ISIA-AF: Orchestrating Reproducible Attacks and Multi-Source Data Collection for OT Systems

Stefan Manfred Haratzmüller[1], Thomas Rosenstatter[1][0000−0002−9587−3423], Olaf Saßnick[1,2][0000−0002−3090−4399], Dalibor Sain[1][0009−0001−4837−9537], and Stefan Huber[1][0000−0002−8871−5814]

[1] Josef Ressel Centre for Intelligent and Secure Industrial Automation, Salzburg University of Applied Sciences, 5412 Puch/Salzburg, Austria
[firstname.lastname]@fh-salzburg.ac.at
[2] Paris Lodron University Salzburg, 5020 Salzburg, Austria

Abstract. Operational Technology (OT) environments require realistic, reproducible security datasets, yet existing approaches often lack automation, multi-source data capture, and sufficient documentation for reuse. This paper presents ISIA-AF, a modular attack framework for orchestrating reproducible attack execution and automated dataset generation on industrial systems. The framework coordinates distributed attack clients, records network traffic and operational data, ultimately leading to a multi-source dataset. We derive functional and non-functional requirements from prior work and stakeholder discussions, and realise the framework following a design science research approach. A case study on the ISIA testbed, comprising a real industrial system and a simulated process, demonstrates how the framework supports centralised control, low communication overhead, and flexible deployment across network segments. The result is a practical basis for generating extensible, multi-source OT security datasets for intrusion detection research.

Keywords: Operational Technology · Cybersecurity · Dataset Generation.

1 Introduction

Operational Technology (OT) requires increased security as threats to these systems have increased tremendously in recent years. This is partly due to the growing interconnectivity and complexity of these systems, which leads to larger attack surfaces. For instance, in 2017 Triton surfaced, the first attempt against safety systems targeting Triconex SIS controllers [9].

To improve security, datasets are essential for training and evaluating security mechanisms. For this, anomaly and Intrusion Detection Systems (IDSs) demand realistic datasets to improve detection rates. These datasets, in turn, need to consist of a wide range of attack scenarios and a diversity of technologies. Despite this, security datasets for OT systems, especially those including newer,

This is the authors' pre-publication version of a paper accepted for Euromicro SEAA 2026. The final authenticated version will appear in the conference proceedings of Euromicro SEAA 2026, published by Springer in the Lecture Notes in Computer Science (LNCS) series.

state-of-the-art protocols like Open Platform Communications Unified Architecture (OPC UA), are scarce. At the time of writing only a very limited number of datasets for security research explicitly use OPC UA as the primary protocol, and prior work [11,12] is based on networks of Raspberry Pi devices running a Python OPC UA implementation. Of these two, only Pinto [11] made the dataset publicly available. X-IIoTID [1], one of the most extensive OT datasets, includes network traffic, host logs, system resources, and IDS alerts, with support for protocols such as Modbus, MQTT, and CoAP. IoTForge Pro [8] introduces a security testbed for IIoT environments that also generates the ForgeIIOT dataset. While IoTForge Pro demonstrates a structured approach to dataset generation, the attack scripts and automation details are not publicly available, leaving the methodology difficult to reproduce or extend.

Conti et al. [3], who reviewed 83 industrial control system testbeds and datasets for security research, identify the most important dataset properties as *reproducibility*, *realism*, and *extensibility*. Goldschmidt and Chudák [4] provide a more recent review on 89 network intrusion datasets, and criticise that security datasets are typically published without the complete documentation for full reproducibility. For instance, X-IIoTID [1] is publicly available, but details of the automation used for traffic generation and attack scenarios are not released. As such, extending the dataset with more recent attacks or building new datasets is more challenging, and the successful paths and tooling required to create high-quality datasets remain unclear.

While general attack simulation and automation frameworks [10,13] are available, they are typically not designed for reproducible dataset generation. MITRE Caldera [10], for example, focuses on mimicking real attackers by adding stochasticity and employing a model based on abilities and facts, which makes it difficult to execute reproducible attack chains. General-purpose automation frameworks like Ansible [14] exist, but they are primarily designed for sequential task execution on remote hosts. Although such frameworks can orchestrate distributed workflows, the SSH-based execution of Ansible adds overhead and may introduce timing artefacts. Ultimately, LCM [6], a lightweight communication framework, was selected for its flexibility while also minimising overhead.

