

AI-based Attacker Models for Enhancing Multi-Stage Cyberattack Simulations in Smart Grids Using Co-Simulation Environments

Ömer Sen^{*†}, Christoph Pohl^{*}, Immanuel Hacker^{*†}, Markus Stroot^{*†}, Andreas Ulbig^{*†},
^{*}RWTH Aachen University, Aachen, Germany — [†]Fraunhofer FIT, Aachen, Germany
 Email: {oemer.sen, immanuel.hacker, markus.stroot, andreas.ulbig}@fit.fraunhofer.de
 christoph.pohl@rwth-aachen.de

Abstract—The transition to smart grids has increased the vulnerability of electrical power systems to advanced cyber threats. To safeguard these systems, comprehensive security measures—including preventive, detective, and reactive strategies—are necessary. As part of the critical infrastructure, securing these systems is a major research focus, particularly against cyberattacks. Many methods are developed to detect anomalies and intrusions and assess the damage potential of attacks. However, these methods require large amounts of data, which are often limited or private due to security concerns. We propose a co-simulation framework that employs an autonomous agent to execute modular cyberattacks within a configurable environment, enabling reproducible and adaptable data generation. The impact of virtual attacks is compared to those in a physical lab targeting real smart grids. We also investigate the use of large language models for automating attack generation, though current models on consumer hardware are unreliable. Our approach offers a flexible, versatile source for data generation, aiding in faster prototyping and reducing development resources and time.

Index Terms—Smart Grid, Cybersecurity, Cyberattack, AI-based Attack, Large Language Models

I. INTRODUCTION

As power grids increasingly incorporate Information and Communication Technologies (ICT), they become more interconnected with external devices, removing the natural barrier of isolation and exposing them to new cybersecurity threats. A notable example is the 2015 Ukraine cyberattack, which resulted in a significant blackout. Cybersecurity in power systems is a vital, ongoing process based on prevention, detection, and reaction, guided by standards such as IEC 62351 and IEC 62443. Additionally, improving cybersecurity resilience and protecting critical infrastructures are emphasized legally with emerging regulations like NIS2.

Smart grids are complex systems and combine a lot of software and components. They include their accumulated vulnerabilities and even the used protocols like IEC 104, commonly used in SCADA systems, lack authentication and encryption. Implementing security enhancements means evaluating new metrics, analyzing component behavior in a network, examining logs, and running Intrusion Detection Systems (IDSs) in smart grids. These steps are promising but all depend on data or reference values for analysis. While some openly accessible datasets exist, many are either poorly maintained, too specific, or have other shortcomings [1].

Grid operators tend to keep their data private due to security concerns, forcing researchers to either build their own laboratories or simulate smart grids themselves. Modelling smart grids in a laboratory requires expertise, time to install hardware, and significant expense to operate. Implementing and setting up even a small laboratory demands professionals from various domains.

Other studies evaluated multiple simulation methods with and without cyberattacks [2]. Some tools could generate reports and run simple Distributed Denial of Service (DDoS) attacks, while others focused solely on co-simulating power grids and their network communication. Manual red teaming approaches face challenges in consistency and scalability, making automation a promising solution. Frameworks like Plan2Defend [3] and autonomous agents for cyber defense [4] use autonomous methods to mitigate ongoing cyberattacks or execute plans, reducing response time. While these methods evaluate system security and monitor activity, they lack the ability to generate attack data, relying on manual implementation of sophisticated attacks for training.

Proactive testing via attack emulation allows for hardening security before actual attacks occur. Although individual cyberattacks on smart grids have been analyzed, there is less focus on general frameworks for complex data generation. IDSs require extensive data to evaluate performance and adapt to varying network topologies.

To address these challenges, we propose a method for automatically running cyberattacks inside a simulation environment, generating network logs and power simulation results. This approach allows for the creation of datasets in configurable network topologies and various attack scenarios, improving data availability and eliminating the need for expensive setups. This comprehensive approach enables the extraction of valuable data, facilitating the study and enhancement of smart grid cybersecurity. Our contributions are the following:

- Definition of the problem and corresponding requirements for our method, and deduction of a general concept for our work.
- Design and assessment of the AI-based attacker model in a simulation environment, detailing the physical laboratory and co-simulation structure, the autonomous agent's

operation, deployment environments, and the data generation process.

- Evaluation of the data generation process via comparative analysis with a physical laboratory using a digital twin approach.

II. BACKGROUND

A. Smart Grid Control Architecture

Compared to early power grids that directly connected energy generators with consumers, modern energy infrastructure is much more complex. The increasing use of energy sources such as Photovoltaic (PV) systems and wind turbines makes regulation and coordination more challenging. When these generators are distributed and connected to a large power grid without bundling their loads in a large converter, special regulation and balancing are required before their energy can be fed into the grid. Smart grids provide a solution as cyber-physical systems that connect energy sources with consumers and enable communication between actors to balance and control the power grid reliably.

