Honeyfarm: AI-based Honeypots for Industrial OPC UA Communication via an On-Demand Deployment Architecture

Olaf Sassnick[1,2,3] ✉[0000−0002−3090−4399],
Florian Entleitner[1,2][0009−0004−7622−7401], Dalibor Sain[1,2][0009−0001−4837−9537],
Thomas Rosenstatter[1,2][0000−0002−9587−3423],
Andreas Unterweger[2,3],[0000−0002−3374−1636]
Simon Hoher[1,2][0000−0001−8415−7520], and Stefan Huber[1,2][0000−0002−8871−5814]

[1] Josef Ressel Centre for Intelligent and Secure Industrial Automation
[2] Salzburg University of Applied Sciences, Salzburg, Austria
[3] Paris Lodron University of Salzburg, Salzburg, Austria
✉ olaf.sassnick@fh-salzburg.ac.at

Abstract. Cyberattacks against Operational Technology (OT) and Industrial Cyber-Physical Systems (ICPS) are continuously increasing, necessitating advanced defence mechanisms that are robust, adaptive, and resource-efficient. In this context, we propose an AI-based honeypot with an on-demand deployment architecture for ICPS communicating via OPC UA. The proposed system isolates attackers from production systems by dynamically redirecting malicious traffic to a honeypot instance, that continues from the production system state. This redirection allows continued observation without alerting the attacker while protecting the industrial system. At the core of the architecture is a generative AI model that learns ICPS behaviour from OPC UA communication data. In contrast, existing approaches typically require a manually parametrised simulation model. Our proposed approach enables automated creation of medium-interaction honeypot functionality with minimal manual effort. We further discuss generative AI model scopes and identify usable deployment points in the Purdue network model for practical integration. A proof-of-principle implementation is realised using actual industrial hardware and a fully containerised environment for the honeypot. As part of the deployment, two different firewall platforms are evaluated regarding their provided capabilities. Finally, performance results, including switchover time and interaction latency, are presented to demonstrate the practicality of the approach.

Keywords: Industrial Control Systems · Cyber-Physical Systems · Operational Technology · Security · Honeypots · Generative AI.

This is the authors' pre-publication version of a paper accepted at the SecIndustry Workshop at ARES 2026. The final authenticated version will appear in the conference proceedings of Availability, Reliability and Security – ARES 2026 Workshops, published by Springer in the Lecture Notes in Computer Science (LNCS) series.

1 Introduction

In recent years, industrial plants have evolved due to Industry 4.0 initiatives and ongoing digitalisation, leading to highly interconnected network structures. This enables a higher degree of automation, allowing for better control and optimisation. Additionally, it facilitates flexible manufacturing, making the mass production of customised products possible.

However, this resulted in a drastically expanded attack surface due to the high interconnectedness of Operational Technology (OT) systems [11], with the number of cyberattacks increasing each year [5]. The motives behind cyberattacks on OT vary, for instance, monetarisation via ransomware, espionage, or sabotage by a targeted manipulation of the industrial process, with the most prominent example being Stuxnet [10]. As a result, more advanced defence mechanisms are required, while still respecting the distinct requirements of OT equipment (e.g., real-time requirements, safety requirements, and minimal downtime).

Honeypots are a well-established security technique for detecting and analysing cyberattacks [1]. On-demand deployment is an alternative to permanent honeypots, where decoy systems are instantiated only during suspected threats or active reconnaissance. This approach shifts the honeypot usage from a detection-oriented mechanism towards an early defence mechanism. It offers the advantage that upon activation, the decoy can be tailored to mimic the targeted OT system for a suspected attacker, while the real OT system is simultaneously isolated from the attacker. Ideally, the attacker's interactions are seamlessly transferred to the honeypot. This strategy provides defenders with time to analyse the attacker's entry methods and activities while continuously monitoring their behaviour, which may also yield insights towards their attack objectives. To make such a deployment effective, the interaction time between the attacker and the honeypot must be maximised. The interaction time depends on the honeypot's level of authenticity, as attackers are likely to abandon the network if they suspect they are interacting with a decoy.

To create authentic honeypots for OT systems, they must also replicate the behaviour of an Industrial Cyber-Physical System (ICPS), which interacts with physical processes. One method to achieve this involves integrating simulations into the honeypot, though this approach requires parametrisation and expert knowledge of the underlying processes. An alternative approach leverages generative AI, offering a scalable solution that demands minimal setup resources. Here, a machine learning model autonomously learns to mimic the behaviour of an observed existing ICPS. This method also provides greater flexibility, as it can mimic any ICPS without requiring domain-specific expertise.

Ultimately, rather than deploying a single honeypot, this work envisions a *honeynet* that replicates a live production environment, as illustrated in Fig. 1. This configuration enables attackers to navigate the network and interact with ICPS as they would in an actual production environment, thereby enhancing the authenticity of the decoy. The approach combines two key security principles: sandboxing for isolation and honeypot technology for deception.