Contributions. In this work, we present *ISIA-AF*, an attack framework that enables the execution of a wide range of attacks on industrial systems. The framework is used not only to coordinate attack scenarios, but also to automate the generation of a security dataset containing data from both network traffic and operational data from the industrial process, thereby *enriching the dataset*. Overall, *ISIA-AF* (i) enables reproducible attack runs, (ii) automates dataset generation, (iii) enriches the dataset with multiple data sources, (iv) provides a modular and extensible architecture for the development of new attacks, and (v) is publicly available[3]. We demonstrate its functionality by deploying it on an industrial testbed that is representative of a real-world industrial environment.

[3] `https://github.com/JRC-ISIA/isia-attack-framework`

2 Attack Framework Design

The design of *ISIA-AF* follows Wieringa's Design Science Research (DSR) methodology [16], which structures artefact construction and evaluation around a *design cycle* of three iterative activities: (i) *problem investigation*, identifying requirements from the security and dataset needs of OT environments; (ii) *treatment design*, specifying the architecture and features of *ISIA-AF*; and (iii) *treatment validation*, demonstrating the framework's utility through a case study on the Intelligent and Secure Industrial Automation (ISIA) testbed.

The central design problem is the lack of systematic and automated means of executing, coordinating, and documenting attack scenarios on OT systems to generate reproducible, enriched security datasets, a gap highlighted in the literature [3,4]. *ISIA-AF* addresses this through a modular and server/client architecture that captures multi-type data.

Objectives. Table 1 provides a list of the functional and non-functional requirements derived from literature [3,4] and discussions with the stakeholders at the ISIA research centre.

To achieve these objectives, such as modular design, centralised control, and minimal overhead, the framework consists of the following component types: *attack orchestrator*, *attack clients*, *capture modules*, *logger*, *feature extractor*, and

Table 1. Functional and non-functional requirements for *ISIA-AF* derived from literature and discussions with stakeholders.

R.#	Requirement Description
Functional Requirements	
FR.1	Load attack definition using a structured data format (e.g., JSON or YAML).
FR.2	Execution without an attack must be possible ("no-attack" scenario).
FR.3	Execution of the scenario can be triggered manually or by a schedule.
FR.4	Record traffic on various OSI layers.
FR.5	Log metadata such as timestamps, attack names, and success/failure flags.
FR.6	Extract relevant features from recorded traffic for dataset creation.
FR.7	Label captured data as *under attack* or *normal/benign*.
FR.8	Provide reliable persistent data storage for later use and archival purposes.
Non-Functional Requirements	
NF.1	The framework must be modular to allow adaptability and easy extension with new features or components.
NF.2	Coordination and control must remain centralised.
NF.3	Attack clients must allow new attack types that can be implemented and integrated independently.
NF.4	Support communication and coordination with minimal communication overhead.
NF.5	Provide temporary (buffer) storage during processing.

storage transfer module. The orchestrator is responsible for the centralised control and monitoring of the framework. The attack clients perform the timed execution of attacks, and the capture clients record network traffic and system response. The logger is responsible for logging attack activities and system responses, and the feature extractor is responsible for data enrichment. The storage transfer is used to archive the logs centrally.

Placement within the Testbed. The attack framework is to be placed in the ISIA testbed, which is a HIL realisation that simulates the physical process of three Injection Moulding Machines (IMMs) and robot arms that lift the produced items on a conveyor belt. The control of the IMMs is performed on Programmable Logic Controllers (PLCs) by B&R Automation and can be manipulated and monitored with a Supervisory Control and Data Acquisition (SCADA) server by COPA-DATA.

Figure 1 details the network view of the ISIA testbed split into three segments following IEC 62443 [7] recommendations. The leftmost segment is the *shop floor*, which contains the three IMMs and the PLCs (PLC 1-4) for controlling the production process. The simulation of the robot arms is performed by automation PCs (KUKA 1-4), each for one arm respectively. The mid-section is the *Industrial Demilitarized Zone (DMZ)* comprising the SCADA system and a security server that provides additional security services. The *enterprise zone* can be found in the rightmost segment and consists of an ERP system and an information engine tasked with monitoring the operational state of the OT system. Typically, this zone is also connected to the internet.