Understanding smart grid operation requires knowledge of its components:

Terminal Units enable smart grid communication by collecting and transmitting data and sending and receiving control commands. These units are divided into Remote Terminal Units (RTUs), connecting to physical equipment, and Master Terminal Units (MTUs), aggregating data from RTUs. **Supervisory Control and Data Acquisition (SCADA) Systems** handle data from terminal units, process it to adjust grid behavior, and provide Human-Machine-Interfaces (HMIs) for data visualization and remote control of grid components. **Intelligent Electronic Devices (IEDs)** are controllers in power equipment that collect data from sensors and issue commands. They communicate with SCADA systems directly or via RTUs. **Decentralized Energy Resources (DERs)** are small components supplying or demanding electricity, connecting directly to the grid or aggregated. Examples include micro-turbines, solar arrays, and energy storage systems. **Protocols** are crucial for data exchange in smart grids. Common ones like IEC 104 and Modbus operate over TCP/IP networks to connect SCADA systems with terminal units but lack integrated security, making them vulnerable to cyberattacks. Secure Shell (SSH) is used for remote access but can pose security threats if compromised.

The overall structure of an Industrial Control System (ICS) can be described using the Purdue model, proposed by Williams et al. in the 1990s [5]. This model segments the network into multiple layers, allowing separation between operational technology (OT) and information technology (IT) systems. The model defines a hierarchy for communication between layers, ranging from external services at the highest level (Level 5) to the actual control of physical actors at the lowest level (Level 0).

The layers of the Purdue model are as follows:

Level 5 (Enterprise Zone) includes general IT systems accessible to external clients, providing the most open services.

Level 4 focuses on data and software for production scheduling and operational management, including mail servers and other widely accessible services. The **Demilitarized Zone (DMZ)** separates IT and OT systems, using firewalls to block most communication protocols and secure sensitive data by preventing outgoing communication. **Level 3 (Manufacturing Zone)** is the highest level within the OT site, containing local workstations and data collections for process data. **Level 2** houses area control systems that collect and monitor data within subsystems. **Level 1** consists of controlling elements that send measured values to data collectors and control physical actors and sensors. **Level 0** includes all machinery performing physical tasks and measurements.

Despite being proposed decades ago, the Purdue model remains widely applied, particularly in the context of cybersecurity for cyber-physical systems.

B. Simulation Aspect

To simulate smart grids, we must understand what simulation entails and how it can be achieved. Generally, simulation involves imitating real-world systems via software. Instead of creating a single, complex model, implementing smaller components separately and combining them to model complex systems increases the framework's sustainability and performance feasibility. These components can run in individual virtual environments to prevent interference.

Virtualization involves running individual processes or entire Operating Systems (OSs) on multiple virtual computers, each using a portion of the host computer's resources. In contrast to the well-established approach of bundling and isolating software in Virtual Machines (VMs), we implemented our work with the help of containerization. While VMs include their own operating systems, containerization allows software to share resources with the host dynamically, increasing performance for parallel execution. These instances are called containers and are built according to a blueprint, that defines the corresponding software, memory and network available when instantiating it. Containers can communicate through virtual network, sending actual network packets between specific instances.

Virtual networks allow communication between specific containers or VMs, creating separated networks on the same host. Orchestration tools further streamline the setup by reading configuration files to define containers and their settings, automating their deployment and management.

Using container orchestration, we can automate the deployment of pre-configured containers and their virtual networks. This enables running complex scenarios with shared computing resources, making large-scale simulations less resource-demanding.

A key goal of our work is to run simulations automatically and incorporate systems classified as artificial intelligence (AI). **Automation** in software refers to the automatic operation of software with minimal user interaction, where a program executes pre-defined tasks. **Autonomy** describes a program's

TABLE I: Example for a multi-staged cyberattack.

Step	Result	Depends on
1. Get hostname	Hostname	
2. Check interfaces	Available networks	(1)
3. Scan network	Reachable targets	(2)
4. Generate SSH keys	SSH keys	
5. Brute-Force credentials	Compromised targets	(3),(4)
6. Connect agent to C2 server		(5)
7. Remove SSH key		

ability to make decisions and react to its environment. According to [6], autonomy is a spectrum, with greater consideration of parameters leading to higher autonomy.

The European Commission’s Artificial Intelligence Act [7] defines AI systems as machine-based systems designed to operate with varying levels of autonomy. These systems may exhibit adaptiveness and generate outputs such as predictions, content, recommendations, or decisions that influence environments.

C. Multi-Staged Cyberattacks

As smart grids increasingly resemble traditional computer networks, they become susceptible to similar cyberattacks [8]. Below are some prevalent attack schemes and their potential combinations for more complex routines.

Attackers often combine techniques from different categories to execute complex attacks, making them harder to detect and defend against.