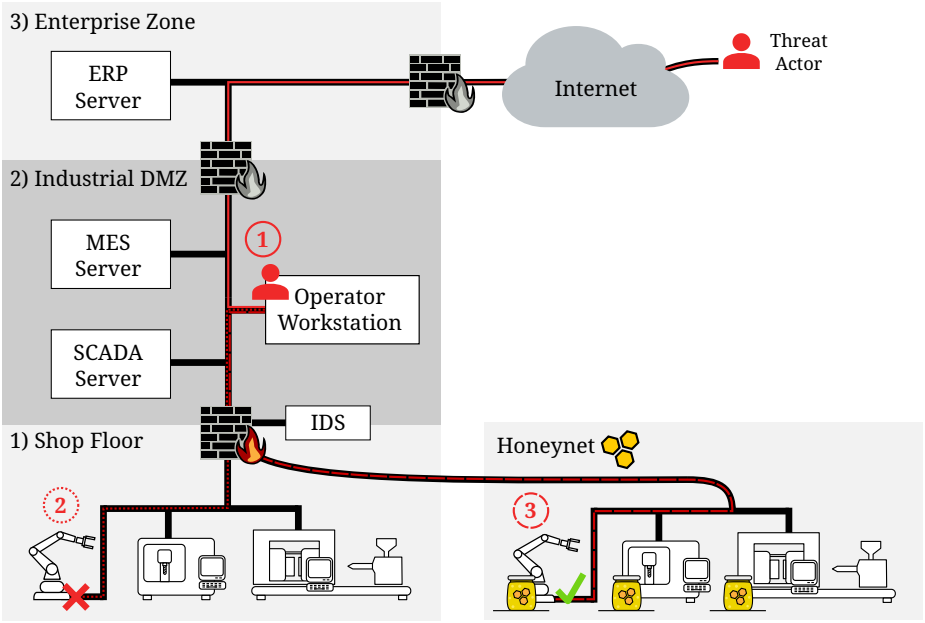

Fig. 1. Overview of a simplified industrial network separated in three zones: the enterprise zone for business operations, the IDMZ as a bridging layer between IT and OT, and the shop floor with a production line. The figure further illustrates the concept of an on-demand honeynet deployment scenario in three steps: (1) an attacker gains remote access to an operator workstation, (2) suspicious interactions by the attacker with the shop floor are detected by an IDS, and (3) the IDS initiates a honeynet and redirects the attacker, as an early response measure, enabling temporary deception and isolation.

Contributions. This work investigates the potential of AI-based honeypots combined with on-demand deployment strategies. We present a proof-of-principle implementation demonstrating honeypot interactions with attackers via Open Platform Communications Unified Architecture (OPC UA), and discuss considerations for honeypot placement. While Maesschalck et al. [9] survey honeypot fit within Industrial Control System (ICS) security architectures and HoneyPLC [13] discusses honeypot deployment, neither provides a consolidated recommendation framework that simultaneously accounts for interaction level, deployment mode (permanent vs. on-demand), and Purdue model. This work addresses that specific gap. Furthermore, we introduce *Honeyfarm*, an architecture for dynamic honeypot deployment that activates decoy systems upon detection of suspicious activity. Our evaluation examines two *Honeyfarm* variants, which are distinguished by firewall capabilities, to provide comprehensive performance metrics, including system responsiveness.

2 Background

2.1 Honeypots

Spitzner defines a honeypot as "a security resource whose value lies in being probed, attacked, or compromised" [24]. Depending on the purpose, two types of honeypots exist: research and production honeypots. Research honeypots mainly exist to collect data regarding attacks. Production honeypots, on the other hand, are typically coupled to an Intrusion Detection System (IDS) with the goal to improve the overall attacker detection rate. Commonly, a honeypot is deployed mimicking a valuable asset with low security measures, where consequently any kind of interaction is suspicious.

Another way to classify honeypots is the level of possible interaction, where low-, mid-, and high-interaction honeypots (LIH, MIH, HIH) are defined. For a low-interaction type it might be sufficient to replay data streams recorded from the existing real system, while for high-interaction honeypots, the intention is to give the attacker full access to the underlying operating system [24]. In between situated are mid-interaction honeypots, which allow a higher degree of interaction than low-interaction honeypots, but where the interaction with the attacker is intended to be restricted to some kind of fake daemon [18], which later on in our case is represented by an OPC UA server.

2.2 Industrial Networks

ICSs typically adopt zone-based architectures, as recommended by IEC 62443 [7]. The model in this work simplifies the Purdue model [26], a six-level framework, by combining, for instance, the process and basic control layers into a single shop floor level, as illustrated in Fig. 1. This simplification aligns with adaptations of the Purdue model under IEC 62443, where zones may be merged if security requirements are preserved.

The **enterprise zone** connects to both the internet and the IDMZ, hosting business operations, like business management software. To retrieve plant data (e.g., production metrics and status updates), these systems access information through the IDMZ [8,7]. The **IDMZ** serves as a secure intermediary zone, hosting the Supervisory Control and Data Acquisition (SCADA) system that monitors, coordinates, and controls shop floor operations. Operators can oversee processes and implement adjustments via Human-Machine Interfaces (HMIs) [8,7]. The **shop floor** contains the ICPSs (i.e., production machines), alongside actuators and sensors. Programmable Logic Controllers (PLCs) in this zone execute real-time control programmes and interface directly with actuators and sensors [8].

Open Platform Communications Unified Architecture. As illustrated in Fig. 1, the communication in the shop floor and to the SCADA system is based on OPC UA. OPC UA is the successor of the OPC standard and standardised in IEC 62541. It has become widely adopted in new products for industrial

environments. Reasons for that being that it is platform-independent, secure-by-design and employs a service-oriented architecture. Moreover, OPC UA supports both, client-server and publish-subscriber as communication model allowing to cover a wide range of use cases [15].

The utilised information model is composed of nodes, i.e., objects, variables, methods, and events, including the references that define their relationships. These nodes are structured within the address space of an OPC UA server. The servers expose the information model, which enables access operations such as reading and writing variable values and invoking defined methods. Each OPC UA node is also uniquely identifiable through a combination of its URI and a distinct node identifier [15].

Having this comprehensive way of representing data and the ability to directly interact with named variables and methods allows attackers to gain much faster understanding (compared to Modbus) of the system. Hence, the servers need to be properly secured against unauthorised access and manipulation. OPC UA employs a security model in two layers. The communication layer utilises TCP/IP and TLS to provide basic security requirements like authenticity and confidentiality, and its application layer offers additionally authentication and authorisation for securing the information model [15].

2.3 Existing Works

Existing works relevant to this paper can be grouped into: generative AI models for honeypot emulation for OT/ICS environments, and on-demand honeypot deployment approaches.

Shan et al. [23] propose NeuPot, a neural network-based honeypot for ICS that uses a trained model to classify and respond to incoming traffic. In contrast to our work, NeuPot targets threat detection rather than sustained attacker deception via a live protocol interface. Vasilatos et al. [25] present LLMPot, which employs fine-tuned large language models to emulate industrial protocols and physical process behaviour across Modbus and S7Comm without manual simulation models. However, LLMPot focuses on protocol-level emulation and does not address on-demand deployment or integration into an OT network.

