358642 degrees
UnityEngine.Debug:Log (object)
[04:38:08] Joint 1 set to 0.2380198 degrees
UnityEngine.Debug:Log (object)
[04:38:08] Joint 2 set to -5.053633 degrees
UnityEngine.Debug:Log (object)
[04:38:08] Joint 3 set to -6.01423578 degrees
UnityEngine.Debug:Log (object)
[04:38:08] Joint 4 set to 33.80891 degrees
UnityEngine.Debug:Log (object)
[04:38:08] Joint 5 set to -0.00283748 degrees
UnityEngine.Debug:Log (object)
```

Figure 28. Unity receiving joint states from ROS

- ROS 2 → Unity (Feedback Transmission)
  - Joint state updates, sensor data, and execution status are published to Unity.
  - DT in Unity is updated dynamically to reflect real-world changes (see Figure 29).

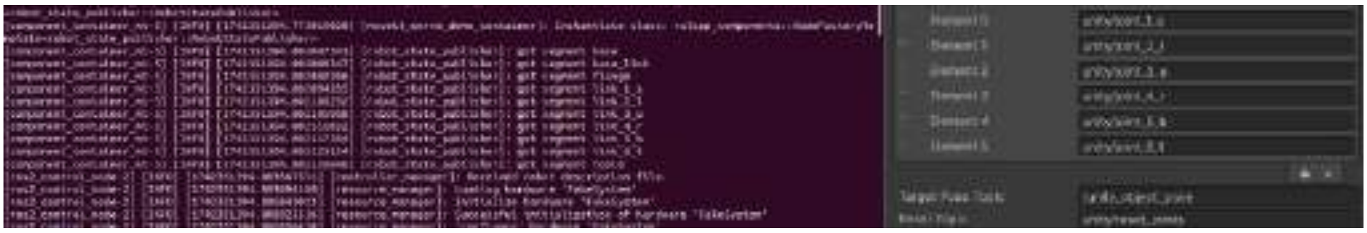


Figure 29. Unity subscribing to ROS topics

4.1.3 Digital Twin - Environmental IoT Devices

The development of the EIoT Devices DT comprises both a real network and a server-side network that holds DT logic and the specific Docker containers for the imitation of each device. The central component of the system is the visualisation module (Figure 30), which not only graphically presents the administration capabilities of ADMIN-M, but also runs the intrinsic logic of the SYNC-M, applying user synchronisation configurations in real-time.

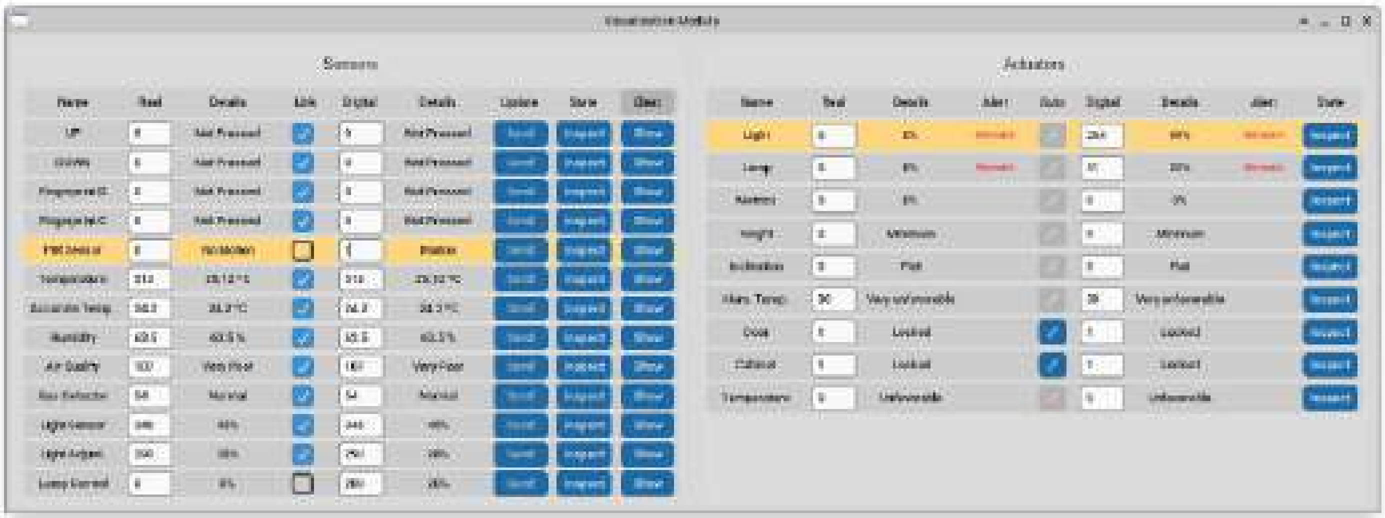
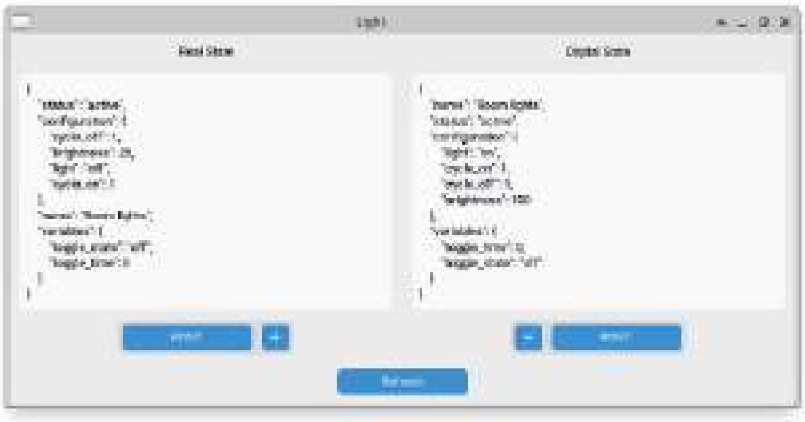


Figure 30. DT graphical interface for visualisation and configuration

Figure 30 illustrates the interaction possibilities within the interface, highlighting the actuators affected by selected sensors, and the mismatch alerts for incongruent output states. Supervisors are able to desynchronise assets and either set static values or run generation functions towards the MQTT topics that controls the DT. Additionally, automatic protection options for actuators are available, enabling the DT to automatically fix incongruences with predefined rules. Configurations can also be adjusted by using the *“inspect”* option, which pops-up an interactive endpoint (see Figure 31) for both the real and digital asset.



```
{
  "status": "active",
  "configuration": {
    "cycle_off": 1,
    "brightness": 25,
    "light": "off",
    "cycle_on": 1
  },
  "name": "Room lights",
  "variables": {
    "toggle_state": "off",
    "toggle_time": 0
  }
}
```