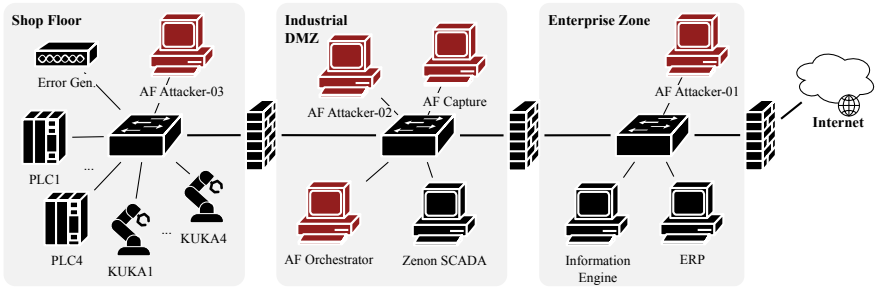

Fig. 1. Network view of the ISIA testbed with the attack framework. Attacker clients are located in each level of the industrial network, and the orchestrator and capture components are located in the industrial DMZ.

As the framework must be modular so that various attacks can be performed (*NF.3*), it is necessary to place attack clients in each network segment of the testbed. This distributed placement ensures that intra-segment attacks (e.g., lateral movement, ARP spoofing, or Denial of Service (DoS) within a single zone) as well as cross-segment attacks (e.g., man-in-the-middle traversing zone boundaries) can be reproduced without constraints.

The placement of the attack orchestrator and the capture module can be adapted, but we suggest placing these components in the industrial DMZ. The industrial DMZ is chosen due to its location, representing a central point for all network traffic between the Information Technology (IT) and OT zones. In addition, the capture module is connected to mirrored switchports, which allows it to capture all network traffic.

2.1 Architecture

All components communicate via LCM [6], a lightweight publish/subscribe middleware that satisfies the minimal-overhead communication requirement (*NF.4*) through UDP multicasts. The components subscribe to their respective channels and may further filter messages by their unique identifier (e.g., `attacker-01`). Fig. 2 provides an overview of the components of ISIA-AF and their interaction via LCM channels.

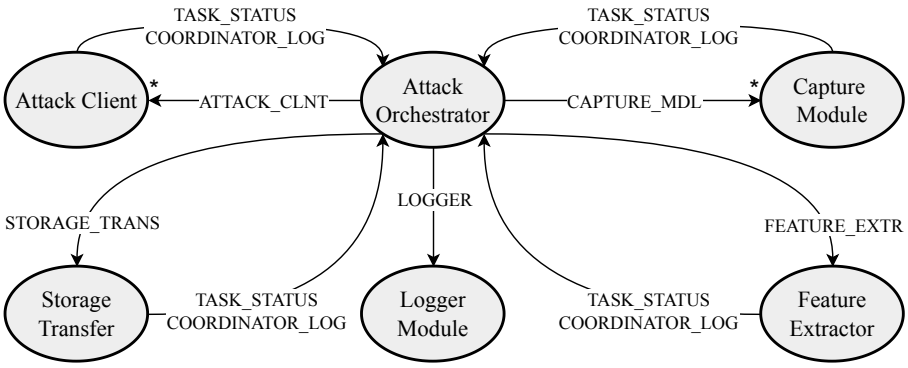

Fig. 2. Overview of the ISIA-AF components and their interaction via LCM channels. The *Attack Orchestrator* is the central component, which coordinates the attack execution and data collection. The arrows further indicate the direction of the LCM messages.

Attack Orchestrator: The orchestrator has sole control of the components and issues timed commands to all registered clients, and aggregates metadata (*FR.5*) in logs received through the logger module. It provides a command line interface, but also allows loading attack scenario definitions encoded in *JSON* (*FR.1*) to either trigger them manually or schedule them for automated execution (*NF.2, FR.3*). Internally, it parses the scenario definition into an ordered list specifying the target attack client, its parameters, and an ISO 8601 timestamp or offset, and broadcasts them to the relevant clients over an LCM channel. Upon completion of each directive, clients publish a status message that the orchestrator uses to update its execution log, recording success or failure flags alongside timestamps

and further responses (*FR.5*). A "no-attack" scenario, in which the framework runs but no attack is issued, is also supported as an execution mode (*FR.2*), thereby enabling the generation and recording of a benign baseline.

Attack Client: On receipt of a directive, the attack client dynamically loads and invokes the specified *attack script* (*NF.3*) enabling new attack types to be integrated without modifying the client itself (*NF.3*). The number of attack clients is variable and allows placement in any network segment. For instance, Figure 1 illustrates three clients (`AF Attacker-01` through `AF Attacker-03`) deployed in the enterprise zone, industrial DMZ, and shop floor, respectively.