Park et al. [16] present DVNH, a system that instantiates a new honeypot via NFV and redirects traffic through SDN as soon as a network IDS raises an alert. While conceptually related, DVNH targets IT environments and does not address OT-specific protocols or state continuity upon switchover. Acosta et al. [2] describe CDES, which redirects live traffic using Open vSwitch and container isolation. Similarly to DVNH, CDES is designed for general-purpose and resource-constrained environments, without consideration of industrial protocols or production system state initialisation.

While existing on-demand honeypot approaches do not consider OT-specific requirements such as a transferable system state, and AI-based honeypots have not yet explored on-demand deployment, this work proposes an on-demand architecture tailored to OPC UA-based industrial systems that contributes to both research directions.

3 Research Design

This section describes the research design used to explore the proposed AI-based honeypot approach for an ICPS. Two research questions (*RQs*) are defined to guide this work:

RQ1 How can an AI-based honeypot be designed for an industrial CPS communicating via OPC UA considering attacks on industrial-process-level?

RQ2 How can the on-demand deployment of an AI-based honeypot be realised in an industrial network?

Overall, the research questions are studied in three stages. First, relevant design considerations are laid out and supported by existing literature. In the second stage, the proposed solutions are described on a conceptual level, while in the third stage, the concrete experimental setup is detailed and results are given.

RQ1 focuses on the design of the AI-based honeypot itself. Design considerations regarding the AI-based model are discussed (Section 4.1), followed by a description of the implementation (Section 5.1) and a performance evaluation of the deployed generative model is provided (Section 7).

RQ2 addresses the deployment of honeypots in industrial network environments such as ICS. First, the placement of honeypots in industrial networks is discussed (Section 4.2). Then, a proof-of-principle implementation of an on-demand deployment mechanism is presented (Section 5.2) including the details of the hardware and software components used in the experiment setup (Section 6). Finally, the proof-of-principle is evaluated focusing on the timing of the deployment process (Section 7).

4 Design Considerations

4.1 AI-based Model Design Spaces for Honeypots

When implementing generative AI models for ICPS honeypots, the intended capabilities for interaction with the attacker must be defined at first. Table 1 provides an overview of the interaction capabilities considered, categorised in four interaction levels ranging from raw network packets to the physical process state. The interaction levels only consider active attacker interactions and malicious system manipulation. Passive reconnaissance and data collection can occur through the same interfaces, but only requires a lower level of interaction.

The capability of a generative model to operate at the adversarial interaction levels listed in Table 1 depends on its design space. The model design space defines the input and output dimensions in which the model operates. Originally, it was proposed in the context of human-computer interaction to categorize interfaces for applications of generative models [14]. In this work, we adapt the concept to the interaction between an attacker and a generative-model-based honeypot. A honeypot model can operate at varying levels of abstraction, ranging from the network-socket level to a closely physics-bound level:

Table 1. Active attacker interaction levels that the generative honeypot model can be trained for, ranging from the Communication Interface to the Physical Process level.

Level	Name	Examples
1	Comm. Interface	Malformed messages/packets/frames, invalid payloads
2	Protocol	Invalid requests, forged commands, wrong message sequences
3	Operational	Invalid parameter values, unexpected configuration states
4	Physical Process	Unsafe process trajectories, increased physical system degradation, reduced product quality

Communication Interface Level. The generative model operates on raw communication exchanges observed at the ICPS interface. Depending on the deployment scenario, they may originate from network traffic (such as TCP/IP and UDP packets), fieldbus communication, serial data frames, etc. The model learns ICPS communication patterns directly from message observations and can be trained and respond to malformed messages, invalid payloads, and other communication-layer attacks. A published work on this level can be found in [25]. Functions such as message encryption, transport mechanisms, integrity checks, and protocol-specific parsing may still be handled externally.

Protocol Level. The generative model processes de-serialised protocol messages such as Modbus, Profibus, or OPC UA. The model can learn protocol semantics based on decoded messages and recognise malicious or invalid protocol interactions, which intend to affect the server logic. Examples include write requests to read-only values, invalid message sequences, or referencing non-existing nodes.

Operational Level. The generative model learns the expected operating behaviour of the ICPS based on the variables exposed through the protocol interface. These variables can represent different types of system state, including purely cyber-state information (for example operating mode, number of produced items, power cycles, or computed wear indicators, firmware version) and as well as values derived from the physical process (for example temperature, actuator voltage, oil pressure, spindle speed). The model learns cross-dependencies between variables and plausible operating ranges. Consequently, its interaction capability with an attacker is to respond plausibly to malicious system inputs. Published implementations can be found in [23] or [21].

Physical Process Level. The generative model learns the underlying behaviour of the physical components present in the ICPS. As such, it operates on variables that describe the physical state of the system. An example would be a honeypot internally based on a NeuralODE [4], which, by learning the derivative of the state-space vector, is naturally well suited to describe physical processes.

Table 2 summarises the input and output dimensions of the presented model spaces and maps them to the possible attacker interaction levels from Table 1. The most universal model can be realised on communication interface level, which however as downside can be expected to have the highest complexity and to require the most training. With each level above the model design space

Table 2. Comparison of input and output dimensions for the different model spaces.

Model Design Space Level	Input	Output	Attacker Interact.
Comm. Interface	Raw communication observations	Raw communication responses	Level 1 - 4
Protocol	Deserialised message observations	Protocol-specific responses	Level 2 - 4
Operational	Cyber-physical state observations	Cyber-physical state trajectories	Level 3 - 4
Physical Process	Physical state observations	Physical state trajectories	Level 4

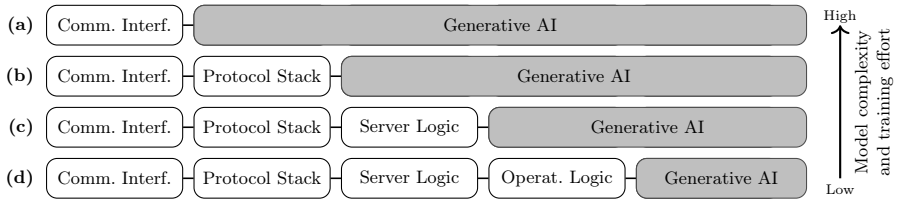

Fig. 2. Illustration of model design space choices for generative AI models as honeypots. (a) is at Communication Interface level, (b) operates at Protocol level, (c) at Operational level and (d) at Physical Process level.

is shrinking, potentially reducing the model size and increasing its robustness, however also limiting its usage possibilities and requiring additional external components to realise a functional honeypot as also visualized in Fig. 2. When focusing on attacks targeting the industrial process level, as indicated by RQ1, the model design space at the operational level provides a balance between model complexity and the requirement for additional external components.

