DOI: 10.1109/ICCCN65249.2025.11133761

Smart Traps for Smart Systems: Scalable Honeynets for IIoT Cybersecurity

Wael Alsabbagh[1,2], Diego Urrego[1], and Peter Langendörfer[1,2]
[1]*Brandenburg University of Technology Cottbus-Senftenberg, Cottbus, Germany*
[2]*IHP – Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany*
E-mail: (Wael.alsabbagh, urregdie, peter.langendoerfer)@b-tu.de

***Abstract*—Honeypots serve as decoy systems that attract and monitor intruders, offering insights into their behavior. When interconnected, these honeypots form honeynets, simulating high-value environments to engage attackers and facilitate deeper analysis. However, in Industrial Internet of Things (IIoT) networks, deploying honeypots presents challenges such as static configurations, lack of network context, and difficulties in scaling.**

In this paper, we introduce *TrapNet*, a scalable, lightweight honeynet framework specifically designed for IIoT environments. *TrapNet* combines compact, on-site honeypots with large-scale, adaptive honeynets deployed on fog and cloud infrastructures using microservices. This approach enables fast deployment, scalability, and flexibility, providing an effective solution for IIoT cybersecurity. Our experimental results show that *TrapNet* efficiently detects intrusions while maintaining low resource overhead. Additionally, all code and configurations used in this study are publicly available, fostering further research and innovation in honeynet design and IIoT security.

***Index Terms*—Honeypots, Cybersecurity, IIoT Security, Fog Computing, Cloud Infrastructure**

I. Introduction

The rapid growth of the Industrial Internet of Things (IIoT) and Industrial Control Systems (ICS) has revolutionized automation and control across critical infrastructure sectors. However, this increased connectivity has also heightened the exposure of these systems to cyber-attacks [1]. Traditional security mechanisms are often ill-equipped to keep pace with the evolving nature of these threats, which necessitates the adoption of more advanced strategies such as anomaly detection and proactive security measures. One promising approach is the use of honeypots [2], which are decoy systems designed to attract attackers and capture valuable insights into their Tactics, Techniques, and Procedures (TTPs). When integrated into a network of decoys, known as a honeynet, honeypots simulate high-value environments, engaging attackers over extended periods and providing deeper insights into their behavior [3].

Despite their effectiveness, deploying honeypots in IIoT environments presents several challenges [4]. Traditional honeypots are often static, isolated from real-world network contexts, and difficult to scale across large, geographically distributed infrastructures. Furthermore, the limited computational resources of IIoT and ICS networks add complexity to the deployment and management of these decoy systems [5], [6]. The emergence of cloud-based and fog computing infrastructures offers a potential solution to these challenges by enabling scalable, flexible, and resource-efficient honeynet architectures that can be centrally managed and dynamically adjusted to the evolving threat landscape.

In this paper, we introduce *TrapNet*, a novel framework designed to deploy honeynets in IIoT environments. *TrapNet* integrates lightweight on-site honeypots with a dynamic, cloud-based honeynet. By leveraging microservices, fog computing, and cloud infrastructure, *TrapNet* enables rapid deployment and seamless scalability, making it an ideal solution for IIoT cybersecurity. The framework is composed of three key layers:

1) **IIoT Remote Site:** This layer includes IIoT devices such as Human Machine Interfaces (HMIs), Programmable Logic Controllers (PLCs), sensors, and acuators. These devices interact with the honeynet via the Honeynet Interface, forwarding malicious activity to the cloud for further analysis.
2) **Honeynet Provisioning Service:** This layer manages the configuration and security of the honeynet and includes:
   - Configuration Backend: that provisions the Honeynet Interface and cloud infrastructure.
   - VPN Gateway: to ensure secure communication between the IIoT site and the cloud.
   - Certificate Authority (CA) and Application Programming Interface (API) Gateway services to enforce secure authentication and communication.
   - Integration with a Logging Service and Security Information and Event Management (SIEM) system for effective intrusion detection and monitoring.
3) **Cloud Service Layer:** The honeynet is hosted in the cloud, utilizing containerized honeypots to emulate ICS environments. This layer supports dynamic scalability, enabling the deployment of honeypots based on traffic volume and attacker behavior.

For the implementation of *TrapNet*, we selected IIoT-relevant honeypot artifacts, including PLCs and MQTT brokers, from two Fischertechnik Lernfabrik systems (24V[1] and 9V[2]). These artifacts were containerized and deployed on

[1]https://www.fischertechnik.de/de-de/industrie-und-hochschulen/technische-dokumente/simulieren/lernfabrik-4-0-24v
[2]https://www.fischertechnik.de/de-de/industrie-und-hochschulen/technische-dokumente/simulieren/lernfabrik-4-0-9v-v-2

cloud infrastructure, allowing for scalable and flexible deployment. A lightweight edge connector securely links the honeypots to the cloud environment, ensuring seamless communication and data flow between on-site IIoT devices and the cloud-based honeynet.