Advanced Persistent Threats (APTs) involve multiple stages executed sequentially. These stages typically include reconnaissance, initial access, persistence, and command-and-control, followed by the final impact. The complexity of APTs, with their numerous specialized steps, targets specific victims over a prolonged period, making them difficult to track and defend against.

For instance, the DARPA 2000 dataset features various DDoS attacks comprising multiple steps [9]. A simpler attack might include IP address scanning, service exploitation, and a subsequent DDoS attack using compromised hosts. A more sophisticated attack, such as island-hopping, spreads malware through infected PDFs sent across the network until a desired database service is found. These examples highlight how combining different attack methods and exploiting vulnerabilities can allow attackers to gather extensive information and manipulate network components.

When trying to breach complex systems, such as critical infrastructure, attacks need a methodical progression. So called *multi-staged cyberattacks* unfold in several distinct steps, where each has an individual objective. The key property of running complex cyberattacks in this manner is the dependency of consecutive steps on the results of their predecessors.

Such dependencies allow individual stages of the attack to rely on information that were previously collected and thus make modelling complex attacks possible. This comes at the cost of having to plan the steps according to the available information and which dependencies are fulfilled. In Table I we picture a multi-staged cyberattack, where for example the

brute-force depends on the results of a network scan. Running a brute-force attack without having a target diminishes the value of the operation; therefore, we can only perform this step if a potential target has been found. Structuring attacks in such a modular way qualifies multi-staged cyberattacks to adapt the environment if planned accordingly.

III. METHODOLOGY

A. Requirements

We aim to create a framework for generating network logs of smart grids during multi-stage cyberattacks. In order to achieving this, there are a few requirements that we postulate from sparsely available datasets.

Virtual Smart Grid Simulation: A simulation resembling real grids, incorporating essential components of real-world smart grids, and entirely virtual to eliminate hardware requirements is necessary. **Recordable Network Communication:** The simulation must log network traffic at the transport layer. **Integrated Attack Simulation:** Capturing anomalies during adversary actions within the simulation is crucial for utilizing the network and component logic effectively. **Realistic Impact on Data Output:** Generated data should be as usable as real hardware data, accurately reflecting cyberattack impacts. **Configuration-Based Setup:** The simulation should be configurable for various scenarios and topologies, allowing automatic setup based on user-defined parameters to ensure scalability and prevent initialization errors. **Autonomous Attacks with Modular Methods:** Supporting multiple attack methods, the simulation should manage multi-stage attacks autonomously, ensuring reproducibility and optimal method selection. **Offline Availability:** The simulation should function without internet access, adhering to the segmented network architecture typical of smart grids. **Clean and Consistent Network Logs:** Generated network traffic should be uniform, minimizing fluctuations across different runs to ensure attack impacts are clear.

For our data, two main properties are crucial: realism and consistency across different scenarios. Realistic data must closely depict the individual nuances of each attack, and consistency ensures that data output is structured similarly across different runs. We identified four main metrics to analyze each attack scenario:

Power Simulation Results: The simulation must accurately reflect the effects on power generation and distribution, showing changes in component behavior due to cyberattacks. **Timely Spread:** Attacks should occur within a consistent timeframe, with tasks performed at similar points during the simulation. **Traffic Volume:** Reliable simulation behavior is indicated by consistent data communication volume and direction during attacks. **Protocol Distribution:** Consistent protocol usage during an attack ensures stable behavior of the involved components.

These metrics help ensure that the generated data is both realistic and consistent, providing a robust foundation for analyzing and improving smart grid cybersecurity. Our methodol-

ogy focuses on these requirements, and we will evaluate how these were satisfied by our framework at the end of this work.

B. Design Idea

To meet the requirements for generating realistic network logs of smart grids during multi-stage cyberattacks, we propose a design that ensures realistic simulation, modularity, and scalability without compromise.

Each component in the simulation must mimic its physical counterpart's behavior. This includes modeling interactions with other elements in the simulation environment, especially data exchanges and remote control messages, to produce realistic data.

A modular structure allows flexible configuration of scenarios using reusable components. Containerization tools, such as Docker, support this modularity. Using Docker images for the smart grid actors enables easy instantiation of multiple components with the same logic but different parameters. Networking configurations ensure that only selected connections are enabled between containers, specifying maximum bandwidth and simulated latency to create realistic network logs, even on a single host. Users can define the topology in readable configuration files, adapting it as needed.

Orchestration tools help manage complex network topologies and prevent user errors. These tools can derive a whole simulation scenario from a single file, handling initialization and setup. Established orchestration tools also support scripts triggered after containers start or stop, facilitating the setup and extraction of network logs.

With these design choices, we chose a Docker-based approach leveraging containerization capabilities. Using a collection of pre-existing components for network communication and power simulation, we adopted the co-simulation environment proposed in [10].