4.2 Placement Criteria for Honeypots in Industrial Networks

The placement of honeypots in industrial networks requires careful consideration of visibility to potential attackers, risk to productive operations, and the quality of the data obtained. In this section we investigate the strategic placement of permanent and on-demand honeypots within OT systems (*RQ2*). Honeypots are often located in a segmented area such as the IDMZ, however, this placement does not reflect all scenarios for placing honeypots in a productive environment. For instance, the placement differs when considering on-demand or permanent honeypots as well as when taking their interaction-level into account. In Table 3 we provide a recommendation for the placement of honeypots in industrial networks based on their interaction-level, deployment mode, and location (consolidated Purdue zones).

Permanent Honeypot Deployment. As summarised in Table 3, the suitability of honeypot deployment varies across network zones. In the enterprise zone, low- and medium-interaction honeypots are well suited, e.g., for simulating email or web services [6], whereas HIHs are not recommended due to their lack of OT relevance and increased attack surface. The IDMZ is often described in the literature as the ideal location for low- and medium-interaction honeypots,

Table 3. Recommended honeypot placement based on their interaction-level in OT.

Deployment Mode	Location	LIH	MIH	HIH
Permanent	Enterprise Zone	+	+	−
	Industrial DMZ	+	+	o
	Shop Floor	o	−	−
On-Demand	Enterprise Zone	+	o	−
	Industrial DMZ	+	+	o
	Shop Floor	+	+	o

Note: (+) Recommended, (o) Limited suitability, (−) Not recommended.

as it is already designed to detect targeted ICS scans without compromising the production environment [6,13]. High-interaction variants are only acceptable in specific scenarios in which a complex OT system can be mimicked. On the shop floor, LIHs can serve as early warning systems in isolated test zones [22], while medium- and high-interaction honeypots are not recommended.

On-Demand Honeypot Deployment. The placement recommendations for on-demand honeypots (see Table 3) reflect the varying suitability across network zones for on-demand deployments. In the enterprise zone, low-interaction on-demand honeypots can be activated at short notice in response to an increased threat level, while medium-interaction variants should only be deployed selectively. High-interaction variants are not recommended, as their safe deployment requires extensive provisioning, segmentation, and monitoring that cannot be reliably ensured under the time constraints inherent to on-demand activation [2]. The IDMZ is described by several studies [2,17] as the optimal location for forensic analysis and active deception, particularly at a medium level of interaction, with high-interaction honeypots being permissible only under strict monitoring, segmentation, and patching, given the elevated risk of compromise and lateral movement [12] when sophisticated adversaries are already actively present in the network. On the shop floor, low-interaction honeypots may serve as temporary deception systems, medium-interaction variants are only advisable under strict monitoring and thorough segmentation [9]. High-interaction honeypots should be reserved for highly controlled scenarios, such as during an active attack provided that operational isolation is in place as a compromised HIH can be a significant risk to the entire network, especially to the OT systems [19].

5 Implementation

5.1 Generative Model with Operational Level Design Space

As discussed in Section 4.1, an AI-based honeypot can be realised in different design spaces. As we previously selected the operational level design space, a multivariate time-series forecasting model can be used to learn the ICPS behaviour. This section provides an overview of the implementation and training procedure for such a honeypot. When implementing a honeypot using the operational level design space, the protocol stack and the system interface are to be

provided externally. As protocol stack, any kind of OPC UA library can be used. The system interface can be generated automatically by traversing the interface of the real ICPS.

Consequently, three steps are required. During the first step, **pre-training**, the interface of the real ICPS is traversed and an interface description is constructed. In the **training** step, a dataset is built by monitoring the interface and sampling data at a fixed sampling frequency F_s. The resulting dataset is then used to train the generative model. The **deployment** describes the step in which the real interface is cloned based on the previously generated interface description.

Typically time-series forecasting models receive an input consisting of n time steps with m variables (the *lookback*) and generate an output horizon of h future time steps with the same m variables [3]. As the honeypot operates in isolation after receiving the initial input from the real ICPS, a recursive forecasting strategy is applied. With this strategy, each generated horizon is subsequently reused as part of the next lookback. At the program level, the honeypot can be implemented using a producer-consumer pattern with two threads and a shared queue. The **producer thread** is responsible for the forecasting process. In this thread, the forecasting model repeatedly generates output horizons of length h and pushes them into the shared queue. The sampling frequency of the training data F_s determines the timing constraints of the system with the required generation frequency given by $F_{gen} = \frac{F_s}{h}$. In the **consumer thread**, the OPC UA server is concurrently operated. It reads new values from the shared queue at frequency F_s and publishes them to the OPC UA interface.

To enable interaction between an attacker and the generative model, a second shared queue is used as a feedback channel. The consumer thread pushes any received OPC UA method calls into this queue. The producer thread periodically checks this queue at frequency F_{gen} and modifies the lookback window accordingly to incorporate input by the attacker.