Extensive testing has validated the framework's reliability and effectiveness across various industrial settings. Our experimental results demonstrate that *TrapNet* provides a scalable, flexible, and efficient solution for IIoT cybersecurity. These results underscore its potential, and we have made the framework's code [19] publicly available to encourage further research and development in IIoT security.

The rest of this paper is organized as follows: Section II reviews related work, highlighting existing approaches and their limitations. Section III provides a detailed description of the *TrapNet* framework, including its architecture and key components. Section IV presents the implementation and deployment of *TrapNet* framework, while Section V shows the results and evaluates the framework's performance. Section VI discusses our results as well as findings, and we conclude this paper with Section VII.

II. Related Work

The use of honeypots in ICS and IIoT environments has been extensively explored, with a primary focus on simulating ICS devices such as PLCs and HMIs to capture attacker behavior [7], [8]. While these efforts have provided valuable insights, they are often based on static, isolated setups that do not integrate with real-world IIoT networks. This limits their scalability and effectiveness. In contrast, the *TrapNet* framework offers a dynamic, scalable solution by integrating key IIoT components—such as MQTT brokers and SCADA servers—through cloud infrastructure, providing greater flexibility and fidelity in capturing cyber threats.

Cloud-based honeypots, such as those deployed on platforms like Amazon Web Services (AWS), offer scalability but face challenges specific to IIoT, such as edge device constraints and hybrid infrastructure integration [9], [10]. *TrapNet* addresses these challenges by combining lightweight, on-premise honeypots with cloud-based microservices. This hybrid approach allows for the localized capture of attacker traffic, which is then securely forwarded to the cloud for further analysis. As a result, *TrapNet* ensures both scalability and accurate IIoT network emulation.

Modular and containerized honeypot systems have also been proposed to enhance scalability and flexibility [11]. However, these systems often overlook practical deployment challenges, such as secure communication and automated provisioning. *TrapNet* improves on these frameworks by incorporating secure Virtual Private Network (VPN) tunneling, automated provisioning, and integration with SIEM systems. These features provide a comprehensive, resource-efficient honeynet solution tailored for IIoT environments.

In summary, while existing research offers valuable foundations, many honeypot solutions fall short of addressing the combined requirements of scalability, hybrid integration, and real-time adaptability in IIoT systems. *TrapNet* overcomes these limitations by providing a dynamic, secure honeynet framework that seamlessly integrates on-premise and cloud components to enhance IIoT cybersecurity.

III. *TrapNet* Framework Architecture

The architecture of the *TrapNet* framework, illustrated in Figure 1, is optimized to minimize the impact on IIoT networks while leveraging cloud infrastructure for robust data processing, analysis, and scalability. At the core of this architecture is the Honeynet Interface, a lightweight, virtualized component that resides within the IIoT network. This interface serves as the primary traffic sink, intercepting and capturing all network traffic directed toward the honeynet, which is then securely routed to the cloud or fog network for further analysis and emulation. Crucially, the honeynet remains isolated from the production network, ensuring that no malicious data can propagate back to critical industrial systems. This isolation is maintained by preventing any outbound communication from the honeynet to the IIoT environment, safeguarding the operational integrity of the network.

The *TrapNet* framework consists of three key layers, each addressing specific challenges in IIoT cybersecurity: 1) IIoT Remote Site, 2) Honeynet Provisioning Service, and 3) Cloud Service Layer. Below, we provide a detailed explanation of each layer.

A. *IIoT Remote Site*

This layer represents the physical on-site environment of the IIoT, which includes a wide range of industrial devices such as HMIs, PLCs, sensors, actuators, SCADA systems, and message brokers like MQTT brokers. These devices interact with the honeynet via the Honeynet Interface, which captures all network traffic, including any malicious or anomalous activity. The Honeynet Interface forwards this data securely to the cloud-based honeynet for further analysis and emulation.

Technically, the Honeynet Interface is designed as a low-overhead virtual component that can be deployed on existing IIoT infrastructure without significant performance degradation. It supports multiple network protocols and integrates seamlessly with IIoT-specific protocols such as Modbus, OPC-UA, and MQTT to ensure comprehensive threat detection. Additionally, the interface supports advanced traffic filtering techniques, enabling it to selectively capture suspicious traffic patterns indicative of known or unknown cyberattacks, including scanning, exploitation attempts, and lateral movement.

B. *Honeynet Provisioning Service*

This layer serves as a critical intermediary layer that manages the security, configuration, and orchestration of the honeynet components. This service enables the dynamic and efficient deployment of the honeynet and ensures seamless communication between IIoT devices and cloud-based honeypots. Key components of this layer include:

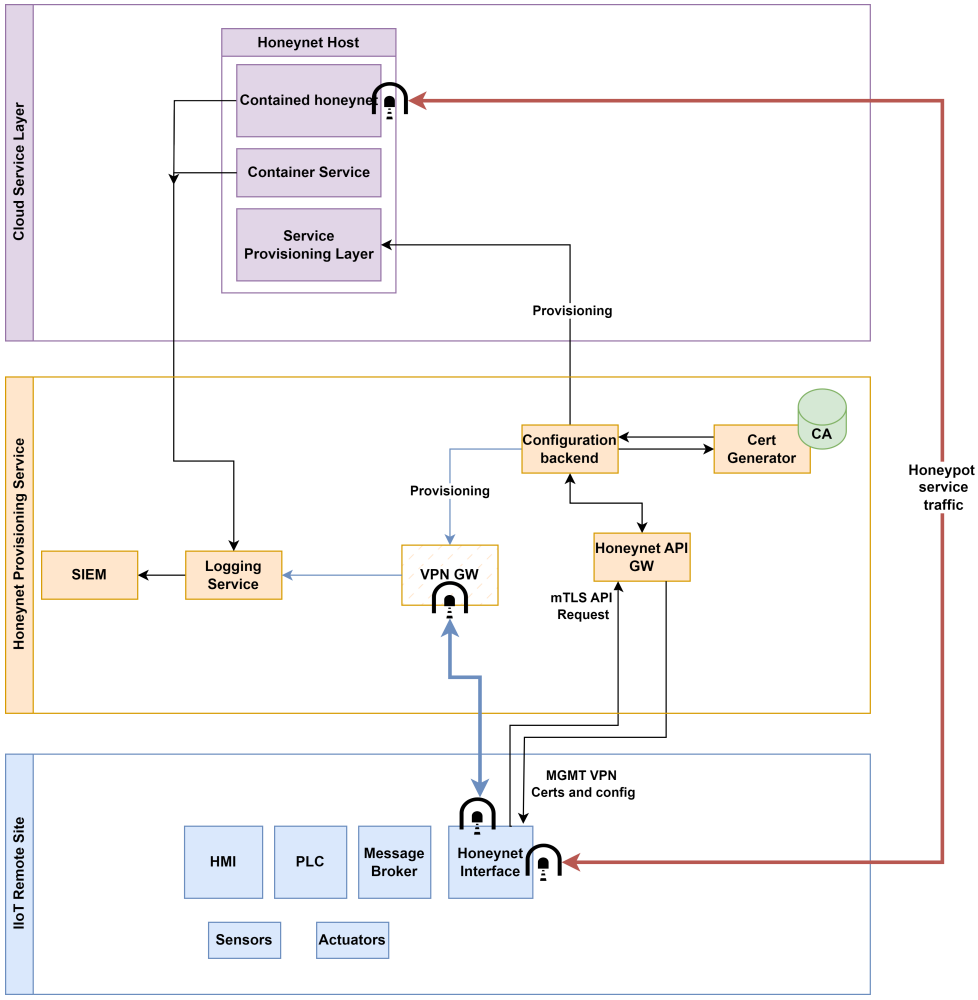

Fig. 1: *TrapNet* Architecture: Interaction Across the Three Layers.

1) ***Configuration Backend****:* This component automates the setup and provisioning of the Honeynet Interface and cloud-based infrastructure, handling tasks such as network configuration, container orchestration, and resource allocation. Tools like *Ansible* and *Terraform* are used to ensure consistent, automated deployments across geographically distributed IIoT sites. The backend is designed to scale with the number of deployed sites, enabling *TrapNet* to operate efficiently in large industrial environments.

2) ***VPN Gateway****:* This component establishes and maintains secure, encrypted communication channels between the IIoT remote site and the cloud honeynet. It uses standard encryption protocols (e.g., IPsec or WireGuard) to ensure data confidentiality, integrity, and authenticity during transit. The VPN gateway also manages the dynamic assignment of IP addresses and network routes, ensuring that only authorized devices can connect to the honeynet.

3) ***CA and API Gateway****:* This system enforces strong authentication mechanisms through mutual Transport Layer Security (mTLS) protocols, ensuring that only trusted devices and services can communicate with the honeynet. The CA is responsible for managing digital certificates, while the API Gateway ensures secure communication between system components by acting as a proxy that verifies access rights and logs API interactions.

4) ***Logging Service and SIEM Integration****:* This component is responsible for capturing and storing logs from all system activities, including honeynet interactions, traffic patterns, and security events. These logs are forwarded to a SIEM system, which analyzes the data to detect security anomalies, track potential intrusions, and generate actionable threat intelligence. The integration of real-time logging and SIEM ensures that any attack or abnormal behavior is immediately identified and responded to.