This approach employs several tools and frameworks that align with our design decisions:

- **rettij**: Utilizes Kubernetes orchestration, allowing the use of configuration files to define components and network connections in virtual smart grids via Docker images.
- **Mosaik and Panda Power**: Add power simulation capabilities, enabling realistic power grid simulations [11].

Based on this co-simulation environment, we developed an attacker container that autonomously performs specified cyberattacks. This attacker integrates seamlessly into the co-simulation, generating network logs and power simulation results applicable to real-world scenarios.

By combining these components, our framework facilitates the generation of realistic, consistent, and high-quality data for analyzing and improving smart grid cybersecurity.

IV. APPROACH

A. Co-Simulation Environment

Our work utilizes the co-simulation software Mosaik, which groups individual simulators in a common context to build scenarios modularly [12]. Each simulator provides the functionality of real-world components, enabling the reconstruction

of existing physical scenarios and the reuse of logic across varying simulator combinations.

To simulate the power grid and communication within a laboratory setup, we use Panda Power and other network communication simulators within Mosaik. Panda Power discretely simulates loads at each time step, while network communication simulators manage the interactions between components. Mosaik synchronizes these simulators, iterating through each simulation step and ensuring timely execution based on the current simulation time.

Mosaik connects modular simulators, synchronizing their communication to create a discrete environment. It iterates through each simulation step, checking if enough time has passed to execute the next step and then instructs the corresponding simulator to proceed.

Panda Power generates measurements for the virtual power grid, simulating components based on a predefined configuration file. This file describes power generators, transformers, buses, and power lines. The framework extracts detailed information about current usages and loads across all components [11].

Terminal unit nodes handle IEC 104 communication, running as either RTUs or MTUs. They connect to a Mosaik server, translating measurements and commands to Panda Power to update the simulation. Terminal units can be configured to use predefined values or send specific commands at designated times.

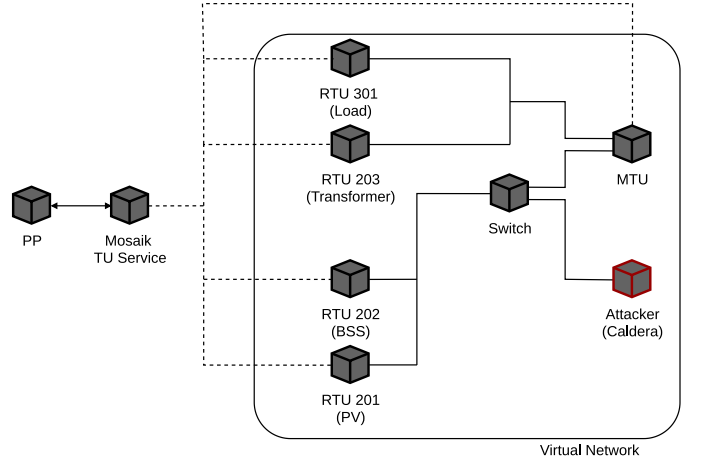


Fig. 1: Topology inside the co-simulation. All components are realized by individual containers. The virtual network emulates local laboratory communication, with only traffic through the switch considered in the network logs.

Rettij initializes virtual networks and starts containers, simplifying network connection definitions. The topology configuration specifies the components and their respective images (see Figure 1 as example). Rettij also handles virtual delays and bandwidth limitations, simulating realistic network conditions for larger grids. With the help a virtual switch, that handles packet forwarding and is developed by the rettij team, we can create virtual networks between relevant components

```

name: Capture Network Traffic (TCPDump with Scapy)
id: 1b27elf8-af08-47eb-b3dc-100c1d697413
platform: linux
command: /bin/python tcpdump.py -t 150
payloads: [tcpdump.py]
cleanup: [/bin/rm tcpdump.py]

```

Fig. 2: Caldera ability for running a Python script.

and capture all passing packets. The virtual network separates the communication of terminal units from Mosaik and Panda Power servers, keeping their messages out of the switch logs. Managing networking between containers is a key challenge. The network must be segmented, with components communicating only with selected others. Rettij allows defining network topology in YAML format, modeling the LAN connections within the laboratory.

The rettij framework uses Kubernetes orchestration, defining components and network connections in virtual smart grids via Docker images. The combination of Mosaik, Panda Power, and rettij provides a robust environment for simulating and analyzing cyberattacks on smart grids.

By integrating modular attacks based on real-world cyberattacks, we ensure flexibility and reduce the need for specialized cybersecurity expertise. Our attacker container autonomously performs specified cyberattacks within the co-simulation, generating realistic and high-quality network logs and power simulation results.

B. Attack Emulation

To complete our scenarios, we need to integrate attacking logic into our simulation environment. This section outlines our approach using MITRE Caldera™, a step-based attack emulation tool that aligns well with our co-simulation needs due to its structured and workflow-oriented nature. Caldera employs a central command-and-control (C2) server with planning algorithms to manage attack steps, facilitating iterative execution of multi-stage attacks within the co-simulation.