```
{
  "name": "Room lights",
  "status": "active",
  "configuration": {
    "light": "on",
    "cycle_on": 1,
    "cycle_off": 1,
    "brightness": 100
  },
  "variables": {
    "toggle_time": 0,
    "toggle_state": "off"
  }
}
```

Figure 31. Interactive HTTP endpoint for asset configuration

4.1.3.1 EIoT Devices-based DT validation

With the aim of proportionating a comprehensive sight of system functioning, Figure 32 shows the resulting monitoring of both a real temperature sensor and its replica in the DT. The perfect synchronisation between devices provokes the traces to overlap in the graphic, highlighting the precise configuration of system inputs.

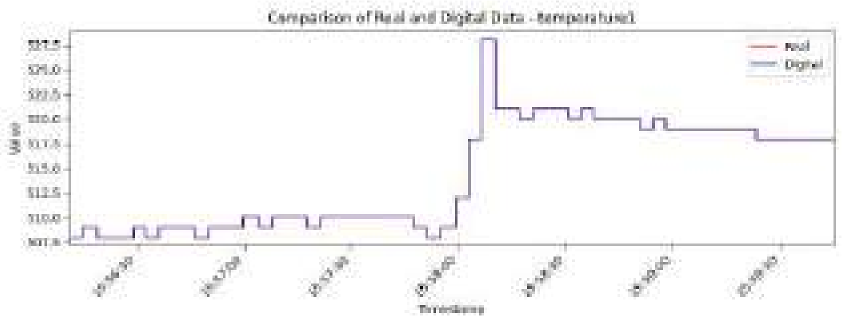


Figure 32. Monitoring of both a real temperature sensor a its digital counterpart

Another example is exposed in the following lines, where the interaction between two devices is shown. Figure 33 presents the states of a potentiometer which input is used to adjust the brightness of the LED in Figure 34. While potentiometer’s traces overlap demonstrating the perfect synchronisation, the graphic for LED presents a small delay in real traces due to the system being monitored from the digital side. Despite that fact, actuator graphics are identical, showcasing the identical behaviour in both systems.

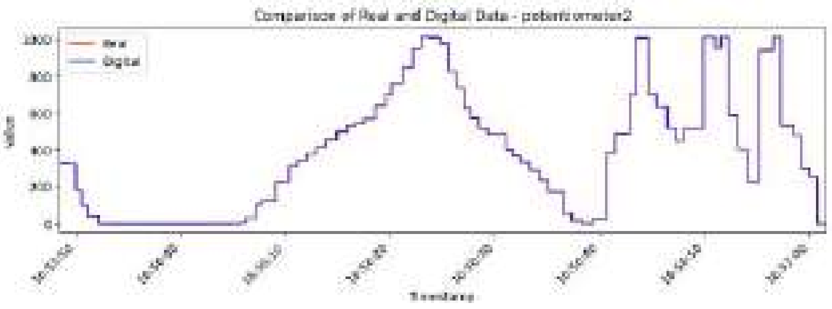


Figure 33. Comparison of real and digital states for a potentiometer

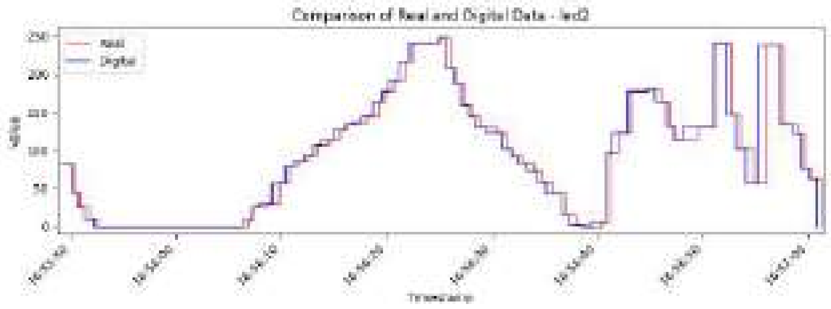


Figure 34. Comparison of real and digital states for a LED device

Regarding the automation of proactive decisions, Figure 35 provides a comprehensible view of the response of the system to mismatching configurations between the real and virtualised system. The graphic shows normal functioning of a LED device that turns on (positive value of brightness) for a period, before turning off (0% brightness) again according to received inputs. Nonetheless, around timestamp 15:21:50, two attempts to alter the state of the device are launched, resulting in different traces for real and digital counterparts. This discrepancy rapidly triggers mechanisms in DT to recover the state of real assets. After recovery, system continues operating in normal condition.

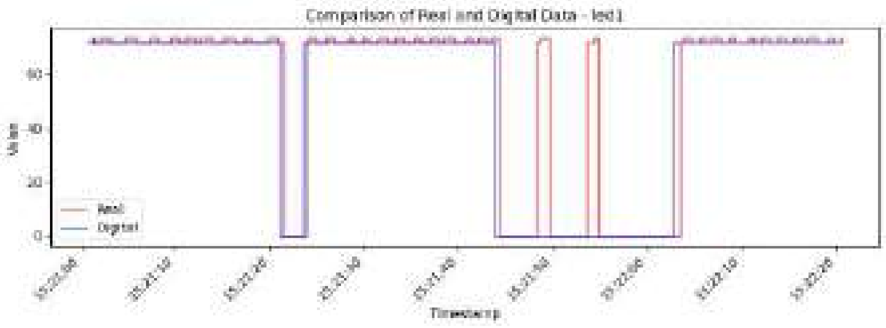


Figure 35. Autonomous response to system incongruences

4.1.4 Virtual Persona

The development of the VP–DT system was conducted within a controlled, virtualized computing environment and structured as a multi-layered software architecture combining artificial intelligence, cyber-physical simulation, anomaly detection, and human-AI interaction. The purpose of the system is to simulate and monitor the real-time dynamics of a hospital ward environment, detect deviations from normative conditions, and provide intelligent, role-specific responses to clinical stakeholders through a virtual interface. Thus, this section details the methodological stages and implementation steps used to build the system from the ground up, with a particular emphasis on interoperability, modularity, reproducibility and scalability.

4.1.4.1 Phase I - VM environment and development stack

To ensure an isolated, reliable and secure development environment, all components were deployed within a VM hosted on a local workstation. This decision supported experimentation with system-level communication protocols (e.g. MQTT, REST), without the risks of dependency conflicts or uncontrolled network behaviour.

The VM was provisioned using Oracle VirtualBox 7.x, hosting a 64-bit instance of Ubuntu 22.04 LTS. The configuration was deliberately constrained to simulate embedded or edge-computing conditions, with:

- 2 CPU cores
- 4 GB RAM
- 30 GB dynamically allocated virtual storage
- Python 3.10.x, managed via venv (virtual environments)
- Key libraries: flask, paho-mqtt, openai, scikit-learn, numpy, pandas, UUID and logging

The VM network was configured using Network Address Translation (NAT) + port forwarding to allow external access to local servers (e.g. Flask app on port 5000, MQTT broker on port 1883). This configuration supports integration with external dashboards, API testing tools (e.g. Postman) and network simulators.

4.1.4.2 Phase II - Digital Twin: virtual sensor simulation layer

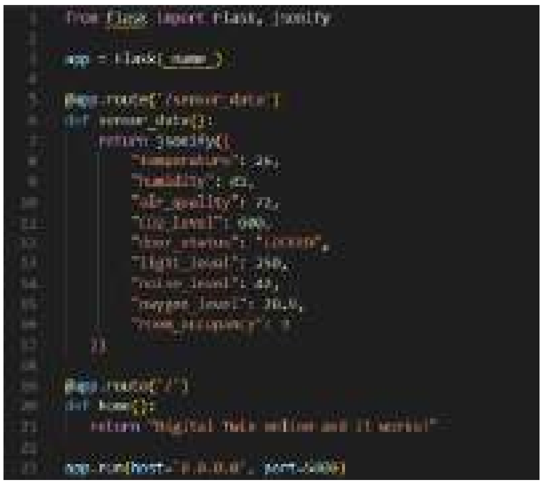
To simulate and evaluate VP mode functionality, a DT component was built to replicate the hospital environment. DT was implemented as a Flask-based RESTful web service, exposing a */sensor_data* endpoint returning synthetic environmental data in JSON format. The payload represents a real-time depiction of a hospital room’s state, including parameters such as the presented in Table 10.

Table 10. Parameters monitored

Parameter	Characteristics
Temperature	°C
Humidity	%
CO_2 concentration	ppm
Air Quality Index	AQI
Oxygen level	%
Noise level	dB
Light level	lux/lx

Motion detection	boolean
Room occupancy	integer
Door status	LOCKED/UNLOCKED

This component abstracts the physical layer of sensor acquisition, serving as a digital proxy for real-time physical conditions, and allows the rest of the system to operate without hardware dependencies. The corresponding code to this part is found in Figure 36.

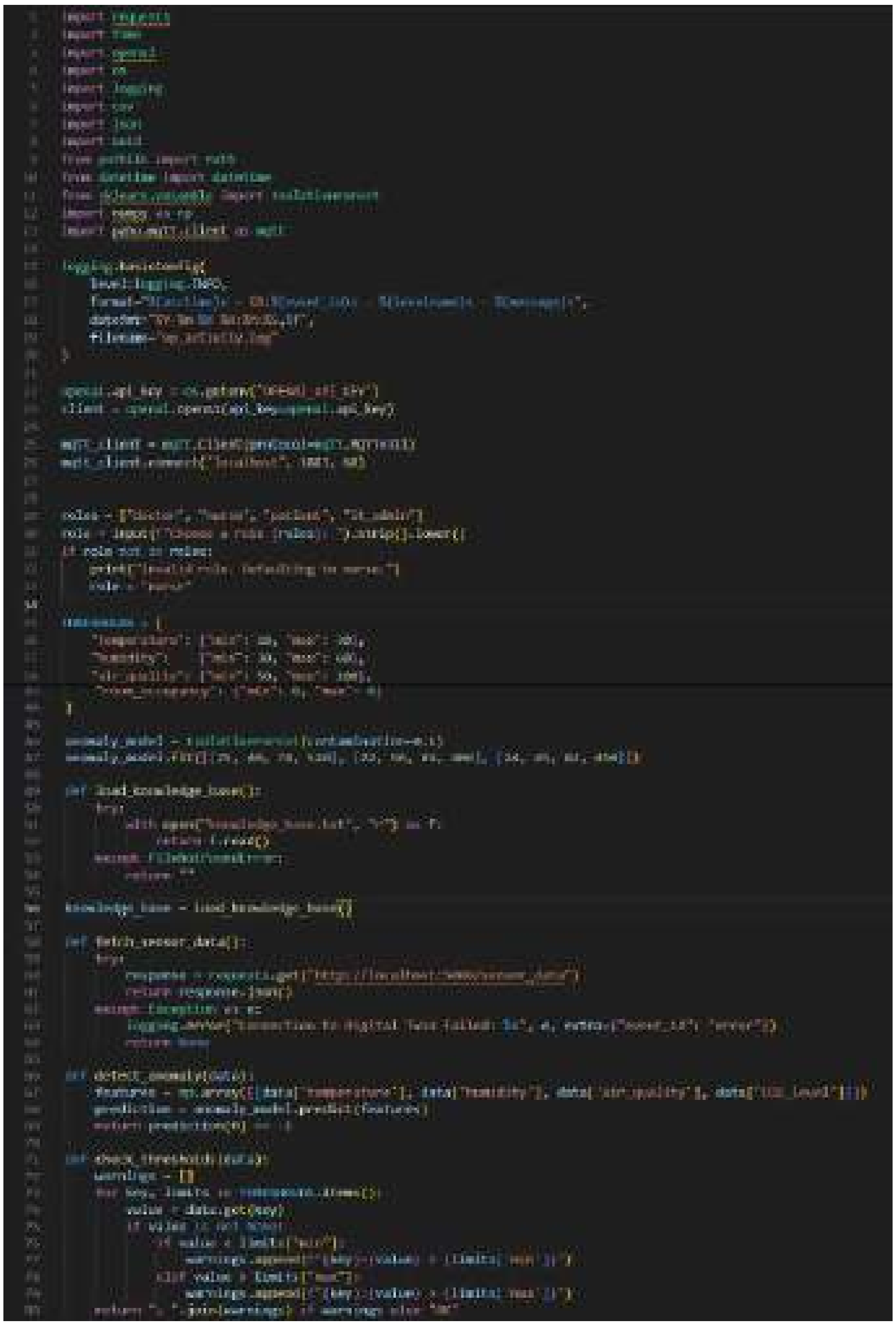


```python
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/sensor_data')
def sensor_data():
    return jsonify({
        "temperature": 26,
        "humidity": 45,
        "air_quality": 72,
        "co2_level": 600,
        "door_status": "LOCKED",
        "light_level": 250,
        "noise_level": 40,
        "oxygen_level": 20.9,
        "room_occupancy": 3
    })

@app.route('/')
def home():
    return "Digital Twin online and it works!"

app.run(host='0.0.0.0', port=5000)
```

Figure 36. Code for DT

4.1.4.3 Phase III - VP core: sensing, reasoning and acting

VP functions as the cognitive agent and main decision-making entity of architecture. It functions within a continuous polling process, collecting data from the DT at 10-second intervals. Upon receiving each payload, the VP performs a hybrid analytic pipeline which includes human-centred design concepts into the AI feedback loop (see associated code in Figure 37):



```python
import requests
import time
import openai
import os
import logging
import csv
import json
import uuid
from pathlib import Path
from datetime import datetime
from sklearn.ensemble import IsolationForest
import numpy as np
import paho.mqtt.client as mqtt

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - [illegible] - %(levelname)s - %(message)s",
    datefmt="%Y-%m-%d %H:%M:%S,%f",
    filename="vp_activity.log"
)

openai.api_key = os.getenv("OPENAI_API_KEY")
client = openai.OpenAI(api_key=openai.api_key)

mqtt_client = mqtt.Client(protocol=mqtt.MQTTv311)
mqtt_client.connect("localhost", 1883, 60)


roles = ["doctor", "nurse", "patient", "it_admin"]
role = input(f"Choose a role {roles}: ").strip().lower()
if role not in roles:
    print("Invalid role. Defaulting to nurse.")
    role = "nurse"

thresholds = {
    "temperature": {"min": 18, "max": 30},
    "humidity":    {"min": 30, "max": 60},
    "air_quality": {"min": 50, "max": 100},
    "room_occupancy": {"min": 0, "max": 4}
}

anomaly_model = IsolationForest(contamination=0.1)
anomaly_model.fit([[25, 40, 70, 420], [22, 50, 65, 400], [24, 45, 60, 450]])

def load_knowledge_base():
    try:
        with open("knowledge_base.txt", "r") as f:
            return f.read()
    except FileNotFoundError:
        return ""

knowledge_base = load_knowledge_base()

def fetch_sensor_data():
    try:
        response = requests.get("http://localhost:5000/sensor_data")
        return response.json()
    except Exception as e:
        logging.error("Connection to digital twin failed: %s", e, extra={"user_id": "error"})
        return None

def detect_anomaly(data):
    features = np.array([[data["temperature"], data["humidity"], data["air_quality"], data["CO2_level"]]])
    prediction = anomaly_model.predict(features)
    return prediction[0] == -1