C. Cloud Service Layer

The cloud service layer hosts the honeynet itself, enabling flexible and scalable deployment of honeypots to emulate a variety of IIoT devices and systems. This layer is designed to dynamically allocate resources, adapt to varying attack patterns, and scale based on traffic volume. Key components of this layer include:

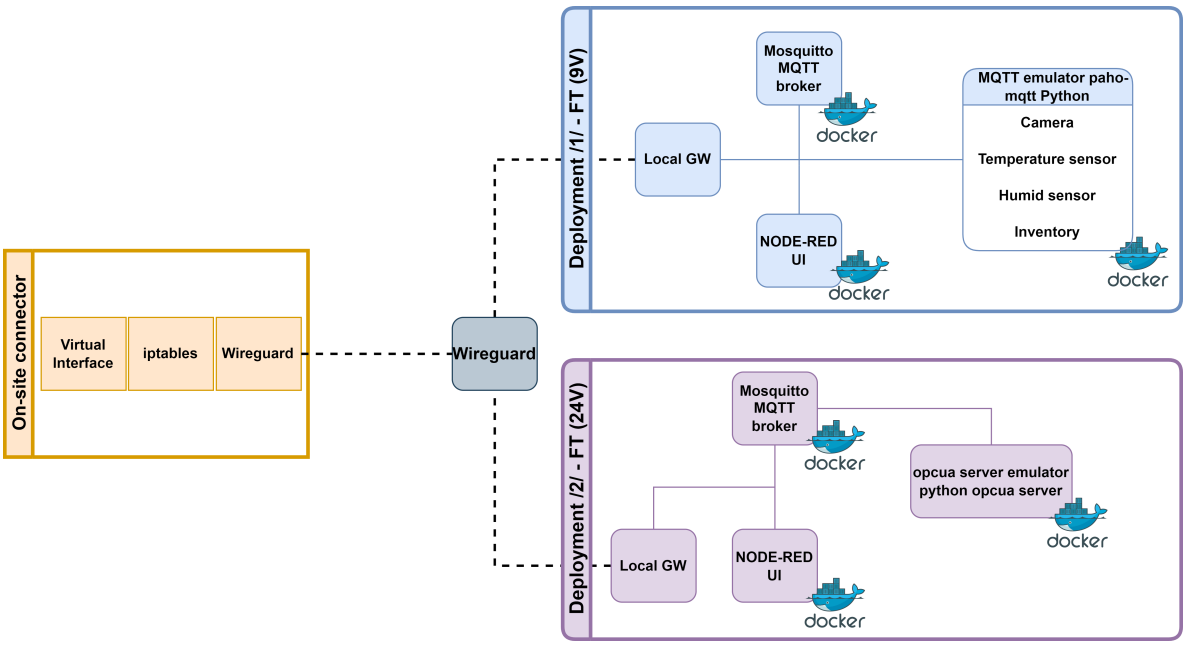

Fig. 2: *TrapNet* Deployment Architectures: 1) Fischertechnik Lernfabrik 9V. 2) Fischertechnik Lernfabrik 24V.

1) ***Containerized Honeynet****:* At the core of this layer are containerized honeypots, which emulate real IIoT devices such as PLCs, MQTT brokers, SCADA servers, and sensors. These honeypots are designed to mimic the behavior and vulnerabilities of industrial systems, providing attackers with the opportunity to interact with them in ways that offer valuable insights into attack methods and techniques. Containerization enables efficient deployment and management of these honeypots, allowing for isolation while sharing resources within the cloud infrastructure.

2) ***Container Service****:* This service manages the deployment, orchestration, and lifecycle of honeypots. It ensures that the appropriate resources are dynamically allocated based on real-time traffic analysis and attack behavior. Container orchestration tools like *Kubernetes* are used to automatically scale honeypot resources, ensuring that the system effectively responds to changing threat landscapes.

3) ***Service Provisioning Layer****:* This component automates the rapid provisioning and scaling of honeynet resources in response to evolving cyber threats. It allows for horizontal scaling (by adding more honeypots) or vertical scaling (by increasing the capacity of existing honeypots) depending on the volume of interactions or the severity of attacks. Cloud-native services, such as AWS Lambda or Google Cloud Functions, are employed for event-driven scaling, ensuring that the system can handle traffic spikes without overloading individual components.

In conclusion, *TrapNet*'s multi-layered architecture integrates advanced features such as containerization, automated provisioning, and secure communications to create a dynamic, scalable, and highly effective solution for IIoT cybersecurity. By incorporating real-time traffic analysis, SIEM integration, and machine learning-driven anomaly detection, *TrapNet* ensures that IIoT networks can swiftly adapt to emerging threats, providing both real-time threat engagement and long-term cybersecurity resilience.

IV. Implementation and Deployment of *TrapNet*

The *TrapNet* system is designed to simulate a training IIoT platform, replicating essential components of a real IIoT environment. The implementation involves two honeynet deployments, each capable of being deployed in either a fog or cloud network, depending on specific requirements. Both deployments are modeled after the Fischertechnik Lernfabrik 24V and 9V systems.

Each component of the deployment is encapsulated as a standalone Docker container, promoting modularity and scalability. This architecture allows for the integration of additional elements, such as SCADA systems, data analysis modules, and logging tools, to meet specific IIoT simulation needs [12].

Figure 2 illustrates the architecture of the two primary deployment setups:

- **Deployment /1/:** Based on the Fischertechnik Lernfabrik 9V, this setup includes components such as an MQTT broker, temperature and humidity sensors, and a Node-RED user interface.
- **Deployment /2/:** Built on the Fischertechnik Lernfabrik 24V, this configuration features an OPC-UA server emulator for simulating PLC communication, along with similar core components.