5.2 On-Demand Deployment via Honeyfarm

The on-demand deployment of honeypots requires the orchestration of several components. The **IDS monitor** continuously receives all new events logged by the IDS. Once suspicious behaviour towards a target host for which a honeypot is available is received, the IDS monitor can forward the information to the Honeyfarm. The **firewall** dynamically enforces network-level forwarding rules. Via an API, a Destination Network Address Translation (DNAT) rule can be activated, which can be used to replace the destination IP address in network packets. This way suspicious traffic can be redirected to a honeypot. The **Honeyfarm** is the component responsible for managing honeypot instances. It provides an API that allows to request the creation and deployment of a honeypot. The Honeyfarm launches a honeypot instance and manages its lifecycle. Lastly, the **data buffer** provides the recent system state of the real ICPS, which is initially required for the honeypot to resume the behaviour of the real ICPS. The data

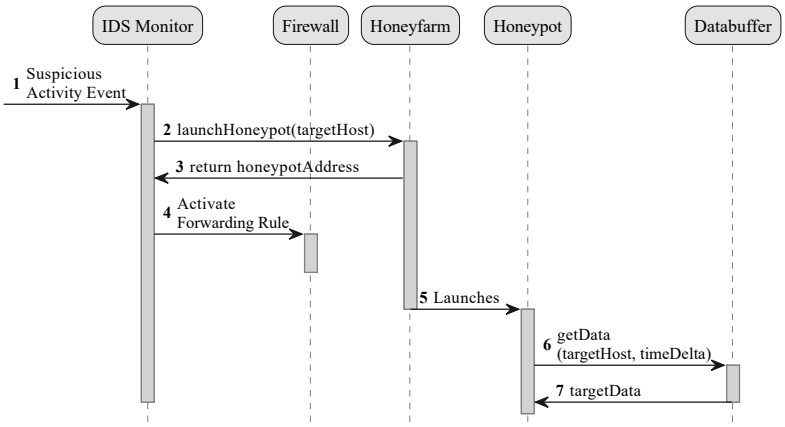

Fig. 3. The deployment sequence for an on-demand honeypot.

buffer subscribes to or continuously polls all relevant nodes of the OPC UA interface and maintains a history of the observed ICPS state. The honeypot can retrieve the latest system data through an API in order to initialise its internal state.

Fig. 3 illustrates the on-demand deployment process as a sequence diagram. It starts with the IDS monitor identifying an attack. The IDS monitor consequently requests a honeypot from the Honeyfarm, including the target host information. The Honeyfarm starts the launching process of the honeypot and returns the honeypot address to the IDS monitor. Consequently, the IDS monitor activates a DNAT rule redirecting the source of the suspicious activity to the honeypot. All other traffic from the source is blocked to guard the network segment.

After the honeypot itself is initialised, it requests the latest state from the data buffer, passing along the target host name, as the data buffer can be responsible for multiple ICPS. The data buffer responds with the latest data, which is then forwarded internally in the honeypot to the generative AI model as initial input. In its current implementation, the honeypot needs to be manually stopped upon inspection, and the number of instantiable honeypots is limited by the Honeyfarm to avoid overload.

6 Experiment Setup

The threat model considered in this scenario is an attacker who gains access to an operator workstation located in the IDMZ, e.g., through social engineering or exploiting a vulnerability. The attacker, in turn, tries to gather information about the OPC UA devices located in the shop floor. Thus, the attacker creates an abnormal number of OPC UA connection requests to the OPC UA server, which ultimately triggers the IDS leading to traffic redirection to an instantiated on-demand honeypot.

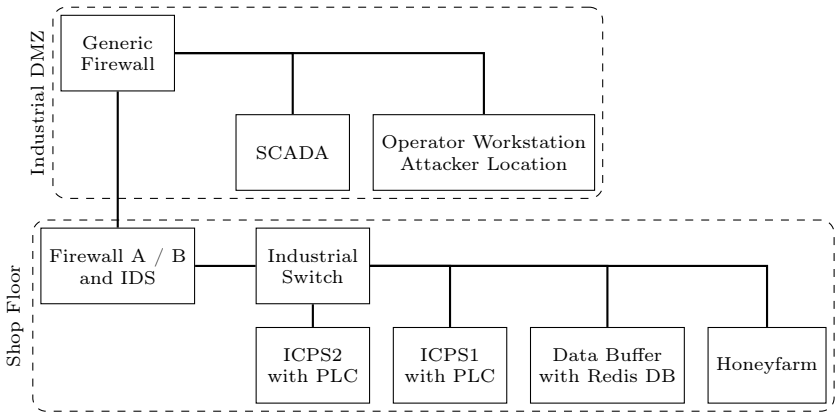

Fig. 4. Overview of the deployment setup of Honeyfarm.

The on-demand activation process for activating the honeypot is evaluated across two different firewalls, which are denoted in the following as firewall A and firewall B. **Firewall A** is combined with an externally hosted Snort IDS, where detection events are relayed to the firewall via a remote REST API. Snort operates on a separate Windows host in the IDMZ and receives traffic via a port mirror on the shop floor switch. In comparison, **firewall B** uses Suricata IDS running directly on the firewall, activating the redirect rule locally. **ICPS1 and ICPS2 with PLCs** simulate the industrial processes being attacked. Both implement the same cyclic procedure and provide an OPC UA interface, where 8 variables are exposed, and additionally method calls are available for the attacker to interact with the Cyber-Physical System (CPS). The cyclic procedure trajectories origin from an actuator resembling a pick-and-place process with two degrees of freedom, and contain actual yaw and pitch angles (Θ, Ψ), actual yaw and pitch angular velocities ($\dot{\Theta}$, $\dot{\Psi}$), and target yaw and pitch angles (Θ_T, Ψ_T), representing the programmed procedure. The **data buffer** is used to collect data from the ICPS1 and ICPS2, which is internally stored in a Redis database. The **honeypot** is implemented according to Section 5.1. The training data is sampled at $F_s = 100$ Hz. The generative AI-model is of Crossformer architecture [27] and is configured with 1600 value lookback and 400 value lookahead. It is trained using a two-pass strategy, as described in [20], to improve the stability. Finally, the **Honeyfarm** manages all Honeypot instances.

The deployment setup of the proof-of-principle depicted in Fig. 4 is based on a three layer industrial network (see Section 2.2). The Honeyfarm, honeypot, and data buffer share the same host, all running within isolated virtual Docker containers. The host system is equipped with an Intel i5 6400 processor and a Nvidia RTX A2000 GPU, while storage is provided via a consumer SATA3 SSD hard drive. Time synchronisation across all systems is ensured using a network time protocol (NTP).

Table 4. Results of each activation phase for both firewalls ($n = 100$ samples).