def check_threshold(data):
    warnings = []
    for key, limits in thresholds.items():
        value = data.get(key)
        if value is not None:
            if value < limits["min"]:
                warnings.append(f"{key}={value} < {limits['min']}")
            elif value > limits["max"]:
                warnings.append(f"{key}={value} > {limits['max']}")
    return "; ".join(warnings) if warnings else "OK"
```

Figure 37. Code for VP

- Rule-based threshold verification
    - Each sensor parameter is evaluated against a range of allowable values defined by the domain - i.e. temperature ∈ [18-30°C]. Breaches trigger notifications and initiate downstream alert systems.
- Unsupervised anomaly detection

    - Multivariate anomalies are found using an Isolation Forest model (trained on baseline operating data), therefore providing defence against either known or emergent failure circumstances. This phase ensures sensitivity to patterns not explicitly captured by hard-coded rules.
- Role-aware decision context
    - The system supports four clinical personas: medical doctor, nurse, patient, and IT administrator. Based on the active role, the VP customizes both the natural language output and response protocols, embedding human-centred design principles into the AI feedback loop.

4.1.4.4 Phase IV - AI-driven natural language generation

For real-time decision support and human-readable interaction, the system integrates with OpenAI’s GPT-3.5 Turbo API via HTTPS. Every sensing cycle, the VP creates a prompt context consisting of:

- current sensor data
- the selected user role
- preloaded knowledge base content (e.g. from knowledge_base.txt)

This prompt is submitted to the /v1/chat/completions endpoint, and the resulting message (AI-generated textual advice) is returned to the console and published via MQTT. This interaction introduces a semantic understanding layer, transforming raw data into actionable guidance.

4.1.4.5 Phase V - Messaging infrastructure: MQTT

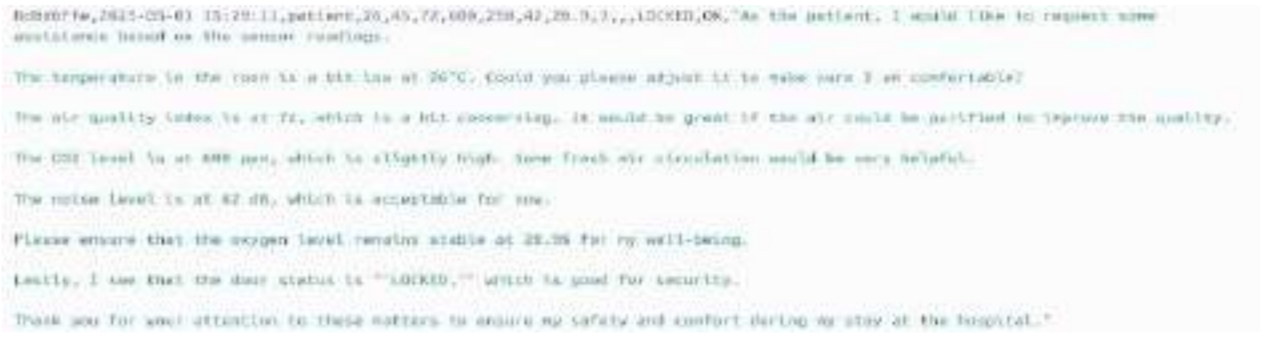
All asynchronous, event-driven communications are controlled using the MQTT protocol. The VP acts as a client to a local MQTT broker and communicates messages to two distinct aspects:

- hospital/alerts: when anomalies or breaches are detected
- hospital/response: when GPT-generated responses are issued

4.1.4.6 Phase VI - Logging and data persistence layer

Every interaction, comprising sensor readings, alert status, AI responses and metadata, is persistently logged in both:

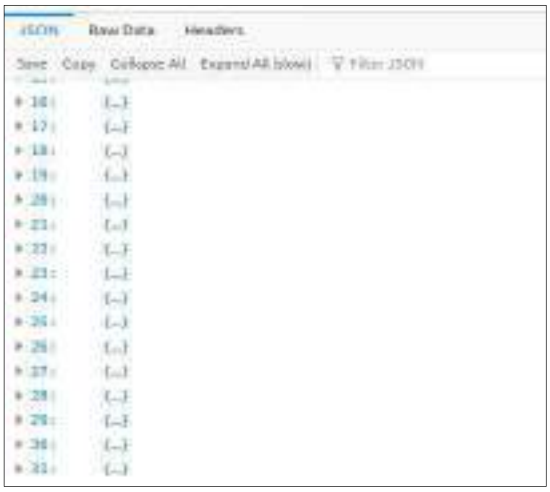
- .csv files (for human review and statistical analysis) – see **Figure 38**.



```
8c8b0ffe,2025-05-03 15:29:11,patient,26,45,72,600,250,42,20.9,3,,,LOCKED,OK,"As the patient, I would like to request some
assistance based on the sensor readings.

The temperature in the room is a bit low at 26°C. Could you please adjust it to make sure I am comfortable?

The air quality index is at 72, which is a bit concerning. It would be great if the air could be purified to improve the quality.

The CO2 level is at 600 ppm, which is slightly high. Some fresh air circulation would be very helpful.

The noise level is at 42 dB, which is acceptable for now.

Please ensure that the oxygen level remains stable at 20.9% for my well-being.

Lastly, I see that the door status is ""LOCKED,"" which is good for security.

Thank you for your attention to these matters to ensure my safety and comfort during my stay at the hospital."
```

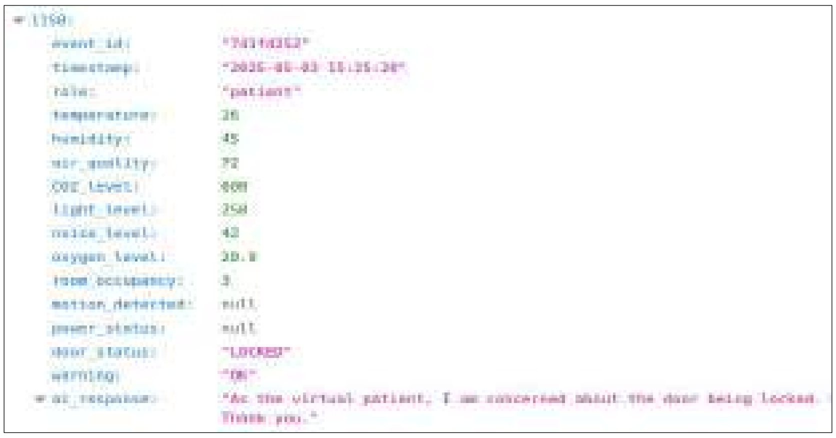
Figure 38. Example of .csv file

- .json files using NDJSON format (machine-readable for dashboards and APIs – see Figure 39 and Figure 40).



```
▸ 16:   {…}
▸ 17:   {…}
▸ 18:   {…}
▸ 19:   {…}
▸ 20:   {…}
▸ 21:   {…}
▸ 22:   {…}
▸ 23:   {…}
▸ 24:   {…}
▸ 25:   {…}
▸ 26:   {…}
▸ 27:   {…}
▸ 28:   {…}
▸ 29:   {…}
▸ 30:   {…}
▸ 31:   {…}
```

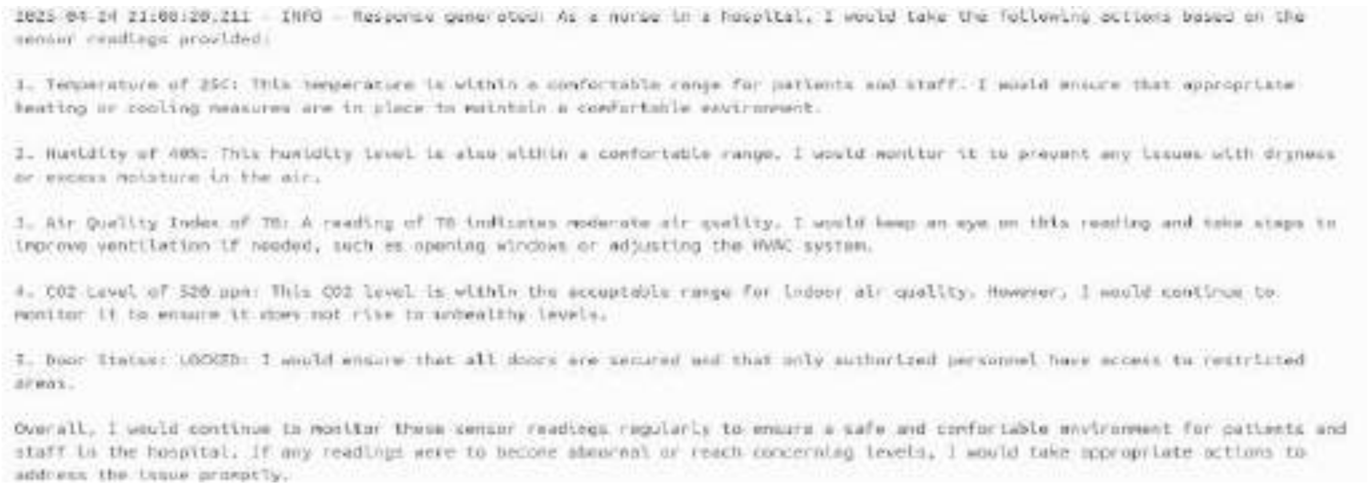
Figure 39. List of logs (NDJSON format)



```
▾ 1150:
    event_id:          "7d1fd352"
    timestamp:         "2025-05-03 15:35:26"
    role:              "patient"
    temperature:       26
    humidity:          45
    air_quality:       72
    CO2_level:         600
    light_level:       250
    noise_level:       42
    oxygen_level:      20.9
    room_occupancy:    3
    motion_detected:   null
    power_status:      null
    door_status:       "LOCKED"
    warning:           "OK"
  ▾ ai_response:       "As the virtual patient, I am concerned about the door being locked.
                       Thank you."
```

Figure 40. Example of NDJSON file

Each line includes a unique *event_id*, ISO-formatted timestamp, warning messages and role context (Figure 41). To ensure JSON compliance, NDJSON logs are automatically converted into array-structured JSON (*_fixed.json) files, compatible with platforms such as Grafana, Power BI and ES. This dual-format logging strategy supports both retrospective analysis and real-time data ingestion for visualization tools.



```
2025-04-24 21:08:20.211 - INFO - Response generated: As a nurse in a hospital, I would take the following actions based on the
sensor readings provided:

1. Temperature of 25C: This temperature is within a comfortable range for patients and staff. I would ensure that appropriate
heating or cooling measures are in place to maintain a comfortable environment.

2. Humidity of 40%: This humidity level is also within a comfortable range. I would monitor it to prevent any issues with dryness
or excess moisture in the air.

3. Air Quality Index of 78: A reading of 78 indicates moderate air quality. I would keep an eye on this reading and take steps to
improve ventilation if needed, such as opening windows or adjusting the HVAC system.

4. CO2 Level of 520 ppm: This CO2 level is within the acceptable range for indoor air quality. However, I would continue to
monitor it to ensure it does not rise to unhealthy levels.

5. Door Status: LOCKED: I would ensure that all doors are secured and that only authorized personnel have access to restricted
areas.

Overall, I would continue to monitor these sensor readings regularly to ensure a safe and comfortable environment for patients and
staff in the hospital. If any readings were to become abnormal or reach concerning levels, I would take appropriate actions to
address the issue promptly.
```

Figure 41. Example of elements of a log

4.1.4.7 Phase VII - Interface integration and extensibility

The architecture is highly modular, exposing standardized interfaces:

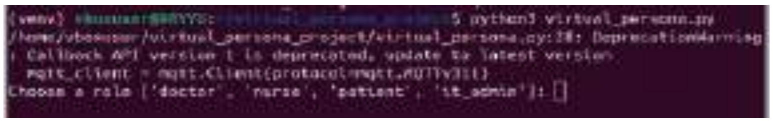
- RESTful HTTP (GET) for sensor polling
- HTTPS POST for AI dialogue generation
- MQTT broker for event distribution
- File I/O for structured logs

This separation of concerns ensures the system can be extended with real sensors, new personas, predictive analytics modules, or integration into clinical IT systems.

4.1.4.8 Interaction between Digital Twin module and VP module

Two independent but interrelated phases of the VP system’s execution flow contribute to the functional integrity of the intelligent agent. To ensure encapsulated dependency management and system reproducibility in the first phase, the VP is activated within a virtualized Python environment on a Linux-based VM.

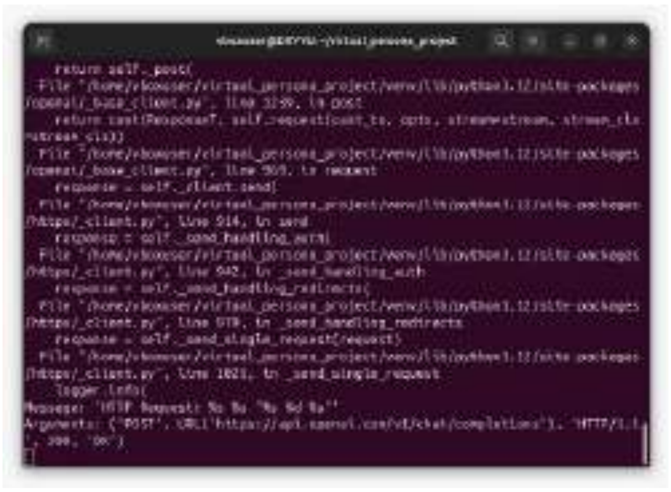
The user runs the *virtual_persona.py* script first by explicitly setting the OpenAI API key as an environment variable, therefore authenticating access to the external language model. Once executed, the system asks the user to choose one of four predefined operational roles (medical doctor, nurse, patient, or IT admin) each controlling the perspective and behavioural context of the AI agent across the conversation (all in Figure 42).