Capture Module: The capture module is a dedicated node connected to mirrored switchports in the industrial DMZ, granting it passive visibility into all inter-zone traffic without injecting additional frames into the network (*NF.4*). It records traffic based on variable filters at the network interface level, and stores the raw `.pcap` files on a network share (*NF.5*) for later processing by the attack orchestrator.

Logger Module: The labelling of all data records, i.e., operational data (OPC UA data) and network traffic (*FR.4*), is driven by the execution log produced by the orchestrator (*FR.7*). Execution logs contain the start and end times of attacks and normal operations, as well as outputs and errors occurring during the execution of the attack scripts.

Feature Extractor: Temporal alignment is achieved through time synchronisation across all testbed nodes. This allows the feature extractor to query the operational data database using the start and end times of the network traffic recordings. The feature extractor then computes and stores the traffic features as flows similarly to [1], and saves the operational data according to the OPC UA data model. The source of the operational data is called the *information engine* and is further described in [5].

For labelling data, the feature extractor uses the execution log to annotate both network traffic and operational data (*FR.6*). The current version of *ISIA-AF* applies the label *"under attack"* to all data during attack periods. Future work will extend labelling via logger functions to include more granular categories such as *attack traffic*, *system reaction/recovery*, and *normal traffic*, as recommended by Conti et al. [3].

Storage Transfer: Once the data is labelled, it is transferred to a persistent storage unit (*FR.8*) in the form of separate JSON files: labelled network flows, labelled operational data, the execution log, and the network capture as `.pcap` file. Keeping the logs separated was a design decision to allow for easier analysis and processing of data in case only one type is of interest for evaluating an IDS. Lastly, the locally stored files are removed.

2.2 Attack Scenarios

Attack scenarios are defined through tasks which include the attacks themselves, but also allow the environment to be prepared through tasks such as copying new scripts to attacker devices and starting or stopping network capture. Listing 1.1 shows the properties of each action, including a unique identifier, the action type, the absolute or relative execution time, and a blocking flag that either allows the attack coordinator to directly continue with the next action (`blocking=false`) or waits for the current action to finish before continuing with the next one (`blocking=true`). Finally, additional action-specific parameters can be passed.

Listing 1.1. Structure of attack definitions allowing to use absolute and relative time, enabling blocking of actions and defining the targets.

```
tasks ::= [ Action* ]
Action ::= {
  id: String,
  type: { execute_local, start_capture, stop_capture,
      set_alias, start_attack_script, start_attack,
      stop_attack, extract_features, store_dataset },
    targets: [ String* ],
    time: (RelativeTime | AbsoluteTime) | null,
  blocking: Boolean,
  params: { (String : Value)* }
}
```

3 Case Study

In the following case study, we define actions for three phases: (i) setup, in which the necessary artifacts are distributed; (ii) recording, during which the attacks are performed; and (iii) post-attack, during which data recordings are copied to the feature extractor for subsequent dataset generation.

The attack scenario definition (*opcua-chunk-flood*) along with a detailed task description can be found in [15]. This example is chosen to highlight the flexibility of *ISIA-AF*, not to demonstrate a new attack vector for OT systems. We consider an attacker who has already managed to get access to the poorly secured monitoring system. From there, the attacker searches the network for OPC UA servers, reads their information model to identify the PLCs used in manufacturing, and then performs a DoS attack.

Phase I (setup): The system is prepared, i.e., files are copied to the attack agents using *execute_ local* and the traffic capture (*start_ capture*) is started with a network filter ignoring the UDP multicast communication and remote access to the devices used to oversee their status. Furthermore, the logging of the PLCs is enabled by running a script with the *execute_ local* command.

Phase II (recording): For defining this attack scenario, the attacker device located in the DMZ first executes a `nmap` scan for TCP/IP port 4840 to identify OPC UA servers. Afterwards, the attacker explores the OPC UA servers to

identify the PLCs used in the plastic moulding process using a Python OPC UA client. Finally, the attacker performs a chunk flood attack on all identified machines utilising the *opcua-exploit-framework* [2].

Phase III (post-attack): The capture is stopped (*stop_ capture*) and the feature extraction started. The logs and pcap are consequently moved to a central device (coordinator) from where a database query is sent to retrieve the operational data during the time of the recording. More logs, like the proprietary logs recorded by the PLCs, are also copied through *execute_ local* to further enrich the dataset. Once all data is collected and labelled, the action *store_ dataset* transfers the data to a network storage for archiving.