Metric	Firewall A (ms)	Firewall B (ms)
IDS Detection Delay	$1,942.8 \pm 558.7$	$\mathbf{126.2} \pm 46.3$
Firewall Rule Triggering	650.8 ± 36.1	$\mathbf{200.9} \pm 137.6$
DNAT Activation Time	$3,923.0 \pm 44.1$	$\mathbf{16.7} \pm 0.5$
Honeypot Startup Time	$\mathbf{566.15} \pm 30.42$	
Honeypot Setup Time	$\mathbf{7,608.95} \pm 196.66$	
Model Generation Time	$\mathbf{394.15} \pm 8.5$	

7 Results and Analysis

The on-demand activation process is evaluated across two different firewall architectures (see Section 6). All measurements were collected over $n = 100$ repeated runs per architecture under identical conditions, using an Nmap port scan against the OPC UA port of a PLC. The port scan is only used as an example, as the IDS itself is not in the focus of this work.

The on-demand activation process is further separated into six individually timed phases: The *IDS detection delay* spans from the first packet of the Nmap scan until the IDS logs an alert. The *firewall rule triggering* time covers the period from the logged alert until the firewall acknowledges the redirect command. This follows the *DNAT activation time* measuring how long the firewall requires to apply the redirect rule after receiving the command. In firewall A this is the time of receiving a REST API call and internally activating and loading the redirection rule. In firewall B the rule is written directly into an anchor via a local shell command. The *honeypot startup time* spans from the API trigger received at the honeypot until the Docker container has started successfully. The *honeypot setup time* covers the remaining period until the OPC UA endpoint is reachable by the attacker. The *model generation time* measures the duration from the endpoint becoming reachable until the generative AI model finishes initialising and serves the first dynamically generated value.

Table 4 presents the mean and standard deviation for each phase in the on-demand activation process over $n = 100$ runs. As the honeypot startup, honeypot setup and model generation time are entirely independent of the firewall architecture in use, the measurements for these three phases are aggregated across both test environments into a combined dataset of $n = 200$ runs.

Moreover, three aggregate metrics are derived from these phases (see Table 5). The switchover time is the total duration from IDS alert to honeypot accessibility. The traffic redirection time covers the period from dispatching the DNAT trigger to the honeypot becoming reachable. The overall on-demand process time covers the full span from attack start to honeypot accessibility. All timestamps are synchronised via NTP to allow cross-host correlation.

The most significant architectural differences are observed in the IDS detection delay, which reduces from 1,942.8 ms for firewall A to 126.2 ms for firewall B and in the DNAT activation time, dropping from 3,923.0 ms to 16.7 ms. As the honeypot startup, setup, and model generation phases are architecture-

Table 5. Aggregate end-to-end timing metrics for both firewalls ($n = 100$ samples).

Metric	Firewall A (ms)	Firewall B (ms)
Overall On-Demand Process Time	$10,512.2 \pm 594.0$	$\mathbf{8,896.4} \pm 247.1$
Switchover Time	$\mathbf{8,569.5} \pm 202.0$	$8,770.2 \pm 244.4$
Traffic Redirection Time	$\mathbf{7,918.7} \pm 205.2$	$8,569.3 \pm 202.0$

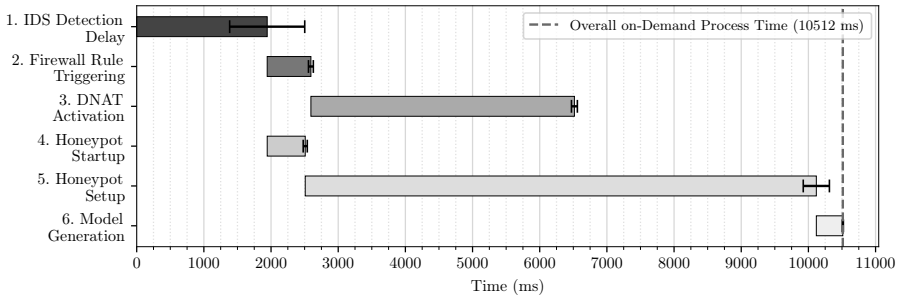

Fig. 5. On-demand activation process for firewall A ($n = 100$ samples).

independent, they share a combined baseline. Figures 5 and 6 show the on-demand activation process for each architecture, averaged over 100 runs.

For variant A the honeypot container and the firewall rule triggering are activated in parallel. The DNAT activation time starts after successful receiving the API call from the external IDS. The honeypot setup phase runs concurrently and takes longer overall, making it the deciding factor for the total process time of $10,512.2 \pm 594.0$ ms.

For firewall B, shown in Fig. 6, detection and rule triggering finish within approximately 126 ms and 201 ms respectively. The `pfctl` anchor is written in under 17 ms. In contrast to variant A the redirect in firewall B is activated before the honeypot startup even completes. The overall on-demand process time of $8,896.4 \pm 247.1$ ms is therefore determined almost entirely by the honeypot startup, setup, and model generation phases.

Although firewall B achieves a shorter overall on-demand process time, Table 5 shows lower switchover and traffic redirection times for firewall A. This is caused by the architectural differences in the timing of the honeypot trigger. In firewall A, the slow IDS detection and rule triggering phases consume approximately 2.6 s before the DNAT trigger is dispatched. During this time, the honeypot has already been starting, so a significant portion of the startup and setup phases runs before the switchover and traffic redirection clocks even begin. In firewall B, detection and rule triggering complete in under 330 ms, meaning the honeypot has barely started when the DNAT trigger is sent. The full startup, setup, and model generation time therefore falls inside the traffic redirection measurement window.

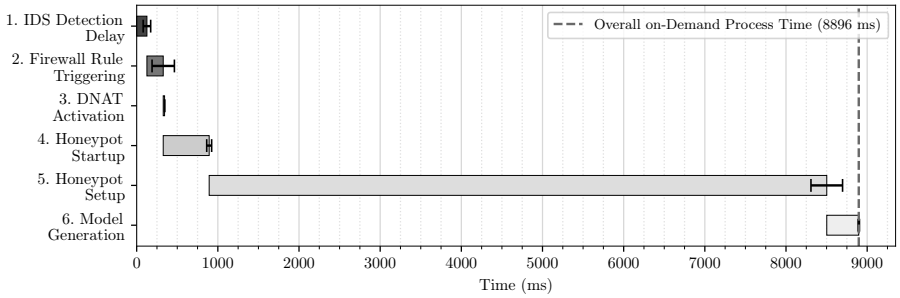

Fig. 6. On-demand activation process for firewall B ($n = 100$ samples).

