

# SERA – An AI Driven Re-Programmable Simulation Environment for Explainable Resiliency Analysis in Cyber-Physical Systems

Rakesh Podder

Colorado State University  
Fort Collins, Colorado, USA  
rakesh.podder@colostate.edu

Indrajit Ray

Colorado State University  
Fort Collins, Colorado, USA  
indrajit.ray@colostate.edu

## Abstract

Testbeds are a practical way to perform security exercises on cyber-physical systems (CPS) to understand vulnerabilities and the progression/impact of cyber-attacks on CPS safety. Most CPS testbeds use a hardware-in-the-loop architecture. However, it limits the testbed's ability to easily reconfigure with new features, new capabilities and new vulnerabilities. Moreover, cyber attacks recreated within the testbed can potentially damage the hardware that the cyber attack targets. These factors suggest a software-based testbed for CPS cyber security and resiliency research. In this paper, we present SERA, an AI-driven, software-based simulation environment for resiliency analysis in CPS. It is specifically geared towards explainable resiliency analysis using the Resiliency Graph model proposed by Bashir et al. [3] and is fully reconfigurable and extendable for experimentation on different types of CPSes.

## CCS Concepts

• **Security and privacy** → **Distributed systems security**; • **Computer systems organization** → *Embedded and cyber-physical systems*; • **Computing methodologies** → *Artificial intelligence*.

## Keywords

Resiliency Graph, AI Agents, LLMs, Cyber Physical Systems

### ACM Reference Format:

Rakesh Podder and Indrajit Ray. 2026. SERA – An AI Driven Re-Programmable Simulation Environment for Explainable Resiliency Analysis in Cyber-Physical Systems. In *Proceedings of the 6th ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (SaT-CPS '26)*, June 23–25, 2026, Frankfurt am Main, Germany. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3806008.3811708>

## 1 Introduction

A Cyber-Physical System (CPS) is composed of an Information Technology (IT) component that connects with the Operational Technology (OT) component in an Industrial Control System and digitalizes the OT. CPS forms the mainstay of many modern day critical infrastructures. Some examples of CPSs are chemical plants, energy distribution grids, gas transmission systems, traffic control

systems, water treatment and supply systems, transportation systems, and others [18, 2]. Digitalization of the OT in industrial control systems has significantly improved operations and management of cyber-physical systems. However, digitalization exposes the OT to novel and advanced cyber attacks that have the potential to cause ICS breakdowns in unanticipated ways. The combination of OT system failures and vulnerability exploits in information technology opens up new attack surfaces in CPS. A single security vulnerability in CPS can lead to catastrophic consequences, which ultimately can cause considerable financial and business loss, human lives, suffering, and others [11, 10]. Additionally, since IT and OT are now interconnected, cyber attacks can trigger failures that can cascade across the CPS and even to multiple interdependent CPSes. Being able to recognize and plan for such situations is critical to building resiliency into the next generation CPS.

Unfortunately, it is challenging to evaluate resiliency in a functioning CPS via testing for security issues. This is because a live CPS cannot afford even a single error or mistake introduced while performing the evaluation. There is a real possibility of the evaluation process damaging the CPS. Consider, for example, a security experiment that needs the introduction of spoofed sensor measurements in the control network of a CPS to simulate an attack. Allowing this experiment to be conducted on a live system will cause it to malfunction. However, the CPS cannot be stopped or interrupted for this purpose. Moreover, such security testing needs to be done periodically since security threats are continuously evolving.

To perform security exercises on cyber-physical systems (CPS), testbeds provide practical way to understand vulnerabilities and the progression/impact of cyber-attacks [21]. The testbed environment emulates the actual behaviors of the different CPS components and simulates their interactions. This allows researchers and engineers to identify security issues in the designed systems before deploying them in the real world and continuously update the knowledge with evolving threats. Since the testbed mimics existing systems' behaviors, it is essential to ensure that different experiments launched in the testbed environment reflect real-world behavior as closely as possible. There have been several works in designing testbeds for CPS cybersecurity [19, 1, 14]. Most of these works have been domain specific such as for smart grids [13], manufacturing [9], water treatment facilities [12], automotive [5], etc. A vast majority of these testbeds use a hardware-in-the-loop architecture. While hardware-in-the-loop architectures may sometimes be unavoidable with current state-of-the-art in CPS testbed design



This work is licensed under a Creative Commons Attribution 4.0 International License. *SaT-CPS '26, Frankfurt am Main, Germany*  
© 2026 Copyright held by the owner/author(s).  
ACM ISBN 979-8-4007-2610-1/2026/06  
<https://doi.org/10.1145/3806008.3811708>

owing to the need to simulate physical processes, it limits the testbed's ability to easily reconfigure with new features, and vulnerabilities. It is also challenging to replicate a large CPS, within the confines of a laboratory with such hardware-in-the-loop architecture that would allow security experiments to be carried out.