```
(venv) vboxuser@BRYYS:[illegible]$ python3 virtual_persona.py
/home/vboxuser/virtual_persona_project/virtual_persona.py:28: DeprecationWarning
: Callback API version 1 is deprecated, update to latest version
  mqtt_client = mqtt.Client(protocol=mqtt.MQTTv311)
Choose a role ['doctor', 'nurse', 'patient', 'it_admin']: 
```

Figure 42. VP – phase 1

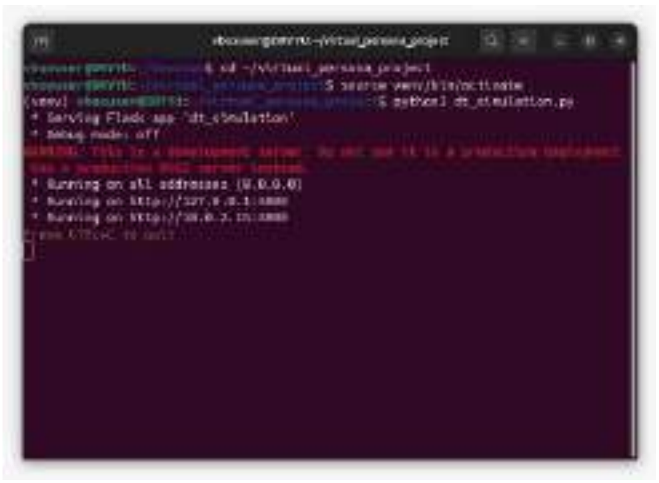
Concurrently, the MQTT client is configured with version MQTTv311, therefore preparing for asynchronous message distribution to subscribed topics. The second step shows how the VP runs through into active decision-making (see Figure 43).



```
    return self._post(
  File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/openai/_base_client.py", line 1239, in post
    return cast(ResponseT, self.request(cast_to, opts, stream=stream, stream_cls
=stream_cls))
  File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/openai/_base_client.py", line 969, in request
    response = self._client.send(
  File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/httpx/_client.py", line 914, in send
    response = self._send_handling_auth(
  File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/httpx/_client.py", line 942, in _send_handling_auth
    response = self._send_handling_redirects(
  File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/httpx/_client.py", line 979, in _send_handling_redirects
    response = self._send_single_request(request)
  File "/home/vboxuser/virtual_persona_project/venv/lib/python3.12/site-packages
/httpx/_client.py", line 1021, in _send_single_request
    logger.info(
Message: 'HTTP Request: %s %s "%s %d %s"'
Arguments: ('POST', URL('https://api.openai.com/v1/chat/completions'), 'HTTP/1.1
', 200, 'OK')
```

Figure 43. VP – phase 2

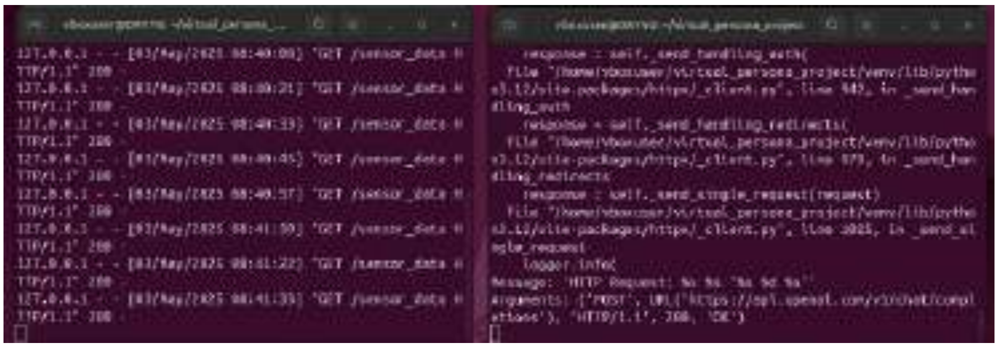
Once sensor data has been extracted from the DT, a structured prompt, embedding environmental measurements and user role context, is built and sent via HTTPS to the OpenAI GPT endpoint (/v1/chat/completions) (see Figure 44).



```
vboxuser@BRYYS:~$ cd ~/virtual_persona_project
vboxuser@BRYYS:[illegible]$ source venv/bin/activate
(venv) vboxuser@BRYYS:[illegible]$ python3 dt_simulation.py
 * Serving Flask app 'dt_simulation'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
 Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.2.15:5000
Press CTRL+C to quit
```

Figure 44. DT created – phase 1

Captured in the logs, the effective HTTP/1.1 200 OK response guarantees bidirectional communication and completion of the cognitive inference process. The acquired GPT-generated natural language output is thereafter logged locally and conditionally disseminated to MQTT channels, that is, hospital/response. All coordinated through modular interfaces and role-specific logic, a defining feature of intelligent CPSs in healthcare environments, this two-phase interaction validates the end-to-end pipeline, including environmental sensing, context-aware reasoning, and multimodal communication (see Figure 45).



```
127.0.0.1 - - [03/May/2025 08:40:08] "GET /sensor_data H
TTP/1.1" 200 -
127.0.0.1 - - [03/May/2025 08:40:21] "GET /sensor_data H
TTP/1.1" 200 -
127.0.0.1 - - [03/May/2025 08:40:33] "GET /sensor_data H
TTP/1.1" 200 -
127.0.0.1 - - [03/May/2025 08:40:45] "GET /sensor_data H
TTP/1.1" 200 -
127.0.0.1 - - [03/May/2025 08:40:57] "GET /sensor_data H
TTP/1.1" 200 -
127.0.0.1 - - [03/May/2025 08:41:10] "GET /sensor_data H
TTP/1.1" 200 -
127.0.0.1 - - [03/May/2025 08:41:22] "GET /sensor_data H
TTP/1.1" 200 -
127.0.0.1 - - [03/May/2025 08:41:35] "GET /sensor_data H
TTP/1.1" 200 -
```