The given scenario highlights the versatility of the framework. Not only are attacks coordinated, but traffic capture is also orchestrated, network flows are automatically annotated with *under attack* and *benign*, and specific scripts can be executed to enhance the dataset with additional information.

4 Conclusion

This paper presents *ISIA-AF*, a modular attack framework for orchestrating reproducible attack execution and automated dataset generation on OT systems. Following Wieringa's DSR methodology, we derived functional and non-functional requirements from prior work and stakeholder discussions, and realised a framework comprising a centralised orchestrator, distributed attack clients, a passive capture module, a logger, and a feature extractor. The framework addresses a key gap identified in the literature [3,4] for OT systems using OPC UA: the lack of systematic tooling for generating reproducible, multi-source OT security datasets with sufficient documentation for reproducibility.

A case study on the ISIA testbed demonstrated that *ISIA-AF* supports flexible, multi-segment attack deployment with minimal communication overhead, and produces enriched datasets combining network traffic and OPC UA operational data. To the best of our knowledge, *ISIA-AF* is among the first publicly documented attack frameworks evaluated on a real-world OT environment that uses OPC UA as the main communication protocol.

Future work will focus on extending the labelling scheme to include more granular categories such as attack traffic, system reaction, and normal traffic, as well as expanding the attack library with additional OT-specific attack scenarios.

Acknowledgments. The financial support by the Austrian Federal Ministry of Economy, Energy and Tourism, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

References

1. Al-Hawawreh, M., Sitnikova, E., Aboutorab, N.: X-IIoTID: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things. IEEE Internet of Things Journal **9**(5), 3962–3977 (2022). doi:10.1109/JIOT.2021.3102056

2. Claroty Research - Team82: OPC-UA exploitation framework (2023), https://github.com/claroty/opcua-exploit-framework
3. Conti, M., Donadel, D., Turrin, F.: A survey on industrial control system testbeds and datasets for security research. IEEE Communications Surveys & Tutorials **23**(4), 2248–2294 (2021). doi:10.1109/COMST.2021.3094360
4. Goldschmidt, P., Chudá, D.: Network intrusion datasets: A survey, limitations, and recommendations. Computers & Security **156** (2025). doi:10.1016/j.cose.2025.104510
5. Hirsch, E., Hoher, S., Huber, S.: An OPC UA-based industrial Big Data architecture. In: 2023 IEEE 21st (INDIN'23). pp. 1–7. IEEE, Lemgo, Germany (07 2023). doi:10.1109/INDIN51400.2023.10217899
6. Huang, A.S., Olson, E., Moore, D.C.: LCM: Lightweight communications and marshalling. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2010). doi:10.1109/IROS.2010.5649358
7. IEC: Industrial Communication Networks: Pt. 3.2: Security for Industrial Automation and Control Systems: Security Risk Assessment for System Design (2020)
8. Kumar, P., Mullick, S., Das, R., Nandi, A., Banerjee, I.: IoTForge Pro: A security testbed for generating intrusion dataset for industrial IoT. IEEE Internet of Things Journal **12**(7), 8453–8460 (Apr 2025). doi:10.1109/JIOT.2024.3501017
9. Makrakis, G.M., Kolias, C., Kambourakis, G., Rieger, C., Benjamin, J.: Industrial and critical infrastructure security: Technical analysis of real-life security incidents. IEEE Access **9**, 165295–165325 (2021). doi:10.1109/ACCESS.2021.3133348
10. MITRE Corporation: MITRE Caldera. https://github.com/mitre/caldera (2024), version 5.0.0, released 2024-02-14
11. Pinto, R.: Dataset: M2M using OPC UA (2020). doi:10.21227/ychv-6c68
12. Radhakrishnan, V., Kabilan, N., Ravi, V., Sowmya, V.: Unsupervised Representation Learning Approach for Intrusion Detection in the Industrial Internet of Things Network Environment, pp. 41–76. Springer (2025). doi:10.1007/978-3-031-72636-1_3
13. Red Canary: Atomic Red Team. https://www.atomicredteam.io/ (2026), open-source library of adversary emulation tests mapped to MITRE ATT&CK
14. Red Hat, Inc.: Ansible: IT automation software. https://www.ansible.com (2026)
15. Saßnick, O., Rosenstatter, T.: Example of using the ISIA-AF framework (Jun 2026). doi:10.5281/zenodo.20488140
16. Wieringa, R.: Design science methodology for information systems and software engineering. Springer (2014). doi:10.1007/978-3-662-43839-8