Recently, we have seen increasing attention to designing software-based testbed design [7]. Digital twin technology is one of the biggest drivers of this shift. However, we still observe some limitations and some opportunities in this domain. From a cyber security perspective, most of these testbeds are focused on intrusion detection, false data injection and command spoofing on the physical processes via man-in-the-middle (MiM) type attacks. Some of the testbeds allow the user to experiment with potential defenses and see which one works. However, the ability of the testbed to provide any suggestions for defenses and resiliency improvement is quite limited. Rather, the CPS researcher has to come up with the potential defenses and try out these defenses to see if they work.

We see an opportunity here through the adoption of artificial intelligence and machine learning (AI/ML). We had recently proposed *Resiliency Graphs* (RG) [3], as a formal framework that uses AI Planning to analyze how exploiting an IT system's vulnerability can trigger a safety event in the OT system and result in cascading failures. Based on this analysis, the framework can subsequently identify pathways to make the CPS resilient. The Resiliency Graph model draws inspiration from two graph based models that are used respectively in the cybersecurity domain – Attack Graphs – and the ICS safety domain – Fault Trees. *Attack Graphs* (AG) are used by IT professionals to assess their systems for security flaws and implement a robust security posture. Similarly, ICS operators have employed *Fault Trees* (FT) to plan for safety hazards. AG can reveal possible ways for attackers to exploit system vulnerabilities, with a focus on cyber risks, whereas FT depicts the logical linkages between system failures and their root causes. However, neither AG nor FT can by themselves determine how cyber attacks lead to cascading safety failures in CPS, but a combination appears to provide an opportunity, which is what RGs seek to exploit. The RG framework is created by combining a notion of AG to model cyber attacks in the CPS and FT to represent failures. This dual method improves threat detection by uncovering complicated attack scenarios that would be missed if only cyber attacks or system failures were considered. It also improves mitigation tactics by allowing for the development of defenses that handle both cyber attacks and the consequent system failures. Furthermore, it enables the prioritization of security measures in high risk areas and crucial failure points, resulting in more effective resource allocation. The combination also helps non-experts understand and visualize the effect of hazards, allowing them to make better decisions. Overall, this integration provides cross-domain insights that are critical for increasing the resilience and reliability of CPSes. RG employs AI Planning to determine the relationship between vulnerabilities and the problems it can cause. This allows us to scale our approach to very large problem spaces, which is critical for a CPS.

In this paper, we present SERA, an AI-driven software-based simulation environment for resiliency analysis in CPS. It is specifically geared towards explainable resiliency analysis using the Resiliency Graph model [3]. Unlike other CPS testbeds, SERA is not

geared towards specific types of CPS such as, chemical plants, digital power grid or advanced manufacturing. Instead, so long as the physical processes in an ICS can be programmed in Factory I/O, Unity or Unreal Engine, 3D simulation framework for industrial control systems that work with PLC, SERA can be utilized to perform resiliency analysis on that and observe the effects of the cyber attack. SERA is also fully reconfigurable and extendable. It extensively uses agentic AI for easy configuration and setup and AI planning for resiliency analysis. State of the art large language models are used for generating explanations in a natural language.

Such a software-based CPS testbed also requires a careful reexamination of requirements and architectural design different from a CPS testbed for non-security related experiments. In addition to supporting the traditional requirements of cyber-physical system centric experimentations – fidelity, repeatability, scalability, adaptability, measurement precision, diversity and cost-effectiveness – on a resiliency-focused testbed there is a need to run real attack scripts for red-teaming/blue-teaming exercises, which are, in the strictest sense of the term, malicious software in nature. Thus, there is a need to protect the testbed itself from these attack experiments that have the potential to go awry. We discuss how these challenges are addressed in SERA.

The rest of the paper is organized as follows. In Section 2, we summarize the art in CPS testbed design specifically geared towards cyber security research. Section 3 presents an architectural overview of the SERA infrastructure environment including the hardware and the various virtual machines. Section 4 provides a functional description of SERA identifying the major components and their roles with SERA. In Section 5 we describe how the SERA testbed is used to perform resiliency analysis of a CPS and how red-teaming and blue-teaming exercises are adaptively created and deployed to validate the results of the analysis. Finally, Section 6 summarizes the contributions and outlines our next steps.