Several alternatives exist for attack emulation, namely Atomic Red Team, Splunk and MulVal. But they do not focus on Linux systems for their code, are closed-source or only report assessments according to vulnerability reports.

Caldera on the other hand includes planning capabilities for autonomous execution, high extensibility due to plugins, a modular configuration and is actively developed as an open-source project. The smallest component the attack execution relies on, are the so-called abilities. These are YAML-defined scripts and commands that implement ATT&CK techniques. Each ability is identified by a unique ID and includes executors specifying the platform and command to be run, such as Shell for Linux. Abilities may include payloads downloaded as needed and cleanup commands to minimize traces on victim machines. An example of how these might look is given in Figure 2.

Adversaries are constructs combining multiple abilities to form multi-stage attack plans. Each adversary definition ensures that abilities are executed in sequence, with dependencies checked for each step. For example, an adversary performing

a Denial of Service (DoS) attack might include abilities for network scanning, host enumeration, and attack execution.

Fact sources store and manage data used across agents and abilities, allowing for dynamic execution based on collected or pre-defined information. Facts are labeled values, such as IP addresses or credentials, that can trigger abilities. Rules filter which facts are used, ensuring targeted execution.

Operations combine adversaries and fact sources, specifying execution details such as agent groups, step intervals, and execution mode (automatic or manual). Operations control the overall attack flow, leveraging modular definitions to reuse attack logic and necessary information.

To execute an attack, we follow these steps:

- 1) **Start the C2 Server:** The C2 server orchestrates components, provides payload access, and listens for agent connections. It runs locally without internet access, maintaining all necessary attack information.
- 2) **Connect an Agent:** An agent, a client executable on a compromised machine, connects to the C2 server. It communicates via HTTP packets, providing minimal obfuscation. Agents are categorized by group identifiers for targeted attack execution.
- 3) **Execute an Operation:** We select the agent group and adversary for execution. The operation uses Caldera’s planning module to determine and send instructions to agents based on available facts and adversary requirements. Agents execute commands, collect outputs, and send results back to the server for parsing and further execution.
- 4) **Cleanup:** To minimize traces, agents execute cleanup commands either remotely by the server or as part of adversary completion.

This structured approach ensures detailed control and parallel execution across multiple agents, enhancing the robustness of our attack emulation within the simulated environment.

C. Generative Methods

We tested three state-of-the-art open-source Large Language Models (LLMs) pre-trained on large code bases and focused on code generation, given our hardware constraints (NVIDIA GeForce RTX 3060 Ti with 8 GB VRAM). Code Llama [13] (13 billion parameters), DeepSeek Coder [14] (6.7 billion parameters), and Starcoder 2 [15] (7 billion parameters).

We tested the LLMs’ understanding and generation capabilities through four tasks, including the analysis and generation of abilities and adversaries.

We transformed all local definitions of Caldera’s abilities and adversaries into an embedding, which the LLMs could use as reference during their inference. This process, Retrieval Augmented Retrieval (RAG), was handled by PrivateGPT [16], allowing us to ingest all local YAML files automatically.

V. RESULTS

All results of the co-simulation were run in the scenario depicted in Figure 1 with the same attacker and a set delay for running the final step of the attack after ten minutes.

Inspired by the Confidentiality-Integrity-Availability (CIA) triad we ran three different attacks in addition to the default scenario without any attack going on. For testing the confidentiality we list all accessible files on the infected machine and sent the results back to the attacker, while the integrity of the system was compromised by manipulating the control commands sent by the MTU. Finally disabling the network connections of the infected hosts assessed the impact on availability. In the following we present excerpts from our results.

A. Framework's Applicability

The framework uses YAML files to define simulation parameters, including network topology, simulators, and connections. Key elements include controller and proxy modules for different backends, and global parameters like runtime and step size for synchronization. The network topology specifies IP addresses, bandwidth, and delay. The co-simulation deploys components, including SCADA devices, networking hardware, and power grid components. The developed approach integrates with co-simulation environments using a structured workflow and central C2 server (cf. Figure 3). This server autonomously executes multi-stage cyberattacks, ideal for Mosaik simulations. Components include abilities (predefined scripts), adversaries (execution plans), fact sources (data stores), and operations (execution flow).

In a simulated environment, the approach can execute DoS attacks. Vulnerable terminal units mimic real-world issues like default SSH passwords. An attacker node initializes the C2 server and connects an agent on a compromised machine. The operation starts with a brute-force SSH attack and disables network interfaces. Attacker actions include NMAP scans, Telnet connections, SCP file transfers, netcat use, and shell commands. Configurations test system performance, standard operation, power manipulation, and scenarios causing power spikes, vRTU slowdowns, or shutdowns. Other setups involve Telnet data exfiltration and reconnaissance via NMAP and Telnet logins. The simulation generates outputs, capturing network traffic, logging actions, and creating reports for analysis.