```
    response = self._send_handling_auth(
  File "/home/vboxuser/virtual_persona_project/venv/lib/pytho
n3.12/site-packages/httpx/_client.py", line 942, in _send_han
dling_auth
    response = self._send_handling_redirects(
  File "/home/vboxuser/virtual_persona_project/venv/lib/pytho
n3.12/site-packages/httpx/_client.py", line 979, in _send_han
dling_redirects
    response = self._send_single_request(request)
  File "/home/vboxuser/virtual_persona_project/venv/lib/pytho
n3.12/site-packages/httpx/_client.py", line 1025, in _send_si
ngle_request
    logger.info(
Message: 'HTTP Request: %s %s "%s %d %s"'
Arguments: ('POST', URL('https://api.openai.com/v1/chat/compl
etions'), 'HTTP/1.1', 200, 'OK')
```

Figure 45. Interaction between DT – VP

4.2 Monitoring And Security Analytics Logic

In this section, the execution of core AIAS MAT is analysed, investigating its feasibility. For the proof-of-concept, the AIAS MAT tool is implemented on a VM inside a server equipped with an 8 x Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz (1 Socket), 32 GB RAM. The tool integrates an ELK stack docker-compose file to set up the environment and configured a public IP. A python script was also used for manual ingestion of the data into ELK. All defined in Table 11

Table 11. AIAS MAT setup

Entity	Setup
Data collection	Elasticsearch
Data ingestion	Elasticsearch
Data Processing	Logstash
Data Analytics	Kibana
Data Visualization	Kibana

The Data Visualization and Reporting as mentioned is the module which transforms processed security data into actionable insights, and we anticipate it as the target of the AIAS MAT. Leveraging Kibana dashboards, automated reporting mechanisms, and analytical tools, present the adversarial activity trends and operational metrics in a user-friendly format. This enables security teams to make informed decisions based on clear and structured visualizations.

4.2.1 Workflow

The workflow is as follows:

1. ***Real-time dashboards:*** critical security metrics, such as adversarial activity trends and log ingestion rates, are displayed on customizable dashboards offered by Kibana. Interactive heat maps, time-series graphs, and attack correlation charts, provide tailored views for different cybersecurity roles and needs.

- ***Data aggregation:*** processed security data is collected and organized into custom predefined categories.

2. ***Custom query and filtering:*** security analysts can drill down into specific events using ES Query DSL*, with structured query interfaces enabling detailed log analysis and categorization.
3. ***Visualization and dashboarding:*** insights are displayed via Kibana dashboards, with analysts able to customize visualizations as needed by each scenario.
4. ***Automated reporting:*** periodic reports summarize key security insights, including log analysis, incident summaries, types of vulnerabilities and recommended defensive actions.
    - ***Report generation and distribution:*** analytical reports are compiled and distributed, highlighting key findings and all the results.
    - ***Event correlation:*** visual clustering techniques highlight potential attack patterns.
    - ***Security insights:*** actionable intelligence guides strategies, with historical trend analysis by helping improve future threat mitigation efforts and helping with existing problems.

*ES provides a full Domain Specific Language (DSL) query based on JSONThink as an Abstract Syntax Tree (AST) of queries, consisting of two types of clauses known as leaf query clauses.

4.2.2 Execution

The tool employs a streamlined workflow designed to ensure cohesive data collection, processing, and presentation. This process unfolds in four interconnected stages:

- ***Data ingestion:*** real-time logs and supplementary datasets are ingested directly from deception tools, such as honeypots and DTs. For non-automated sources, a REST API (inherent to the ELK stack) enables manual log submission, ensuring flexibility in data sourcing. Figure 46 depicts data coming from AIAS data sources, and Figure 47 presents how input is ingested into the ELK-based AIAS MAT tool.

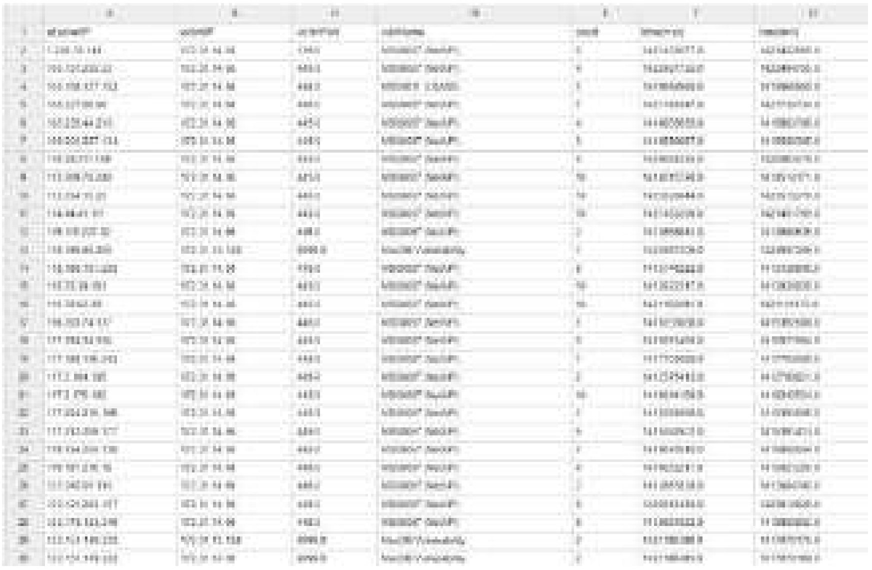
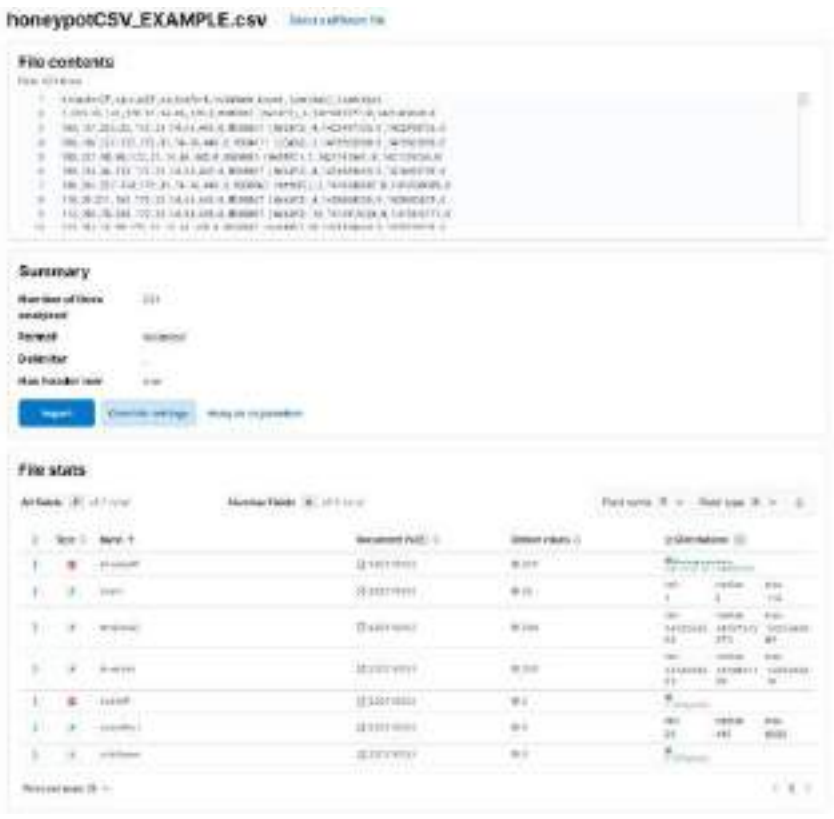


Figure 46. Honeypot CSV_EXAMPLE.csv



honeypotCSV_EXAMPLE.csv

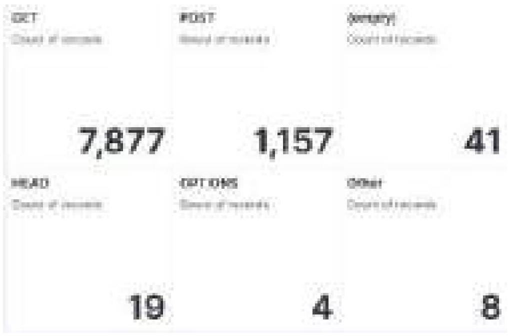
File contents

Summary

File stats

Figure 47. Ingested data

- ***Data harmonization and processing:*** at this stage, raw logs undergo standardization via Logstash pipelines. These pipelines filter extraneous noise, parse unstructured entries, and enrich data with contextual markers transforming fragmented inputs into a cohesive, analysable format. These types of records are visualized as *(empty)* or *other* – see Figure 48.



GET	POST	(empty)
Count of records	Count of records	Count of records
7,877	1,157	41
HEAD	**OPTIONS**	**Other**
Count of records	Count of records	Count of records
19	4	8

Figure 48. Data processing

- ***Analysis and correlation:*** statistical models then interrogate the processed data, identifying patterns (e.g. recurring attack signatures) and anomalies (e.g. unexpected behavioural spikes). Logs are categorized by severity and type, enabling prioritized threat response – see Figure 49.

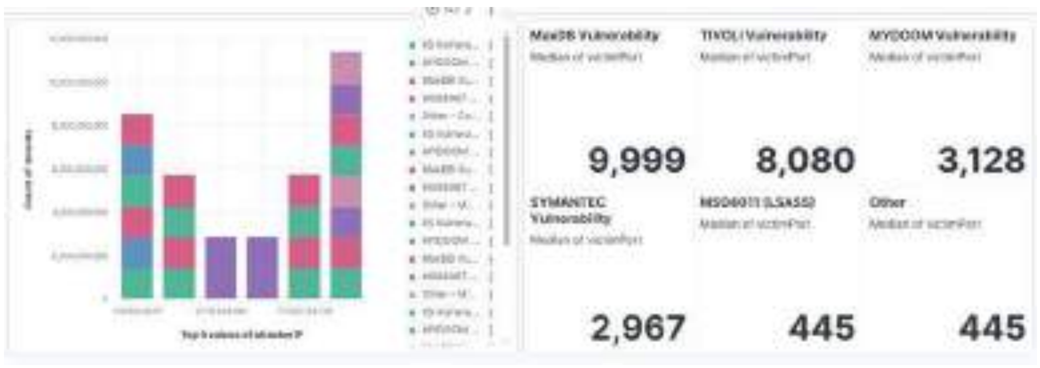


Figure 49. Data analysis

- ***Visualization and reporting:*** customizable Kibana dashboards render these insights visually, offering interactive heatmaps, timelines, and correlation charts. Simultaneously, automated reports distil findings into actionable recommendations for security teams – see Figure 50.

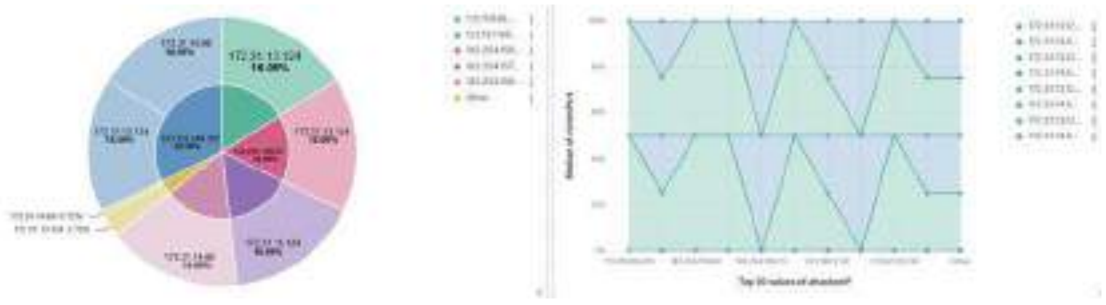


Figure 50. Visualization

4.2.3 Outcomes

The AIAS MAT elevates cybersecurity efficacy through measurable outcomes. First, it grants organizations unparalleled visibility into adversarial tactics, revealing attack methodologies that might otherwise evade detection. Real-time dashboards further empower analysts by synthesizing live data streams into intuitive visuals, fostering proactive threat mitigation and enhancing awareness. Beyond situational awareness, the tool generates actionable intelligence such as risk-prioritized incident summaries to guide strategic decisions, improving the decision making. Finally, its modular architecture ensures adaptability, scaling seamlessly to accommodate evolving organizational needs, such as security auditing and compliance to reports. Together, these capabilities fortify defences against adversarial AI threats, enabling both retrospective analysis and real-time countermeasures for improved attack detection and forecasting.

4.2.4 Communication

Regarding the connection between AIAS MAT and AIAS Deception Layer, such as DT, VP, and HIH, the use of Secure Copy Protocol (SCP) protocol is utilised to share the input/output and receive it. The SCP protocol is used with a simple SSH connection. The deception layer output, after being saved in a CSV format, it is transmitted to AIAS MAT with just a command. The communication consists of three stages, as illustrated in Figure 51.

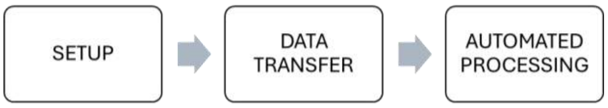


Figure 51. MAT communication stages

AIAS MAT has the port 22 responsible for SSH connections open, but the access is restricted strictly to SCP transfers, and it can be globally connected and receive data from various sources of AIAS Deception Layer tools. For the automated process of receiving the data, (preferably in the CSV format that previously shown using the tool), we use scripts to automatically receive all the files being sent to AIAS MAT to a specific location, so they are directly processed.

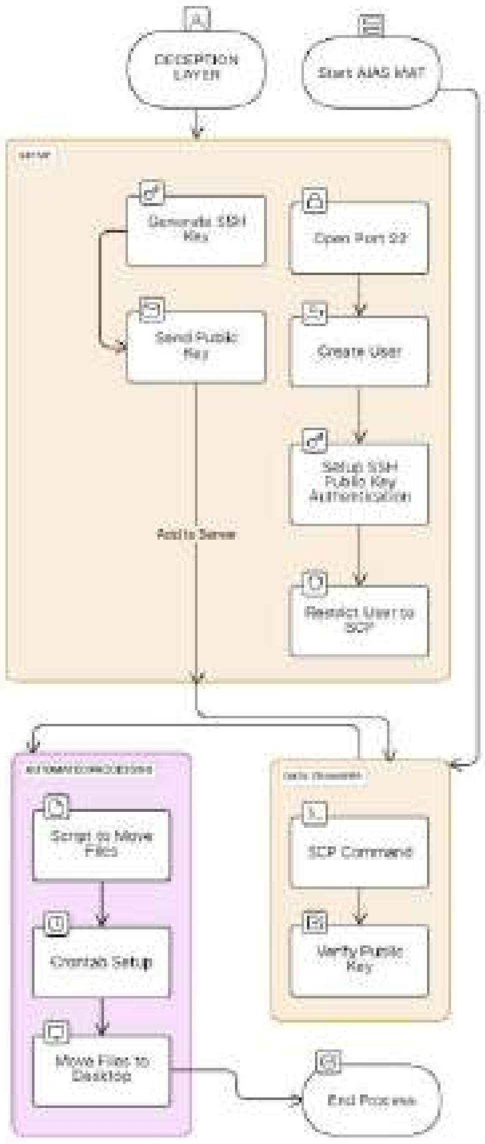


Figure 52. Communication Process

The process of implementing this method is explained in the diagram below to be further analysed before presenting it, so there is an optimized understanding of the scheme.

There are two parts connecting and interacting with each other, the deception layer with the MAT. Firstly, the system has the setup stage where in the MAT there was created a new user and then an SSH public key authentication for that user. After that, as mentioned before, access for security reasons was restricted to SCP. Then the system has the Deception Layer part, where the deception mechanisms need to generate an SSH key (public) and send it to MAT. Thus, it can be added to the server, for the connection with SSH to succeed and getting closer to transferring the data.

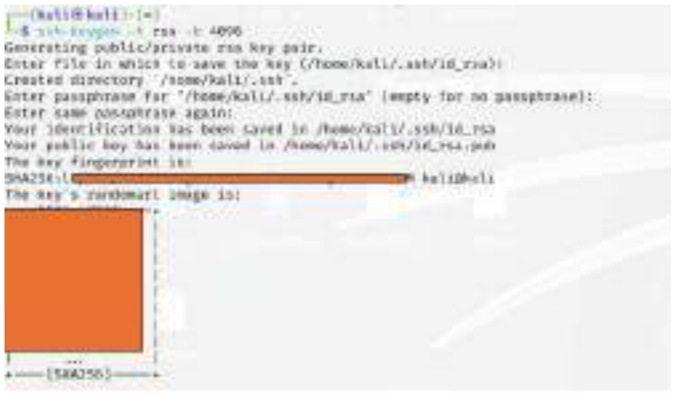
Secondly there is the data transfer stage, where the Deception Layer tools (having their output ready) need to run one simple command in their terminal which will be formatted exactly (see Figure 52). The server (AIAS MAT) then, just verifies that the public key has the matching private key and accepts or rejects the connection. Rejection, therefore, happens only if the public key is not recognized, meaning it is not in the server list of acceptable SSH connections.

Finally, the third stage is automated processing. At this stage, two scripts were created, both for a specific reason providing an extra layer of security and automation. The first script automatically moves files to the specific folder created for the user after the above shown command runs, the path file is /home/datauploader/uploads in AIAS MAT interface. The second script makes use of the Crontab, which is a Linux tool, part of the cron job scheduling system, widely used on Unix-like operating systems. The purpose of Crontab is to define tasks that need to run automatically at specific times or intervals.

The benefits of the Crontab setup have a dual purpose, providing both easier access for the files to be processed on the AIAS MAT Desktop and working as a backup of everything that is being sent. Crontab automatically checks once per hour and moves any CSV files copies from uploads to the desktop folder.

4.2.5 Proof of Work

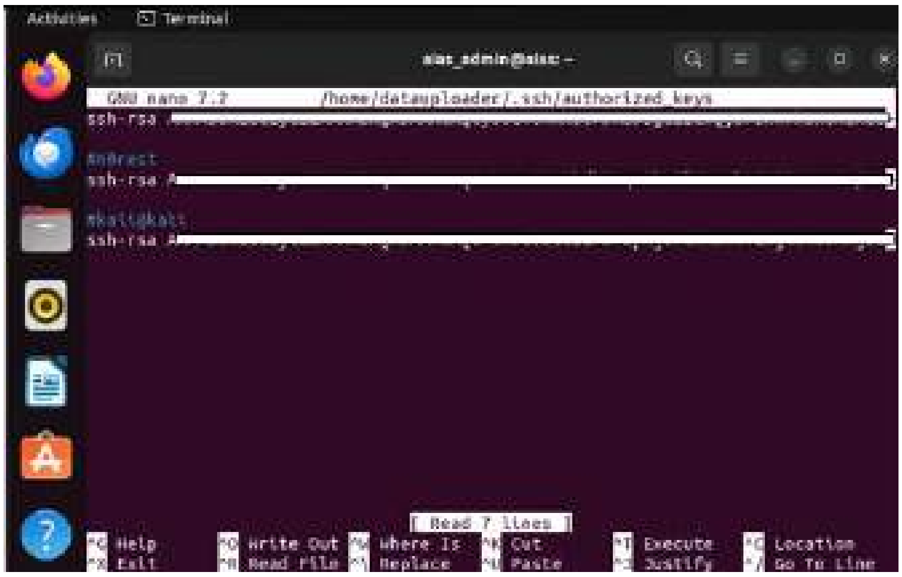
In this section, the steps for the automated process of transferring a file from the deception mechanism tools to AIAS MAT is demonstrated.