Table 6. Evaluation results for honeypot stability and interaction. Errors relative to the real CPS are reported as the mean absolute error (MAE) and mean squared error (MSE), averaged across all normalized variables recorded via an OPC UA client. Each column reports the mean ± standard deviation over $n = 100$ samples. For the stability evaluation, each sample consists of a 60 s recording collected after an increasing honeypot deployment time t_{dp} with random start time. For the interaction evaluation, the immediate response of the honeypot to the real CPS is compared to a manual input of the variable Θ_T .

	Stability			**Interaction**
Metric	$t_{dp} = 0\,s$	$t_{dp} = 200\,s$	$t_{dp} = 2000\,s$	
MAE	0.208 ± 0.005	0.209 ± 0.009	0.220 ± 0.003	0.245 ± 0.094
MSE	0.235 ± 0.014	0.242 ± 0.021	0.263 ± 0.007	0.254 ± 0.127

The inference time of the generative model on the host system is on average 0.34 ± 0.11 seconds ($n = 100$ samples). As interaction input is processed during short idle periods between inference, this also represents the lower bound of the responsiveness of the honeypot.

Visual samples of the generated data trajectories are given in Fig. 7. It shows a sample of the generative model output for the programmed procedure and an interaction sample. In both cases, data trajectories of the real CPS are provided for comparison. While the overall time characteristics remain consistent between both, as also indicated by the low MSE and MAE in Table 6, the real CPS exhibits low amplitude higher-frequency dynamics on the angular velocities ($\dot{\Theta}, \dot{\Psi}$), that are not captured by the generative model. A further important aspect is the long-term stability of the generative model. As a recursive forecasting strategy needs to be applied, the generative model is using its previous generated output as input subsequently. For example, a 60 second window of generated trajectories, as shown in Fig. 7 corresponds to 15 recursions. Results regarding the stability as MAE and MSE compared to the real CPS over increased operating time are therefore presented in Table 6 as well.

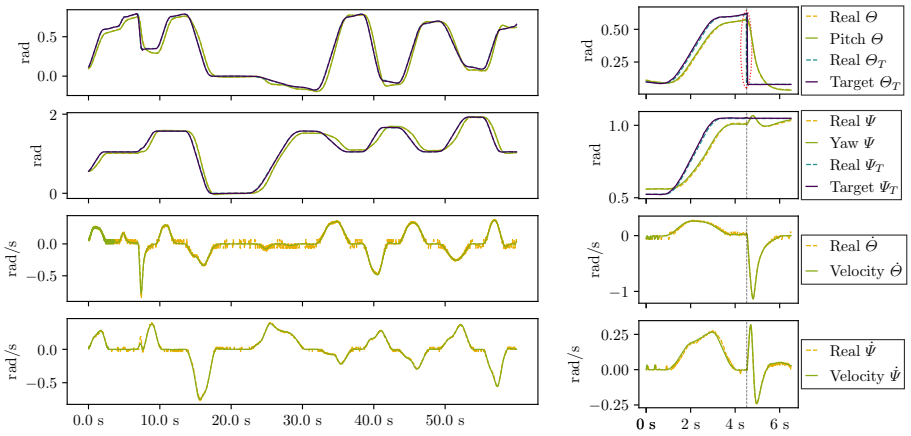

Fig. 7. Visual comparison of the honeypot with the real CPS. **Left:** The programmed procedure defines target angle trajectories Θ_T and Ψ_T. **Right:** The target angle Θ_T is overridden via a method call on the OPC UA interface by the attacker in the circled area. The system response of the real CPS, in a comparable state, is compared to the generated results. With a MAE of 0.217 and a MSE of 0.252 it represents the average quality of the generative model.

8 Conclusion

This paper introduces *Honeyfarm*, a novel architecture for the on-demand deployment of AI-driven honeypots focused on OPC UA communication in ICPS. It bridges the gap of prior work by integrating generative AI-based emulation with dynamic activation in a unified framework. To address *RQ1*, four model design spaces were characterised, with the operational level design space emerging as the most practical choice for emulating ICPS behaviour without domain-specific process models. The generative model initialises from a live system state, which is provided by a data buffer, enabling seamless behavioural continuity with the real ICPS upon activation. For *RQ2*, placement guidelines were established considering interaction level, deployment mode, and a consolidated network model based on Purdue. The proof-of-principle evaluation of Honeyfarm is conducted with two different firewalls. It revealed that the honeypot container initialisation and its configuration dominate the total switchover time, rather than firewall rule activation, with the second firewall model (firewall B) demonstrating superior performance by reducing the total process time to $8,896$ ms, (vs.$10,512$ ms, for firewall A).

Future Work. While overall the proposed architecture is functional, some limitations are present, which are mainly the result of the dynamic forwarding of the attacker to the honeypot. First, currently no TCP/IP session handling is implemented for the switchover. The TCP/IP session terminates during honeypot switchover, potentially enabling the attacker to detect the redirection. To sup-

port common OPC UA security features, for example, the forwarding of existing authentication needs to be realised. Similarly, existing OPC UA node subscriptions need to be forwarded to the launched honeypot. Performance-wise, the system switchover time requires further analysis and optimisation. The primary contributing factor, the honeypot container startup time, could be reduced using higher-performance hardware and storage. An additional optimisation involves forward-streaming the data buffer to bridge the initial model generation time. However, this contributes only marginally to reducing the total switchover time. Finally, the scalability of the AI model requires deeper exploration by deploying the honeypot to mimic more complex ICPS, e.g., a multi-axis industrial robot. Beyond scalability, the lifecycle management of honeypots requires further development, as currently no mechanism exists to terminate the session and expel the attacker once sufficient behavioural data has been collected. Such automated shutdown policies and attacker eviction strategies are necessary for deployment in production.