In cyber-physical labs, the approach's lightweight scripts and modular components adapt to various attack scenarios. The C2 server commands infected agents, executing abilities and returning results. This data helps evaluate the effectiveness of countermeasures and understand cyberattack impacts on smart grids.

B. Power Simulation

Looking at the measured power levels via the three-phase power measurement device of the laboratory (Figure 4), we see the balancing of the components in a self-consumption optimization operation scenario. The initial ten minutes of the DoS scenario are almost identical to running the laboratory without any attacker in it. But then the communication between the MTU and RTU is disrupted by the attack and the regulatory commands do not reach the corresponding Distributed Energy

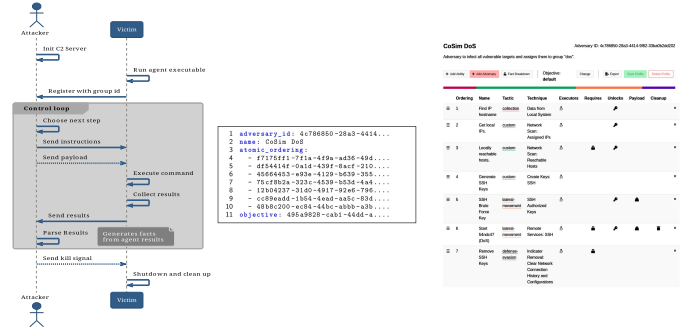


Fig. 3: Control sequence between an attacker using Caldera and a compromised victim.

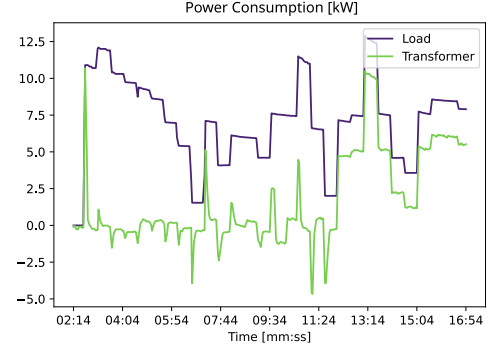


Fig. 4: Measured power curve at substation of the DoS scenario in the laboratory.

Resources (DER). Thus the imbalances occur and the grid sum now mirrors the load induces to the power grid.

Inside the co-simulation we observe the same effect of imbalances after the attack is executed (cf. Figure 5). Noteworthy is the exact timing of when the impact happens, which is contrasted by the timing in the laboratory. This is due to the manual setup that is required when running the physical components, which can easily be automated inside the co-simulation.

C. Timely Spread

As a next metric we inspected the number of packets sent over time. In Figure 6 we see how the amount of network

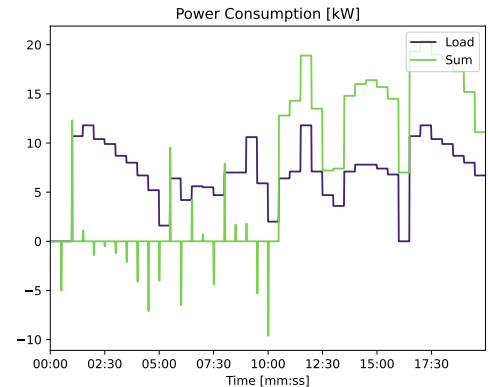


Fig. 5: Measured power curve at substation of the DoS scenario run in the co-simulation.

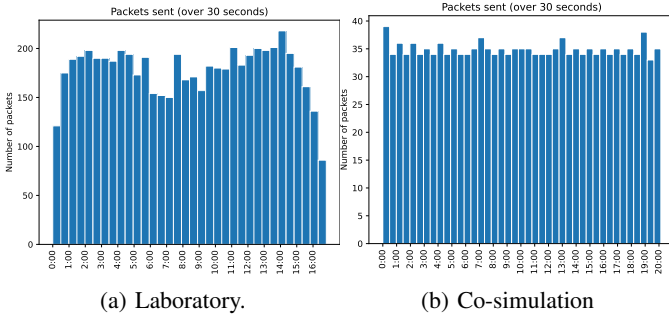


Fig. 6: Packet count over time in normal scenarios.

traffic fluctuates over time inside the laboratory, while the co-simulation produces a more stable and homogeneous amount of network packets during the scenario.

One of our requirements is the consistency of the data, which is hard to ensure inside physical setups due to the warm start of the environment. This means that organisational communications happen before the actual scenario is run, rendering the recorded network traffic incomplete. By fully instantiating the network during the co-simulation, we can collect all of this traffic and reproduce even the organisational communication during each scenario.

D. Connections

Next in line is the connection overview. Here we show how much traffic occurs between the individual hosts. As one key result we see how much data the connections for controlling the terminal units make up. This communication is only required because the Virtual Remote Terminal Units (VRTUs) need to be started by hand to correctly time their execution.