```
┌──(kali㉿kali)-[~]
└─$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa):
Created directory '/home/kali/.ssh'.
Enter passphrase for "/home/kali/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kali/.ssh/id_rsa
Your public key has been saved in /home/kali/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:[redacted] kali@kali
The key's randomart image is:
[redacted]
+----[SHA256]-----+
```

Figure 53. Public key creation

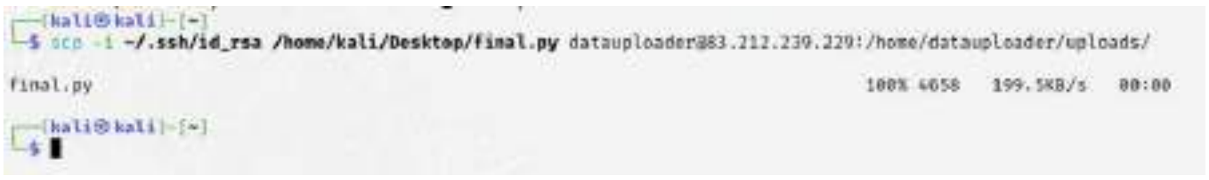
- ***Step 1:*** to create the public key on the machine that want to start transferring files to the MAT (see Figure 53).
- ***Step 2:*** to send the key to AIAS MAT and add it to the *“authorized_keys”* file, so the connection can be automatically approved from now and onwards (see Figure 54).



```
  GNU nano 7.2          /home/datauploader/.ssh/authorized_keys
ssh-rsa [redacted]

an0rest
ssh-rsa A[redacted]

mkali@kali
ssh-rsa A[redacted]




                         [ Read 7 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

Figure 54. “authorized_keys” file

- ***Step 3:*** transference of the file to the MAT tool:



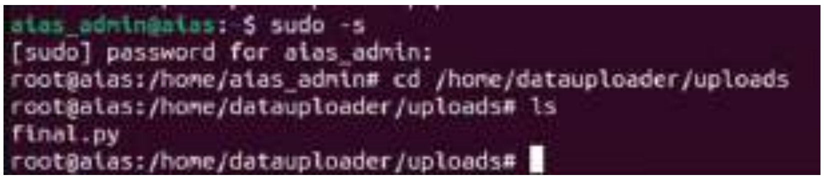
```
┌──(kali㉿kali)-[~]
└─$ scp -i ~/.ssh/id_rsa /home/kali/Desktop/final.py datauploader@83.212.239.229:/home/datauploader/uploads/

final.py                                                          100% 4658    199.5KB/s   00:00

┌──(kali㉿kali)-[~]
└─$ ▮
```

Figure 55. Transference of the file to the MAT tool

The command shown in Figure 55 corresponds to a SCP command used to securely transfer a file from our local machine to the AIAS MAT tool. Specifically, it uses the “*-i”* option to specify a private SSH key located at “*~/.ssh/id_rsa”* for authentication instead of a password (previously generated key).

The file transferred is “*final.py*”, which is located on the local system at “*/home/kali/Desktop/final.py*” as illustrated in Figure 56. The destination is the MAT with IP address 83.212.239.229, and the file is copied into *“/home/datauploader/uploads/”* directory on that server, under the user account datauploader.

The command output confirms that the transfer was successful, showing a 100% completion of the upload of files along with the file size and transfer speed.



```
aias_admin@aias:~$ sudo -s
[sudo] password for aias_admin:
root@aias:/home/aias_admin# cd /home/datauploader/uploads
root@aias:/home/datauploader/uploads# ls
final.py
root@aias:/home/datauploader/uploads# 
```

Figure 56. “final.py” file location

Once the file is transferred, the verification of the uploaded file on the remote server confirms the success. Then, the user *“aias_admin”* must elevate its permissions to root user by using “*sudo -s”* and providing the appropriate password. After gaining the root privileges, the user must navigate to “*/home/datauploader/uploads”* directory using “*cd”* command. After listing the content of the directory with “*ls”*, the user can confirm that “*final.py*” has been successfully received and exists in the expected location. This step helps to validate that the file transfer was completed correctly and the destination path in the server is accessible.

The file appears in the designated directory “*datauploader/uploads”* and will also be visible in the “*desktop/received_csvs*” folder once the scheduled crontab task executes its hourly routine. This demonstrates that the file has been successfully transferred to the AIAS MAT automatically.

5 Conclusions

This deliverable has contributed the AIAS project with four innovative deception solutions, tailored with the use cases defined in previous deliverable D2.2 [D2.2], by exploiting the capabilities of cutting-edge technologies such as DT, VP, and HIH for the protection of AI systems. Additionally, a monitoring tool has been developed to gather information from target AI system and execute smart analytics to generate informative assets regarding the attacks on AI systems.

This contribution would not have been possible without the previous research and study of the current related work, which has provided valuable insights and guidance in the design and development process of each of the proposed solutions listed below.

- The implemented HIH presents a general-purpose honeypot platform based on TPoT that offers capabilities for the simulation and monitoring of common protocols such as SMTP, SSH, Telnet, FTP, HTTP, HTTPs, etc. While the high-interaction approach allows for a deeper engagement with the attackers, the user-friendly data representation provided by Kibana permits administrators to monitor attacks in real time and perform detailed event analysis.

- The Robotic Arm DT presents an industry-based solution that conceptualise the simulation, analysis, and adjustment of motion-controlled devices within critical scenarios. By leveraging advanced simulation and communication technologies such as Unity and ROS, the solution integrates both physical and virtualised robotic components in a smart DT capable of understanding and restoring the proper functioning of the scenario.

- The EIoT Devices DT enables for the smart administration of environmental sensors within critical IoT scenarios. Primarily empowered by real-time synchronisation and analysis of the conditions, the solution can provide alerts and take initiative-taking actions that protect the system from failures or inconsistent states, aiming for the welfare of human actors directly influenced by the environment.

- The Hospital Monitoring VP allows for the simulation and analysis of main healthcare stakeholders, mimicking their behaviour and responses to environmental perturbances. Combined with DT features, the solution aims for improving environmental conditions within this extremely critical infrastructure.

- The MAT comprises all monitoring features withing the deception layer and further analysis of the target systems. While deception strategies detect adversarial activity in AI systems, MAT analyses and visualizes the data to provide actionable insights that strengthen cyber defences, continuously improving the performance and reliability of IT systems.

Therefore, all the implemented components conform the AIAS deception layer, providing a deceptive and informative framework for the virtualisation. Lastly, it is worth noting the specifics proof-of-concept conducted for each solution, validating and highlighting the strengthen and relevance of this deliverable.

References

[AAF] Apruzzese, G., Andreolini, M., Ferretti, L., Marchetti, M., & Colajanni, M. (2022). Modelling realistic adversarial attacks against network intrusion detection systems. Digital Threats: Research and Practice (DTRAP), 3(3), 1-19.

[ABD] Suhaib Abdurahman, Mohammad Atari, Farzan Karimi-Malekabadi, Mona J Xue, Jackson Trager, Peter S Park, Preni Golazizian, Ali Omrani, Morteza Dehghani, Perils and opportunities in using large language models in psychological research, PNAS Nexus, Volume 3, Issue 7, July 2024, pgae245, https://doi.org/10.1093/pnasnexus/pgae245

[ACY] Apruzzese, G., Conti, M., & Yuan, Y. (2022, December). Spacephish: The evasion-space of adversarial attacks against phishing website detectors using machine learning. In Proceedings of the 38th Annual Computer Security Applications Conference (pp. 171-185).

[ADL] Adler, S., Hitzig, Z., Jain, S., Brewer, C., Chang, W., DiResta, R., Lazzarin, E., McGregor, S., Seltzer, W., Siddarth, D., Soliman, N., South, T., Spelliscy, C., Sporny, M., Srivastava, V., Bailey, J., Christian, B., Critch, A., Falcon, R., Flanagan, H., Hamilton Duffy, K., Ho, E., Leibowicz, C. R., Nadhamuni, S., Rozenshtein, A. Z., Schnurr, D., Shapiro, E., Strahm, L., Trask, A., Weinberg, Z., Whitney, C., & Zick, T. (2024, August 15). Personhood credentials: Artificial intelligence and the value of privacy-preserving tools to distinguish who is real online (arXiv:2408.07892). arXiv. https://arxiv.org/abs/2408.07892

[AHM] Ahmad K. Sleiti, Jayanta S. Kapat, Ladislav Vesely, "Digital twin in energy industry: Proposed robust digital twin for power plant and other complex capital-intensive large engineering systems", 2022, Pages 3704-3726, ISSN 2352-4847, https://doi.org/10.1016/j.egyr.2022.02.305.

[AIG] AI Granny by O2 and Re:Scam, https://www.o2.co.uk/inspiration/the-drop/meet-daisy-the-scam-fighting-ai-bot

[AIR] NIST, A risk management framework, https://www.nist.gov/itl/ai-risk-management-framework , Accessed on 28-07-2025

[AJK] Agnihotri, S., Jung, S., & Keuper, M. (2023). Cospgd: a unified white-box adversarial attack for pixel-wise prediction tasks. arXiv preprint arXiv:2302.02213.

[AJT] Ayub, M. A., Johnson, W. A., Talbert, D. A., & Siraj, A. (2020, March). Model evasion attack on intrusion detection systems using adversarial machine learning. In 2020 54th annual conference on information sciences and systems (CISS) (pp. 1-6). IEEE.

[ALC] C. Alcaraz and J. Lopez, "Digital Twin: A Comprehensive Survey of Security Threats," in *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1475-1503, thirdquarter 2022, doi: 10.1109/COMST.2022.3171465.

[ARR] Eclipse Arrowhead A framework and implementation platform for SoS, IoT and OT integration , https://arrowhead.eu/eclipse-arrowhead-2/ , Accessed on 28-07-2025

[ASK] Askell, Amanda, et al. "A general language assistant as a laboratory for alignment." arXiv preprint arXiv:2112.00861 (2021).

[ATA] M. Atalay and P. Angin, "A Digital Twins Approach to Smart Grid Security Testing and Standardization," 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Roma, Italy, 2020, pp. 435-440, doi: 10.1109/MetroInd4.0IoT48571.2020.9138264.

[AZA] Azaria, Amos, and Tom Mitchell. "The internal state of an LLM knows when it's lying." arXiv preprint arXiv:2304.13734 (2023).

[BAI] Bai, Yuntao, et al. "Constitutional ai: Harmlessness from ai feedback." arXiv preprint arXiv:2212.08073 (2022).

[BAJ] Piyush Bajaj, Matthew Edwards, Automatic Scam-Baiting Using ChatGPT (2024), https://arxiv.org/abs/2309.01586

[BAR] B. R. Barricelli, E. Casiraghi and D. Fogli, “A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications,” in IEEE Access, vol. 7, pp. 167653-167671, 2019, doi: 10.1109/ACCESS.2019.2953499.

[BAS] Baston, V. J., and F. A. Bostock. “Deception games.” (1988): 129-134.

[BER] Bertrand, A., Belloum, R., Maxwell, W., & Eagan, J. R. “How cognitive biases affect XAI-assisted decision-making: A systematic review.” In Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society, 1-22 (2022).

[BOT] https://www.rand.org/research/projects/truth-decay/fighting-disinformation/search/items/botometer.html

[BUR] Burns, Collin, et al. “Discovering latent knowledge in language models without supervision.” arXiv preprint arXiv:2212.03827 (2022).