Both setups employ a virtual interface on the on-site connector, securely routing traffic through Iptables and a WireGuard VPN to the cloud infrastructure.

A. *Developed IIoT Components*

While many components, such as MQTT brokers and Secure Shell (SSH) honeypots, utilize existing Docker images, certain honeynet components required custom development. These include sensors and actuators designed to simulate realistic IIoT applications, ensuring the honeynet appears authentic and is not easily detected by attackers [13].

*1) **MQTT Sensors and Actuators**:* The emulated sensors generate realistic industrial metrics—such as air quality, humidity, temperature, and pressure—and publish them to the MQTT broker. This behavior mimics that of real IIoT sensors, enhancing the honeynet's deception capability. The workflow, shown in Figure 3, demonstrates the process from initialization and metric generation to the publishing of data via MQTT.

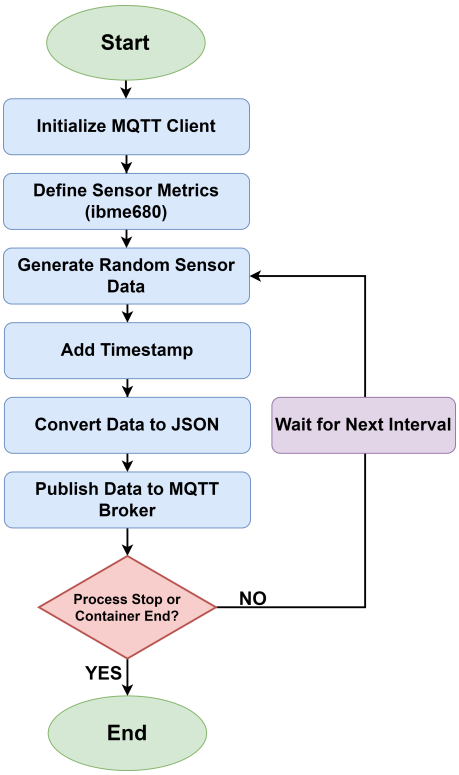

Fig. 3: Sensor Emulation Workflow.

*2) **Camera Emulator**:* The camera emulator simulates the Fischertechnik HMI-controlled camera. It processes movement commands sent via MQTT and responds with a Base64-encoded image of the IIoT environment. Figure 4 outlines its functionality, from MQTT subscription to command execution and image transmission.

*3) **OPC-UA Emulation Server**:* In deployment /2/ (Fischertechnik 24V), the Siemens PLC communicates via OPC-UA. To replicate this functionality, we developed a containerized OPC-UA server. This server emulates PLC operations, such as variable reading and configuration changes.

The emulation process involves two main steps:

- **Data Harvesting:** A script named *opc_collector.py* utilizes the *Python-OPC-UA* library to explore the structure of the real PLC's data nodes. The script collects detailed information such as variable names, values, and configurations, creating a dataset that accurately represents the PLC's functionality. This data is stored for use during the emulation phase.
- **Server Emulation:** Another script, *server.py*, uses the harvested data to deploy it in a virtual OPC-UA server. The server behaves like a real PLC, offering realistic responses to client queries. For example, attackers attempting to read or modify variables will interact with the emulated server, which responds as though it were an authentic PLC.

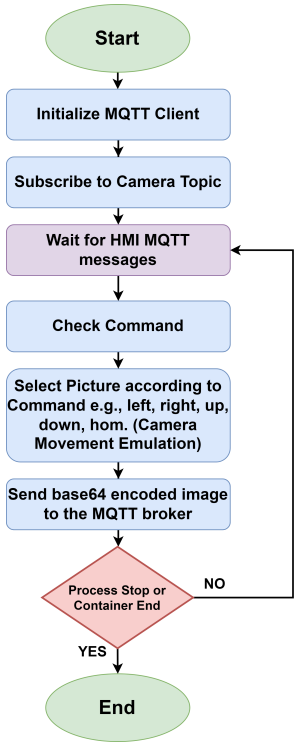

Fig. 4: Camera Emulation Workflow.

By presenting a convincing façade, this setup lures attackers into revealing their tactics without risking real infrastructure. This emulated server provides a safe yet realistic target for cyberattacks, enabling the system to analyze attacker behavior and collect valuable intelligence for securing IIoT environments.

V. Evaluation & Results

The evaluation of the *TrapNet* framework focused on three primary aspects: functionality testing, performance benchmarking, and simulated attack scenarios. These evaluations demonstrate the framework's ability to replicate IIoT environments accurately, deliver realistic response times, and capture detailed attack traces.

A. *Functionality Testing*

A series of functionality tests, presented in Table I, validated the interaction between various honeynet components, includ-

TABLE I: Actions Tested for Various Elements in the Honeynet Deployments.