Acknowledgments. The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the Federal State of Salzburg is gratefully acknowledged. The authors would also like to thank Barracuda Networks for providing complimentary firewall licenses, which supported the experimental infrastructure used in this research.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Security and privacy controls for information systems and organizations. Tech. Rep. NIST SP 800-53 Revision 5, National Institute of Standards and Technology (2020). doi:10.6028/NIST.SP.800-53r5
2. Acosta, J.C., Basak, A., Kiekintveld, C., Kamhoua, C.: Lightweight on-demand honeypot deployment for cyber deception. In: Digital Forensics and Cyber Crime. pp. 294–312. Springer (2022). doi:10.1007/978-3-031-06365-7_18
3. Bontempi, G., Ben Taieb, S., Le Borgne, Y.A.: Machine Learning Strategies for Time Series Forecasting. In: Business Intelligence, vol. 138, pp. 62–77. Springer (2013). doi:10.1007/978-3-642-36318-4_3
4. Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In: NeurIPS (2018)
5. Dragos Inc.: 2026 OT cybersecurity Year in Review: OT/ICS cybersecurity report. Tech. rep., Dragos (2026)
6. Fraunholz, D., Zimmermann, M., Schotten, H.D.: An adaptive honeypot configuration, deployment and maintenance strategy. In: 2017 19th International Conference on Advanced Communication Technology (ICACT). pp. 53–57. IEEE (2017). doi:10.23919/ICACT.2017.7890056
7. International Electrotechnical Commission: Industrial communication networks – network and system security – part 1-1: Terminology, concepts and models. Tech. Rep. IEC/TS 62443-1-1:2009 (2009)

8. Knapp, E.D., Langill, J.T.: Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems. Syngress (2011), ISBN: 9781597496452
9. Maesschalck, S., Giotsas, V., Green, B., Race, N.: Don't get stung, cover your ICS in honey: How do honeypots fit within industrial control system security. Computers & Security **114**, 102598 (2022). doi:10.1016/j.cose.2021.102598
10. Matrosov, A., Rodionov, E., Harley, D., Malcho, J.: Stuxnet under the microscope – revision 1.31. Tech. rep., ESET LLC (2011)
11. Miller, T., Staves, A., Maesschalck, S., Sturdee, M., Green, B.: Looking back to look forward: Lessons learnt from cyber-attacks on industrial control systems. Int. J. of Critical Infrastructure Protection **35** (2021). doi:10.1016/j.ijcip.2021.100464
12. MITRE Corporation: Lateral Movement, Tactic TA0109 – ICS ATT&CK v18 (2025), https://attack.mitre.org/versions/v18/tactics/TA0109/
13. Morales, E.L., Rubio-Medrano, C.E., Doupé, A., Wang, R., Shoshitaishvili, Y., Bao, T., Ahn, G.J.: HoneyPLC: A Next-Generation Honeypot for ICS, vol. 89, pp. 145–181. Springer (2023). doi:10.1007/978-3-031-16613-6_8
14. Morris, M.R., Cai, C.J., Holbrook, J., Kulkarni, C., Terry, M.: The Design Space of Generative Models (2023). doi:10.48550/arXiv.2304.10547
15. OPC Foundation: OPC unified architecture—part 1: Overview and concepts. Tech. Rep. OPC 10000-1 v1.05, OPC Foundation (2025), https://reference.opcfoundation.org/10000-1/
16. Park, B., Dang, S.P., Noh, S., Yi, J., Park, M.: Dynamic virtual network honeypot. In: 2019 Int. Conference on Information and Communication Tech. Convergence (ICTC). pp. 375–377. doi:10.1109/ICTC46691.2019.8939791
17. Pittman, J.M., Hoffpauir, K., Markle, N., Meadows, C.: A taxonomy for dynamic honeypot measures of effectiveness (2020). doi:10.48550/arXiv.2005.12969
18. Pouget, F., Dacier, M., Debar, H.: White paper: honeypot, honeynet, honeytoken: terminological issues. Rapport technique EURECOM **1275** (2003)
19. Salazar, L., López-Morales, E., Lozano, J., Rubio-Medrano, C., Cárdenas, A.A.: ICSNet: A hybrid-interaction honeynet for industrial control systems. In: Proc. 6th Workshop on CPS&IoT Security and Privacy. p. 68–79. ACM (2024). doi:10.1145/3690134.3694813
20. Saßnick, O., Rosenstatter, T., Unterweger, A., Huber, S.: Deep learning-based time series forecasting for industrial discrete process data. In: 8th IEEE Conference on Industrial CPS (ICPS). IEEE (2025). doi:10.1109/ICPS65515.2025.11087869
21. Saßnick, O., Schäfer, G., Rosenstatter, T., Huber, S.: A generative model based honeypot for industrial OPC UA communication. In: Computer Aided Systems Theory – EUROCAST 2024 (2024). doi:10.48550/arXiv.2410.21574
22. Serbanescu, A.V., Obermeier, S., Yu, D.Y.: A flexible architecture for industrial control system honeypots. In: Proc. 12th Int. Conf. Security and Cryptography. pp. 16–26. SCITEPRESS (2015). doi:10.5220/0005522500160026
23. Shan, Y., Yao, Y., Zhao, T., Yang, W.: Neupot: A neural network-based honeypot for detecting cyber threats in industrial control systems. IEEE Trans. Industrial Informatics **19**(10), 10512–10522 (2023). doi:10.1109/TII.2023.3240739
24. Spitzner, L.: Honeypots: tracking hackers. Addison-Wesley (2003), ISBN: 9780321108951
25. Vasilatos, C., Mahboobeh, D.J., Lamri, H., Alam, M., Maniatakos, M.: LLMPot: Dynamically Configured LLM-based Honeypot for Industrial Protocol and Physical Process Emulation. In: 10th European Symposium on Security and Privacy. pp. 963–979. IEEE (2025). doi:10.1109/EuroSP63326.2025.00059

26. Williams, T.J.: A Reference Model for Computer Integrated Manufacturing (CIM): A Description from the Viewpoint of Industrial Automation. Instrument Society of America (1989), http://www.pera.net/Pera/PurdueReferenceModel/ReferenceModel.htm, cIM Reference Model Committee, International Purdue Workshop on Industrial Computer Systems
27. Zhang, Y., Yan, J.: Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: International Conference on Learning Representations (2023)