[BUT] Burtell, Matthew, and Thomas Woodside. “Artificial influence: An analysis of AI-driven persuasion.” arXiv preprint arXiv:2303.08721 (2023).

[CAR] Carroll, Micah, et al. “Characterizing manipulation from AI systems.” Proceedings of the 3rd ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization (2023).

[CCQ] Cinelli, Matteo & Cresci, Stefano & Quattrociocchi, Walter & Tesconi, Maurizio & Zola, Paola. (2025). Coordinated Inauthentic Behaviour and Information Spreading on Twitter. 10.48550/arXiv.2503.15720.

[CHA] Chatgpt, https://chatgpt.com , Accessed on 28-07-2025

[CHE] Chen, Dejun, Quanjun Yin, and Kai Xu. “Reverse Thinking Approach to Deceptive Path Planning Problems.” Mathematics (2227-7390) 12.16 (2024).

[CHG] Chongqi Guan, Heting Liu, Guohong Cao, Sencun Zhu, and Thomas La Porta. 2023. HoneyIoT: Adaptive High-Interaction Honeypot for IoT Devices Through Reinforcement Learning. In proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec ‘23). Association for Computing Machinery, New York, NY, USA, 49–59. https://doi.org/10.1145/3558482.3590195

[CHO] Cho, In-Koo, and David M. Kreps. “Signaling games and stable equilibria.” The Quarterly Journal of Economics 102.2 (1987): 179-221.

[CLA] Claude, https://claude.ai/login?returnTo=%2F%3F , Accessed on 28-07-2025

[COH] Cohen, Kobi, and Amir Leshem. “Distributed game-theoretic optimization and management of multichannel ALOHA networks.” IEEE/ACM Transactions on Networking 24.3 (2015): 1718-1731.

[CON] Constâncio, Alex Sebastião, et al. “Deception detection with machine learning: A systematic review and statistical analysis.” PLOS ONE 18.2 (2023): e0281323.

[CRA] Cyber Resilience Act (CRA) | Updates, Compliance, https://www.european-cyber-resilience-act.com/ Accessed on 28-07-2025

[CTP] Cinà, A. E., Torcinovich, A., & Pelillo, M. (2022). A black-box adversarial attack for poisoning clustering. Pattern Recognition, 122, 108306.

[D2.1] AIAS Deliverable D2.1 “Requirements and Reference Architecture”, https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e516b61a44&appId=PPGMS

[D2.2] AIAS Deliverable D2.2 “Specifications and Business cases”

[DAN] Dancy, C. L. “Using a Cognitive Architecture to consider antiblackness in design and development of AI systems.” arXiv preprint arXiv:2022.

[DAV] Davis, Austin L. “Deception in game theory: a survey and multiobjective model.” (2016).

[DEF] DeFake, Available Online: https://defake.app/ , Accessed on 28-07-2025

[DNM] Deldjoo, Y., Noia, T. D., & Merra, F. A. (2021). A survey on adversarial recommender systems: from attack/defence strategies to generative adversarial networks. ACM Computing Surveys (CSUR), 54(2), 1-38.

[EBC] R. Eramo, F. Bordeleau, B. Combemale, M. v. d. Brand, M. Wimmer and A. Wortmann, "Conceptualizing Digital Twins," in IEEE Software, vol. 39, no. 2, pp. 39-46, March-April 2022, doi: 10.1109/MS.2021.3130755.

[EUA] EU AI Act: first regulation on artificial intelligence, European Parliament Online: https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence [Last access: 26/9/2024].

[EVA] Evans, Owain, et al. "Truthful AI: Developing and governing AI that does not lie." arXiv preprint arXiv:2110.06674 (2021).

[FAB] Fabi, S., & Hagendorff, T. "Why we need biased AI - How including cognitive and ethical machine biases can enhance AI systems." Journal of Experimental & Theoretical Artificial Intelligence, 34(3), 1-21 (2022).

[FAR] F. Araujo and T. Taylor. SysFlow: Scalable System Telemetry for Improved Security Analytics. IEEE International Conference on Big Data, 2020.

[FLU] Fluri, Lukas, Daniel Paleka, and Florian Tramèr. "Evaluating superhuman models with consistency checks." 2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML). IEEE (2024).

[GAW] Gawrylowicz, J., et al. "Does practice make the perfect liar? The effect of rehearsal and increased cognitive load on cues to deception." Applied Cognitive Psychology, 30(2), 250-259 (2016).

[GBJ] Gabriel, N.A., Broniatowski, D.A. & Johnson, N.F. Inductive detection of influence operations via graph learning. Sci Rep 13, 22571 (2023). https://doi.org/10.1038/s41598-023-49676-z

[GDP] https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng

[GON] Gonzalez, Cleotilde, et al. "Adaptive cyberdefence with deception: A human–AI cognitive approach." Cyber Deception: Techniques, Strategies, and Human Aspects (2022): 41-57.

[GOO] Goodfellow, I. J., Shlens, J., & Szegedy, C. "Explaining and harnessing adversarial examples." International Conference on Learning Representations (ICLR) (2015).

[HAD] Halawi, Danny, et al. "Overthinking the truth: Understanding how language models process false demonstrations." arXiv preprint arXiv:2307.09476 (2023).

[HAG] Hagendorff, Thilo. "Deception abilities emerged in large language models." Proceedings of the National Academy of Sciences 121.24 (2024): e2317967121.

[HAL] Halabi, Talal, et al. "Protecting the Internet of Vehicles Against Advanced Persistent Threats: A Bayesian Stackelberg Game." IEEE Transactions on Reliability 70.3 (2021): 970-985.

[HAS] Ha, Sean, et al. "Thwarting Adversarial Network Reconnaissance Through Vulnerability Scan Denial and Deception with Data Plane Programming and P4." MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM). IEEE (2023).

[HEG] Hegedűs, C., Tóth, Á., Bancsics, M., & Varga, P. (2024). Multi-agentic Plan&Solve Engine for System of Systems Management via Natural Language. Budapest University of Technology and Economics.

[HUA] Huang, Linan, and Quanyan Zhu. "A dynamic game framework for rational and persistent robot deception with an application to deceptive pursuit-evasion." IEEE Transactions on Automation Science and Engineering 19.4 (2021): 2918-2932.

[HUB] Hubinger, Evan, et al. "Sleeper agents: Training deceptive LLMs that persist through safety training." arXiv preprint arXiv:2401.05566 (2024).

[IMS] Iqbal, Mubashar, Sabah Suhail, and Raimundas Matulevicius. "DECEPTWIN: Proactive Security Approach for IoV by Leveraging Deception-based Digital Twins and Blockchain." Proceedings of the 19th International Conference on Availability, Reliability and Security. 2024.

[ITU] ITU-T. 2023. Y.3091: Digital twin network - Capability levels and evaluation methods. International Telecommunication Union. https://standards.globalspec.com/std/14652768/y-3091

[JAN] Bernard J. Jansen, Joni O. Salminen, Soon-Gyo Jung, Data-Driven Personas for Enhanced User Understanding: Combining Empathy with Rationality for Better Insights to Analytics, Data and Information Management, Volume 4, Issue 1, 2020, Pages 1-17, ISSN 2543-9251,

https://doi.org/10.2478/dim-2020-0005.
(https://www.sciencedirect.com/science/article/pii/S2543925122000560)

[JAV] Javadpour, A., Ja'fari, F., Taleb, T., Shojafar, M., & Benzaïd, C. (2024). A comprehensive survey on cyber deception techniques to improve honeypot performance. Computers & Security, 103792.

[JER] Jericho Security. (n.d.). AI-powered Phishing Simulations. Available Online: https://www.jerichosecurity.com/

[JJS] Jansen, Jim & Jung, Soon-Gyo & Salminen, Joni. (2020). Data-Driven Personas for Enhanced User Understanding: Combining Empathy with Rationality for Better Insights to Analytics. Data and Information Management. 4. 1-17. 10.2478/dim-2020-0005.

[JST] Steinhardt, Jacob. "Emergent deception and emergent optimization." Bounded Regret 19 (2023): 2023.

[KB4] KnowBe4. (n.d.). AI-Enhanced Cybersecurity Awareness Training. Available Online: https://www.knowbe4.com/ , Accessed on 28-07-2025

[KHA] Mohd, Nor, et al. "Recent Developments in Game-Theory Approaches for the Detection and Defence against Advanced Persistent Threats (APTs): A Systematic Review." Mathematics 11.6 (2023): 1353.

[LAB] Lingenfelter, B., Vakilinia, I., & Sengupta, S. (2020). Analyzing Variation Among IoT Botnets Using Medium Interaction Honeypots. 2020 10th Annual Computing and Communication Workshop and Conference (CCWC). doi:10.1109/ccwc47524.2020.9031234

[LEV] Levinstein, Benjamin A., and Daniel A. Herrmann. "Still no lie detector for language models: Probing empirical and conceptual roadblocks." Philosophical Studies (2024).

[LIA] A. Liatifis et al., "SiHoneypot: A Digital Twin-Based Honeypot for Autonomous Vehicles," 2024 13th International Conference on Modern Circuits and Systems Technologies (MOCAST), Sofia, Bulgaria, 2024, pp. 1-4, doi: 10.1109/MOCAST61810.2024.10615785.

[LIK] Li, Kenneth, et al. "Inference-time intervention: Eliciting truthful answers from a language model." Advances in Neural Information Processing Systems 36 (2024).

[LIN] Lin, Stephanie, et al. "Truthfulqa: Measuring how models mimic human falsehoods." arXiv preprint arXiv:2109.07958 (2021).

[LIU] Liu, J., et al. "Deception Maze: A Stackelberg Game-Theoretic Defence Mechanism for Intranet Threats." ICC 2021-IEEE International Conference on Communications. IEEE (2021).

[LOG] Logix, Production vs Research Honeypots: What's the Difference?
https://logixconsulting.com/2020/06/22/production-vs-research-honeypots-whats-the-difference/
Accessed on 25-09-2024

[LQL] Li, B., Qi, P., Liu, B., Di, S., Liu, J., Pei, J., Yi, J., & Zhou, B. (2021, October 4). Trustworthy AI: From principles to practices (arXiv:2110.01167). arXiv. https://arxiv.org/abs/2110.01167

[LUO] Bingqiao Luo, Zhen Zhang, Qian Wang, Anli Ke, Shengliang Lu, and Bingsheng He. 2024. AI-powered Fraud Detection in Decentralized Finance: A Project Life Cycle Perspective. ACM Comput. Surv. 57, 4, Article 96 (April 2025), 38 pages. https://doi.org/10.1145/3705296.

[LWZ] Ling, X., Wu, L., Zhang, J., Qu, Z., Deng, W., Chen, X., ... & Wu, Y. (2023). Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art. Computers & Security, 128, 103134.

[MAD] Ma, Duohe, et al. "Game theory approaches for evaluating the deception-based moving target defence." Proceedings of the 9th ACM workshop on moving target defence (2022).

[MAL] Marco Lucchese, Francesco Lupia, Massimo Merro, Federica Paci, Nicola Zannone, and Angelo Furfaro. 2023. HoneyICS: A High-interaction Physics-aware Honeynet for Industrial Control Systems. In Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES '23). Association for Computing Machinery, New York, NY, USA, Article 113, 1–10. https://doi.org/10.1145/3600160.3604984

[MAN] Mannocci, Lorenzo, et al. Detection and Characterization of Coordinated Online Behaviour. arXiv, 2024, https://arxiv.org/html/2408.01257v1.

[MEN] Mengnan Liu, Shuiliang Fang, Huiyue Dong, Cunzhi Xu. "Review of digital twin about concepts, technologies, and industrial applications" (2021), ISSN 0278-6125

[MET] Meta Fundamental AI Research Diplomacy Team (FAIR)†, et al. "Human-level play in the game of Diplomacy by combining language models with strategic reasoning." Science 378.6624 (2022): 1067-1074.