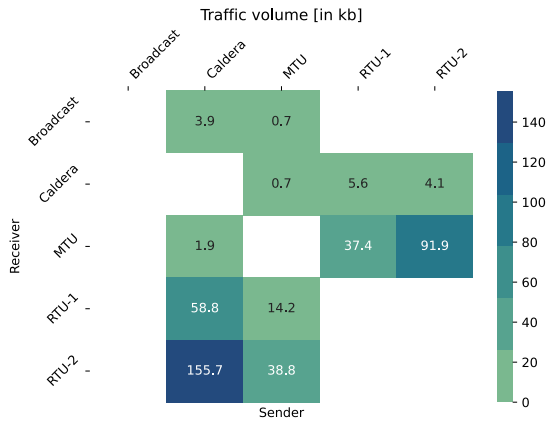


Fig. 7: Network connections in the laboratory.

In Figure 7 we see the default communication without any attack performed. Here the *Caldera* entry refers to the host, that connects to the terminal units to run the required scripts.

When we compare these results to those of the co-simulation in Figure 8, we clearly see how the network log only contains the communication between the MTU and the RTUs and nothing else. Working with clean data allows the data to make sure even slightest irregularities due to the attackers behaviour

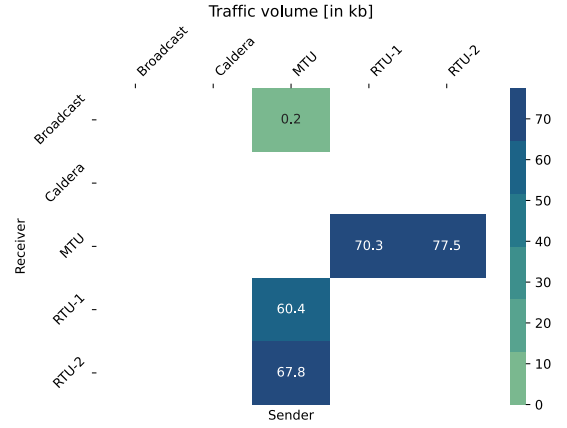


Fig. 8: Network connections in the co-simulation.

TABLE II: Protocol percentages and absolute counts of network traffic for different scenarios (normal, manipulation, and DoS), with total packet numbers below each scenario name.

Protocol	Laboratory			Co-Simulation		
	Normal (1509)	Manip. (3,577)	DoS (2,205)	Normal (1,431)	Manip. (2,588)	DoS (1,792)
IEC 104	0.34	0.15	0.13	0.67	0.39	0.28
SSH	0.54	0.77	0.79	0.00	0.36	0.46
ARP	0.01	0.04	0.07	0.32	0.24	0.24
LLC	0.07	0.04	0.00	0.00	0.00	0.00
Other	0.04	0.01	0.01	0.02	0.02	0.03
IEC 104	510	542	280	956	1008	500
SSH	811	2743	1744	0	926	818
ARP	20	144	164	452	614	429
LLC	113	129	0	0	0	0
Other	55	19	17	23	40	45

can be accounted for, while the laboratory's heterogeneous equipment is prone to potential variations.

E. Protocol Distribution

The impacts of the attacks can also be seen in the composition of different protocols that are contained in the network logs. In Table II we listed the percentages that the most frequent protocols contributed to the overall packet count.

In the laboratory we captured a lot of Secure Shell (SSH) packets originating from the host that had to start the processes manually on the VRTUs. These packets obfuscate the traffic, that is introduced by the attacker and its SSH brute-force ability. In the co-simulation we do not have to rely on manual connections for re-/starting the processes on any component, which reduces the noise inside the exported network log. Another example is the Address Resolution Protocol (ARP) that is only partially captured in the laboratory, while we see all packets in the log of the co-simulation. We also see protocols like the Logic Link Control (LLC) that are introduced by the hardware switch, which only occurred inconsistently during the test in the laboratory and are an example of how unwanted packets can pollute the captured network logs. Even though filtering is a valid option, this

involves granular post-processing of the collected data. The DoS attack nearly halves IEC 104 packet counts in both environments and reduces SSH packet amounts.

F. Further Findings

The generation of data in the laboratory proved to be more volatile due to manually starting the communication process on the involved components, which poses a big challenge for reproducibility. Similarly we observed more noise due to specific hardware like the switches and network taps, that are required for recording network traffic inside the local network.

Furthermore, the fact that the VRTU in the physical laboratory did not expose additional network interfaces for separately running the SSH connections to start their internal workings, introduced a lot of network traffic, that would not occur on productive RTU components.

To conclude the experience from setting up the test scenario inside a physical laboratory, we can say that minimizing the noise was difficult and in the end not completely possible due to the available hardware and communication channels. In general the modular configuration of the co-simulation allows for less restrictive setups and also reduces potential interference with unwanted processes.