Tested Element	Target	Actions Tested
SSH	Cowrie	Login
		Shell commands
MQTT	Mosquitto broker	Connect
		Authenticate
		Ping
		Subscribe
		Unsubscribe
HMI (MQTT)	Sensor/Actuators	Write data
		Receive data
OPC-UA	PLC	Session request
		Session disconnect
		Variable read
		Write request

ing SSH, MQTT, HMI, and OPC-UA services. These tests ensured that the deployed honeypots accurately simulated IIoT devices and supported a wide range of attack scenarios [10], [13], [15]–[17]. The results confirmed seamless interactivity and compatibility, replicating the behavior of genuine IIoT systems under both user and adversary interactions.

B. Performance Evaluation

Performance was assessed by measuring response times across different environments. These tests compared the performance of the honeynet deployed in: 1) the cloud, 2) a local honeypot connected directly to the local interface i.e., fog network, and 3) the actual Fischertechnik Lernfabrik. The goal was to evaluate how similar the user or attacker experience is when interacting with the honeynet. Long delays or poor performance could undermine the honeynet's deception capabilities, emphasizing the need for realistic response times.

*1) **MQTT Performance**:* Figure 5 illustrates the response times of MQTT operations across different deployments. The cloud honeynet deployment shows slightly higher response times compared to the local fog network deployment, which more closely mirrors the actual Fischertechnik Lernfabrik. These findings show that the fog deployment offers the best balance between performance and realism.

*2) **OPC-UA Performance**:* As shown in Figure 6, the OPC-UA response times for the cloud deployment are slightly higher compared to the fog network deployment. While the cloud deployment excels in scalability, the fog deployment provides a more realistic response time suitable for immediate interactions. This flexibility ensures the honeynet remains effective regardless of deployment conditions.

C. Attack Scenario Evaluation

The simulated attack scenario validated the honeynet's ability to detect and log adversarial actions. The attack process consisted of multiple phases:

- **Network Access**: The attacker connects to the network and begins reconnaissance activities.
- **Scanning and Fingerprinting**: The attacker scans and fingerprints the network, interacting with decoy IP addresses assigned to *TrapNet*. This triggers logging and real-time alerts.
- **Device Enumeration**: The attacker identifies both *TrapNet* components and legitimate IIoT devices, often focusing on *TrapNet* due to perceived vulnerabilities.
- **Service Interaction**: Interactions include port scans, brute-force attacks, and authentication attempts. Detailed logs of these activities are captured and forwarded to the cloud logging service.
- **Service Exploitation**: The attacker executes commands and explores vulnerabilities within the honeynet, such as accessing MQTT topics or interacting with the HMI. Every action is meticulously logged for post-attack analysis.
- **Attack Logs and Insights:** Table II provides examples of logs generated during the attack, illustrating the detailed traceability of adversarial actions.

VI. Discussion and Findings

The evaluation of the *TrapNet* framework demonstrated its capability to simulate realistic IIoT environments and detect adversarial activities, with a particular emphasis on its unique contributions to IIoT security.

A. Contributions of TrapNet Framework

One of the key innovations of the *TrapNet* framework is the deployment of honeypots that emulate a variety of IIoT services e.g., SSH, MQTT, HMI, OPC UA, etc. within a scalable architecture that can operate both in cloud and fog network environments. By leveraging containerization, *TrapNet* achieves a modular and flexible design that allows for dynamic provisioning and seamless interaction between emulated and real devices.

Another aspect is the integration of custom-developed components such as MQTT sensors, actuators, and OPC UA servers, which are crucial for maintaining the realism of the simulated IIoT environment. These components enhance the honeynet's ability to deceive attackers and make detection more difficult, providing a high level of engagement in both the cloud and fog deployments.

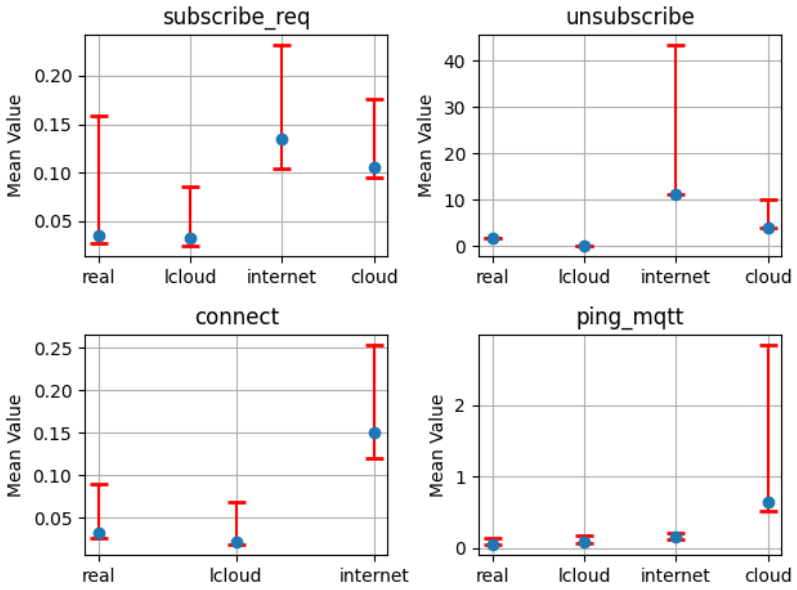

Fig. 5: MQTT Response Time Comparison across Cloud and Fog Deployments.