[MGV] Muñoz-Ortiz, A., Gómez-Rodríguez, C. & Vilares, D. Contrasting Linguistic Patterns in Human and LLM-Generated News Text. Artif Intell Rev 57, 265 (2024). https://doi.org/10.1007/s10462-024-10903-2

[MLY] Morris, J. X., Lifland, E., Yoo, J. Y., Grigsby, J., Jin, D., & Qi, Y. (2020). Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. arXiv preprint arXiv:2005.05909.

[MRA] M.R. Amal, P. Venkadesh, H-DOCTOR: Honeypot based firewall tuning for attack prevention, Measurement: Sensors, Volume 25, 2023, 100664, ISSN 2665-9174, https://doi.org/10.1016/j.measen.2022.100664.

[NGK] M. Nintsiou, E. Grigoriou, P. A. Karypidis, T. Saoulidis, E. Fountoukidis and P. Sarigiannidis, "Threat intelligence using Digital Twin honeypots in Cybersecurity," 2023 IEEE International Conference on Cyber Security and Resilience (CSR), Venice, Italy, 2023, pp. 530-537, doi: 10.1109/CSR57506.2023.10224997.

[NIC] Niclas Ilg, Paul Duplys, Dominik Sisejkovic, Michael Menth, A survey of contemporary open-source honeypots, frameworks, and tools, Journal of Network and Computer Applications, Volume 220, 2023, 103737, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2023.103737.

[NIS] https://www.nist.gov/itl/ai-risk-management-framework

[OGA] O'Gara, Aidan. "Hoodwinked: Deception and cooperation in a text-based game for language models." arXiv preprint arXiv:2308.01404 (2023).

[OHM] Ohmyadd, 2018. wetland - A high interaction SSH honeypot. https://github.com/ ohmyadd/wetland. (Last Commit: 76d296e on 26 December 2018).

[PAC] Pacchiardi, Lorenzo, et al. "How to catch an AI liar: Lie detection in black-box LLMs by asking unrelated questions." arXiv preprint arXiv:2309.15840 (2023).

[PAN] Panda, S., et al. "HoneyCar: A framework to configure honeypot vulnerabilities on the Internet of Vehicles." IEEE Access 10 (2021): 104671-104685.

[PAP] Papernot, N., et al. "Distillation as a defence to adversarial perturbations against deep neural networks." 2016 IEEE Symposium on Security and Privacy (SP). IEEE (2016).

[PAR] Park, Peter S., et al. "AI deception: A survey of examples, risks, and potential solutions." Patterns 5.5 (2024).

[PER] Perez, Ethan, et al. "Discovering language model behaviours with model-written evaluations." arXiv preprint arXiv:2212.09251 (2022).

[PER] Persona QCRI. (n.d.). What is an AI-generated persona? https://persona.qcri.org/blog/what-is-an-ai-generated-persona

[PFE] Pfeffer, Avi, and Ya'akov Gal. "On the reasoning patterns of agents in games." Proceedings of the National Conference on Artificial Intelligence 22.1 (2007).

[QCL] Qiu, H., Custode, L. L., & Iacca, G. (2021, July). Black-box adversarial attacks using evolution strategies. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 1827-1833).

[RAH] Jonas Ricker, Dennis Assenmacher, Thorsten Holz, Asja Fischer, Erwin Quiring, AI-Generated Faces in the Real World: A Large-Scale Case Study of Twitter Profile Images. arXiv, 2024, https://arxiv.org/abs/2404.14244.

[RAP] Rapid7, Honeypots, https://www.rapid7.com/fundamentals/honeypots/ , Accessed on 25-09-2024

[RAS] Rastogi, C., et al. "Deciding fast and slow: The role of cognitive biases in AI-assisted decision-making." arXiv preprint arXiv:2020.

[REA] Ycombinator, Reality Defender, Available Online: https://www.ycombinator.com/companies/reality-defender , Accessed on 28-07-2025

[RES] Re:scam Available Online: https://netsafe.org.nz/rescam , Accessed on 28-07-2025

[ROD] Rodríguez-Alonso, C., Pena-Regueiro, I., & García, Ó. (2024). Digital Twin Platform for Water Treatment Plants Using Microservices Architecture. Sensors, 24(5), 1568. https://doi.org/10.3390/s24051568

[ROW] Rowe, N. C. "Deception in defence of computer systems from cyber-attack." Computers & Security 23.8 (2004): 680-686.

[RPM] Muhammad Irfaan Hossen Rujeedawa, Sameerchand Pudaruth, Vusumuzi Malele. Unmasking AI-Generated Texts Using Linguistic and Stylistic Features. The Scientific World Journal, 2024, https://thesai.org/Downloads/Volume16No3/Paper_21-Unmasking_AI_Generated_Texts.pdf.

[SAC] Sanders, C., & Smith, J. (2014). Chapter 12 - Using Canary Honeypots for Detection. Applied Network Security Monitoring, 317–338. doi:10.1016/b978-0-12-417208-1.00012-x

[SAI] Saikia, P., Dholaria, D., Yadav, P., Patel, V., & Roy, M. (2022, July 28). A hybrid CNN-LSTM model for video deepfake detection by leveraging optical flow features (arXiv:2208.00788). arXiv. https://arxiv.org/abs/2208.00788

[SAY] Sayed, Md Abu, et al. "Honeypot allocation for cyber deception in dynamic tactical networks: A game-theoretic approach." Decision and Game Theory for Security. Springer Nature Switzerland (2023).

[SCH] Scheurer, Jérémy, et al. "Large language models can strategically deceive their users when put under pressure." arXiv preprint arXiv:2311.07590 (2023).

[SIN] Sinha, A. R., et al. "Personalized detection of cognitive biases in actions of users from their logs: Anchoring and recency biases." arXiv preprint arXiv:2022

[SLA] A. El Saddik, F. Laamarti and M. Alja'Afreh, "The Potential of Digital Twins," in IEEE Instrumentation & Measurement Magazine, vol. 24, no. 3, pp. 36-41, May 2021, doi: 10.1109/MIM.2021.9436090.

[SRI] Srivasthav, D. P., & Subudhi, B. N. (2024, November 12). Adaptive meta-learning for robust deepfake detection: A multi-agent framework to data drift and model generalization (arXiv:2411.08148). arXiv. https://arxiv.org/abs/2411.08148

[STE] Stein, Alexander, et al. "Stackelberg evolutionary game theory: How to manage evolving systems." Philosophical Transactions of the Royal Society B 378.1876 (2023): 20210495.

[STR] Strümke, Inga, et al. "Against Algorithmic Exploitation of Human Vulnerabilities." arXiv preprint arXiv:2301.04993 (2023).

[SUH] S. Suhail, M. Iqbal, and K. McLaughlin, ''Digital-Twin-Driven deception platform: Vision and way forward,'' IEEE Internet Comput., vol. 28, no. 4, pp. 40–47, Jul. 2024.

[SXW] Shu, R., Xia, T., Williams, L., & Menzies, T. (2022). Omni: Automated ensemble with unexpected models against adversarial evasion attack. Empirical Software Engineering, 27, 1-32.

[SZF] Karishma Sharma, Yizhou Zhang, Emilio Ferrara, Yan Liu. Identifying Coordinated Accounts on Social Media through Hidden Influence and Group Behaviours. arXiv, 2020, https://arxiv.org/abs/2008.11308.

[THA] Thakoor, Omkar, et al. "Cyber camouflage games for strategic deception." Decision and Game Theory for Security. Springer International Publishing (2019).

[THA2] Thakoor, Omkar, et al. "Exploiting bounded rationality in risk-based cyber camouflage games." GameSec 2020, Springer International Publishing (2020).

[TPO] Tpotce, "T-Pot - The All In One Multi Honeypot Platform", https://github.com/telekom-security/tpotce, (2025)

[TRA] Tramèr, Florian, et al. "The space of transferable adversarial examples." arXiv preprint arXiv:1704.03453 (2017).

[VAI] A. O. Vaidya, M. Dangore, V. K. Borate, N. Raut, Y. K. Mali and A. Chaudhari, "Deep Fake Detection for Preventing Audio and Video Frauds Using Advanced Deep Learning Techniques," 2024 IEEE Recent

Advances in Intelligent Computational Systems (RAICS), Kothamangalam, Kerala, India, 2024, pp. 1-6, doi:10.1109/RAICS61201.2024.10689785

[VAR] Varol, O., Ferrara, E., Davis, C., Menczer, F., & Flammini, A. (2017). Online Human-Bot Interactions: Detection, Estimation, and Characterization. Proceedings of the International AAAI Conference on Web and Social Media, 11(1), 280-289. https://doi.org/10.1609/icwsm.v11i1.14871

[VOL] Volz, K. G., et al. "The neural basis of deception in strategic interactions." Frontiers in Behavioural Neuroscience 9 (2015): 1-12.

[WGL] Wei, X., Guo, Y., & Li, B. (2021). Black-box adversarial attacks by manipulating image attributes. Information sciences, 550, 285-296.

[WIR] Wired. (2024). Real-Time Video Deepfake Scams and Reality Defender. Available Online: https://www.wired.com/story/real-time-video-deepfake-scams-reality-defender/

[WXW] Wang, W., Xu, H., Wan, Y., Ren, J., & Tang, J. (2022, August). Towards adversarial learning: from evasion attacks to poisoning attacks. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 4830-4831).

[WXW1] Wu, S., Xue, J., Wang, Y., & Kong, Z. (2023). Black-Box Evasion Attack Method Based on Confidence Score of Benign Samples. Electronics, 12(11), 2346.

[YAN] Yang, Kai-Cheng & Ferrara, Emilio & Menczer, Filippo. (2022). Botometer 101: social bot practicum for computational social scientists. Journal of Computational Social Science. 5. 1511-1528. 10.1007/s42001-022-00177-5.

[YGA] Y. Gao, S. Qian, Z. Li, P. Wang, F. Wang and Q. He, "Digital Twin and Its Application in Transportation Infrastructure," 2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)

[YKD] Y. Yigit, O. K. Kinaci, T. Q. Duong and B. Canberk, "TwinPot: Digital Twin-assisted Honeypot for Cyber-Secure Smart Seaports," 2023 IEEE International Conference on Communications Workshops (ICC Workshops), Rome, Italy, 2023, pp. 740-745, doi: 10.1109/ICCWorkshops57953.2023.10283756.

[YRW] Yan, S., Ren, J., Wang, W., Sun, L., Zhang, W., & Yu, Q. (2022). A survey of adversarial attack and defence methods for malware classification in cyber security. IEEE Communications Surveys & Tutorials, 25(1), 467-496.

[YSA] Yin, M., Li, S., Song, C., Asif, M. S., Roy-Chowdhury, A. K., & Krishnamurthy, S. V. (2022). Adc: Adversarial attacks against object detection that evade context consistency checks. In Proceedings of the IEEE/CVF winter conference on applications of computer vision (pp. 3278-3287).

[ZAH] Zahid, Iqra et al. Probing the Uniquely Identifiable Linguistic Patterns of AI-Generated Texts. ACL Anthology, 2024, https://aclanthology.org/2024.findings-acl.274.pdf.

[ZHA] Zhang, Tao, et al. "How to disturb network reconnaissance: A moving target defence approach based on deep reinforcement learning." IEEE Transactions on Information Forensics and Security (2023).

[ZHA1] Zhang, R., Wang, F., Cai, J. et al. Digital twin and its applications: A survey. Int J Adv Manuf Technol 123, 4123–4136 (2022)

[ZIE] Ziegler, Daniel M., et al. "Fine-tuning language models from human preferences." arXiv preprint arXiv:1909.08593 (2019).

[ZIY] Wang, Ziyin. (2023). The Influence of the Online Persona on University Students. Journal of Education, Humanities and Social Sciences. 13. 59-66. 10.54097/ehss.v13i.7855.

[ZOU] Zou, Andy, et al. "Representation engineering: A top-down approach to AI transparency." arXiv preprint arXiv:2310.01405 (2023).

[PFB] Petihakis, G., Farao, A., Bountakas, P., Sabazioti, A., Polley, J., & Xenakis, C. (2024, July). AIAS: AI-ASsisted cybersecurity platform to defend against adversarial AI attacks. In Proceedings of the 19th International Conference on Availability, Reliability and Security (pp. 1-7).