Also worth mentioning is the fact, that our autonomous agent, that was developed inside the co-simulation, could immediately run most of the attack steps without any adaption to its configuration. Only the binary that is sent to the target machine needed to be recompiled for the ARM architecture of the VRTU. Besides that, the whole attack scenario could just be run as is.

Finally the results from querying the LLMs did not yield immediately usable code fragments. Instead the generated text contained partially usable code, that still needed to be adapted to make use of Caldera's facts mechanism. Thus the usage of these models might be helpful during the development, they are no source for reliable code, that can be embedded directly into a running attacker.

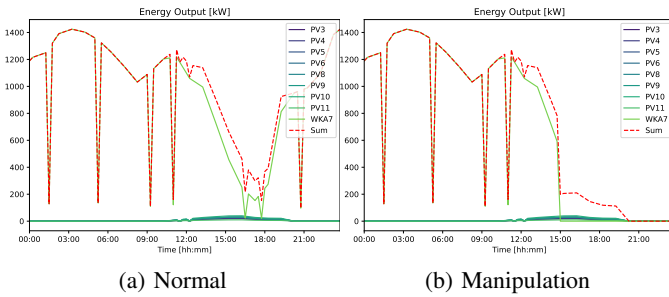


Fig. 9: Power output in the CIGRE MV [17] scenario.

Finally we also successfully ran our framework on the topology suggested by the CIGRE Task Force [17]. Again the attacker was able to adapt to the more complex network without further configurations (see Figure 9). Only the target values for the manipulation had to be set. This indicates the potential flexibility of our framework, but requires further research.

VI. CONCLUSION

With this work we demonstrated how the overall data scarcity for attack data on smart grids can be tackled. We introduced a framework, that allows for configurable data generation and compared the impacts on the physical and simulated smart grids, showing that even though the data is not identical, the impacts the attacks have can be well reproduced. In general the virtually generated data contains less noise due to precise definitions of what happens in the network.

Our proposed methods allowed us to not only develop cyberattacks that could be deployed in the physical laboratory with almost no additional work, but also generated data that achieves higher consistency between different runs. The easily adaptable setup of the virtual environment enabled a fast reproduction of the complex physical setup and thus increased the overall development efficiency.

In the future we hope to see more complex cyberattacks on larger grid topologies. By allowing for quick prototyping and testing of attack methods we also see a lot of potential for more in depth analysis of overall cybersecurity of smart grids without the limitations of openly available datasets.

ACKNOWLEDGMENT

Received funding from the BMBF under project no. 03SF0694A (Beautiful).



REFERENCES

- [1] A. Kenyon *et al.*, "Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets," *Computers & Security*, 2020.
- [2] T. D. Le *et al.*, "Smart grid co-simulation tools: Review and cybersecurity case study," in *icSmartGrid*, 2019.
- [3] T. Choi *et al.*, "Plan2defend: Ai planning for cybersecurity in smart grids," in *ISGT Asia*, 2021.
- [4] S. Vyas *et al.*, "Professor pete burnap. automated cyber defence: A review," *arXiv preprint arXiv:2303.04926*, 2023.
- [5] T. J. Williams, "The purdue enterprise reference architecture," *Computers in industry*, 1994.
- [6] P. Formosa, "Robot autonomy vs. human autonomy: social robots, artificial intelligence (ai), and the nature of autonomy," *Minds and Machines*, 2021.
- [7] Council of European Union, "Artificial intelligence act," 2024. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206>
- [8] X. Li *et al.*, "Securing smart grid: cyber attacks, countermeasures, and challenges," *IEEE Communications Magazine*, 2012.
- [9] A. Ahmadian Ramaki *et al.*, "Causal knowledge analysis for detecting and modeling multi-step attacks," *secur. commun. netw.*, 2016.
- [10] D. van der Velde *et al.*, "Towards a scalable and flexible smart grid co-simulation environment to investigate communication infrastructures for resilient distribution grid operation," in *SEST*, 2021.
- [11] L. Thurner *et al.*, "pandapower — an open-source python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Transactions on Power Systems*, 2018.
- [12] A. Ofenloch *et al.*, "Mosaik 3.0: Combining time-stepped and discrete event simulation," in *OSMSES*, 2022.
- [13] B. Roziere *et al.*, "Code llama: Open foundation models for code," *arXiv:2308.12950*, 2023.
- [14] D. Guo *et al.*, "Deepseek-coder: When the large language model meets programming—the rise of code intelligence," *arXiv:2401.14196*, 2024.
- [15] A. Lozhkov *et al.*, "Starcode 2 and the stack v2: The next generation," *arXiv:2402.19173*, 2024.
- [16] I. Martínez Toro *et al.*, "PrivateGPT," 2023. [Online]. Available: <https://github.com/imartinez/privateGPT>
- [17] K. Rudion *et al.*, "Design of benchmark of medium voltage distribution network for investigation of dg integration," in *IEEE PESGM*, 2006.