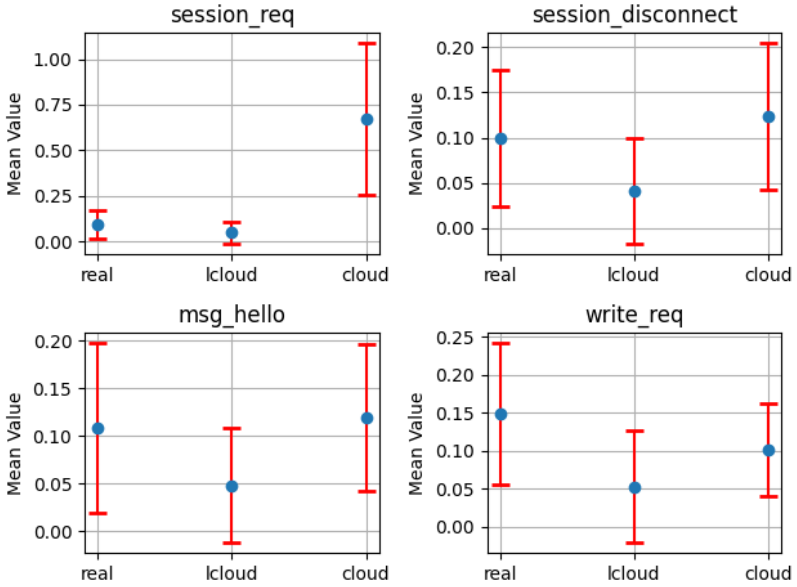

Fig. 6: OPC-UA Response Time Comparison across Cloud and Fog Deployments.

B. *Performance and Realism*

Performance testing revealed that the fog network deployment offers the best balance between realism and response time, with latency close to that of the real IIoT system. In contrast, the cloud deployment demonstrated scalability but with higher latency due to network overhead, highlighting a trade-off between performance and scalability in honeynet deployment.

The results from MQTT and OPC UA performance tests indicate that *TrapNet* can successfully emulate real-time interaction in IIoT systems, capturing attacker behaviors while maintaining realistic response times in the fog network deployment. This makes *TrapNet* an effective tool for simulating IIoT environments that closely mimic real-world conditions.

C. *Attack Scenario Findings*

In the simulated attack scenarios, *TrapNet* successfully attracted adversarial interactions by using decoy IP addresses and exposing realistic services. The attacker's reconnaissance and exploitation attempts were logged in detail, providing valuable insights into attack patterns and techniques. This ability to simulate and log complex attack sequences, such as

TABLE II: Example Logs Captured During Attack Scenarios.

Timestamp	Service	Captured Activity
2024-12-15 14:32:01	MQTT Broker	Unauthorized access attempt on topic “/sensor/data”
2024-12-15 14:33:45	SSH Honeypot	Login attempt with credentials: user=root, pass=123456
2024-12-15 14:34:10	OPC-UA Server	Variable write request for node “ns=2;i=2043”
2024-12-15 14:35:02	HMI Web Interface	Command injection attempt detected in URL query: “/exec?cmd=rm -rf /”

port scans and command injections, sets *TrapNet* apart from traditional security systems, which may not capture such in-depth attack behavior.

VII. Conclusion

This paper introduced the *TrapNet* framework, a honeynet solution designed to enhance IIoT security by simulating realistic environments, enabling dynamic resource scaling, and providing robust attack detection. The evaluation confirmed that *TrapNet* effectively replicates key IIoT components like MQTT brokers, OPC-UA servers, and HMIs, capturing detailed attack data and supporting proactive defense strategies. Performance tests showed a balance between realism and scalability, with the fog network deployment offering optimal response times, while the cloud deployment excelled in scalability.

Overall, *TrapNet* represents a significant advancement in IIoT cybersecurity, enabling realistic attack simulations and improving network security. Future work will focus on integrating machine learning for anomaly detection and expanding support for additional IIoT protocols.

References

[1] W. Alsabbagh and P. Langendörfer, "Security of Programmable Logic Controllers and Related Systems: Today and Tomorrow," IEEE Open Journal of the Industrial Electronics Society, vol. 4, pp. 659-693, 2023, doi: 10.1109/OJIES.2023.3335976.

[2] L. Spitzner, "Honeypots: catching the insider threat," 19th Annual Computer Security Applications Conference, 2003. Proceedings., Las Vegas, NV, USA, 2003, pp. 170-179, doi: 10.1109/CSAC.2003.1254322.

[3] J. Franco, A. Aris, B. Canberk and A. S. Uluagac, "A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems," in IEEE Communications Surveys & Tutorials, vol. 23, no. 4, pp. 2351-2383, Fourthquarter 2021, doi: 10.1109/COMST.2021.3106669.

[4] W. Tian, M. Du, X. Ji, G. Liu, Y. Dai and Z. Han, "Honeypot Detection Strategy Against Advanced Persistent Threats in Industrial Internet of Things: A Prospect Theoretic Game," in IEEE Internet of Things Journal, vol. 8, no. 24, pp. 17372-17381, 15 Dec.15, 2021, doi: 10.1109/JIOT.2021.3080527.

[5] Ph. Church, H. Mueller, C. Ryan, S. V. Gogouvitis, A. Goscinski, H. Haitof and Z. Tari, "SCADA systems in the Cloud," Handbook of Big Data Technologies (2017): 691-718, doi: 10.1007/978-3-319-49340-4.

[6] M. Alam, J. Rufino, J. Ferreira, S. H. Ahmed, N. Shah and Y. Chen, "Orchestration of Microservices for IoT Using Docker and Edge Computing," in IEEE Communications Magazine, vol. 56, no. 9, pp. 118-123, Sept. 2018, doi: 10.1109/MCOM.2018.1701233.

[7] M. Lucchese, F. Lupia, M. Merro, F. Paci, N. Zannone and A. Furfaro, "HoneyICS: A High-interaction Physics-aware Honeynet for Industrial Control Systems," ARES ’23: Proceedings of the 18th International Conference on Availability, Reliability and Security, 113, PP. 1-10, 2023, doi: 10.1145/3600160.3604984.

[8] E. López-Morales, C. Rubio-Medrano, A. Doupé, Y. Shoshitaishvili, R. Wang, T. Bao and G. Ahn, "HoneyPLC: A Next-Generation Honeypot for Industrial Control Systems," in CCS ’20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, PP. 279-291, 2020, doi:10.1145/3372297.342335.

[9] S. Rashid, A. Haq, S. T. Hasan, Md. H. Furhad, M. Ahmed and A. B. Ullah, "Faking smart industry: exploring cyber-threat landscape deploying cloud-based honeypot," Wireless Netw 30, 4527–4541 (2024), doi: 10.1007/s11276-022-03057-y.

[10] S. Ivanova and N. Moradpoor, "Fake PLC in the Cloud, We Thought the Attackers Believed that: How ICS Honeypot Deception Gets Impacted by Cloud Deployments?," 2023 IEEE 19th International Conference on Factory Communication Systems (WFCS), Pavia, Italy, 2023, pp. 1-4, doi: 10.1109/WFCS57264.2023.10144119.

[11] T. Yu, Y. Xin and Ch. Zhang, "HoneyFactory: Container-Based Comprehensive Cyber Deception Honeynet Architecture," Electronics 2024, 13(2), 361; doi: 10.3390/electronics13020361.

[12] M. Alabadi, A. Habbal and X. Wei, "Industrial Internet of Things: Requirements, Architecture, Challenges, and Future Research Directions," in IEEE Access, vol. 10, pp. 66374-66400, 2022, doi: 10.1109/ACCESS.2022.3185049.

[13] W. Alsabbagh, C. Kim and P. Langendörfer, "Silent Sabotage: A Stealthy Control Logic Injection in IIoT Systems," 2024 Silicon Valley Cybersecurity Conference (SVCC), Seoul, Korea, Republic of, 2024, pp. 1-8, doi: 10.1109/SVCC61185.2024.10637363.

[14] W. Alsabbagh, S. Amogbonjaye, C. Kim and P. Langendörfer, "Pirates of the MQTT: Raiding IIoT Systems with a Rogue Client," presented at the 8th Cyber Security in Networking Conference (CSNet), 2024, Paris, France, doi: 10.13140/RG.2.2.11963.84004.

[15] W. Alsabbagh, C. Kim and P. Langendörfer, "A Payload of Lies: False Data Injection Attacks on MQTT-based IIoT Systems," presented at 50th Annual Conference of the IEEE Industrial Electronics Society, Chicago, USA, 2024, doi: 10.13140/RG.2.2.14002.58565.

[16] W. Alsabbagh, C. Kim, N. S. Patil and P. Langendörfer, "Beyond the Lens: False Data Injection Attacks on IIoT-Cameras through MQTT Manipulation," 2024 7th Conference on Cloud and Internet of Things (CIoT), Montreal, QC, Canada, 2024, pp. 1-7, doi: 10.1109/CIoT63799.2024.10757025.

[17] W. Alsabbagh, C. Kim, N. S. Patil and P. Langendörfer, "Hacking the Backbone: Shell Reverse Attacks on IIoT Systems," presented at the 21st International Conference on Embedded Wireless Systems and Networks (EWSN), Abu Dhabi, Emirates, 2024, doi: 10.13140/RG.2.2.11963.84004.

[18] A. L. de Sousa and A. S. de Oliveira, "Order-Controlled Production Employing Multi-Agent and Flexible Job-Shop Scheduling on a Physical Simulation Platform," 2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE), São Bernardo do Campo, Brazil, 2022, pp. 229-234, doi: 10.1109/LARS/SBR/WRE56824.2022.9995868.

[19] D. Urrego, "TrapNet Framework," GitHub. [Online]. Available: https://github.com/r00tPl4nt3r/icscloudhnet. [Accessed: Jan. 8, 2025].