## 2 Related Work

There have been several attempts in designing testbeds for CPS security research. Most of these works are domain specific such as, smart grids, networks, manufacturing, water process plants, etc.

Siaterlis et al. [20] present EPIC framework that can accurately assess the effects of cyber-attacks on the cyber and physical dimensions of networked critical infrastructures (NCIs), such as power plants. Ashok et al. [1] utilize simulation data from the Internet Scale Event and Attack Generation Environment (ISEAGE) to characterize the system architecture of a security testbed for Power-Cyber developed at Iowa State University. Oyewumi et al. [13] propose an architecture, ISAAC, for performing security experiments on a testbed for smart grid systems. It is a cross-domain, reconfigurable, and distributed framework that simulates data from power generation in operations. It allows researchers to develop, test, evaluate, and validate holistic cyber-physical security techniques for cyber-physical systems and the Smart Grid. Sahu et al. [17] introduce RESLab, a scalable cyber-physical power system testbed combining PowerWorld simulation, network emulation using CORE, and real control devices. The testbed supports evaluation of coordinated cyber attacks and resilience strategies in

power grid environments. Chamana [4] present a smart grid testbed for cybersecurity research. It incorporates SCADA networks, intelligent electronic devices (IEDs), and real-time power simulations. The testbed allows experiments involving intrusion detection, attack propagation, and resiliency analysis. Hosseinzadeh [9] presents a transportation infrastructure CPS testbed focusing on railway level-crossing control systems. The testbed uses single-board computers, networked sensors, and programmable logic controllers to simulate realistic cyber attacks against a small segment of the railway transportation infrastructure. Mathur et al. [12] present the Secure Water Treatment (SWaT) testbed, a scaled version of an operational water treatment facility designed for cybersecurity research. The system includes PLC-controlled physical processes, enabling researchers to study attacks such as sensor spoofing, actuator manipulation, and stealthy process manipulation. Guarino et al. [8] discusses the design of a water distribution testbed that allows intrusion detection systems research for these types of CPS. The platform combines simulated hydraulic models with real control components and provides labeled attack scenarios for machine learning-based detection studies.

A vast majority of these testbeds use a hardware-in-the-loop architecture. (SWaT, in fact is more than a hardware-in-the-loop design. It is a fully functional water treatment facility.) While with current state-of-the-art in CPS testbed design, hardware-in-the-loop architectures may sometimes be unavoidable, it limits the testbed's ability to easily reconfigure with new features, new capabilities and new vulnerabilities. Hardware-in-the-loop also poses the significant possibility that the hardware is damaged from the cyberattacks that are recreated in the testbed making experiment reproducibility extremely expensive. As capabilities of digital twins and frameworks such as Factory I/O<sup>1</sup>, GridLAB-D etc., improve in scale and scope, it is likely that future CPS testbeds can forego hardware-in-the-loop. SERA is built with that vision in mind. But, more importantly from a cyber security perspective, most of these testbeds discussed in the literature are focused on intrusion detection, false data injection and command spoofing on the physical processes via man-in-the-middle (MiM) type attacks. Some of the testbeds allow the user to experiment with potential defenses and see which one works. However, the testbed itself does not provide any suggestions for the defenses. Rather, the user has to come up with the potential defenses. The testbed provides a facility to tryout these defenses and see if they work. The SERA testbed, on the other hand, is significantly autonomous and reconfigurable in its operation. A user can instruct an AI agent to create and configure the testbed using various component libraries. Such as ability allows SERA to switch from being a testbed for a chemical plant to a testbed for a flare system to a testbed for smart manufacturing. SERA is geared towards identifying how a set of vulnerabilities can be exploited to launch an attack (not necessarily just an MiM attack). It uses AI agents to recreate and validate the attacks. SERA autonomously identifies a priority list of courses of actions to make the CPS resilient to these attacks. Finally, AI agents can implement these actions to validate the solution. Explainability is a key feature on SERA at every step. To the best of our knowledge, ours is the first CPS testbed that provides all these features.

<sup>1</sup><https://factoryio.com/>

### 3 SERA Operations and Architectural Design

To present the design of SERA and its capabilities, we briefly present here an overview of how we envisage using SERA for CPS resiliency experiments. Later on, in Section 5, we would elaborate on this process using a running example.

#### 3.1 Overview of SERA Operations

The following steps are followed to use SERA.

- (a) *Step 1* – User specifies CPS structure, components and scenario requirements via SERA interface.
- (b) *Step 2* – SERA interacts with user as needed to program and configure the desired CPS simulation, including integrating the corresponding digital twin of the physical process.
- (c) *Step 3* – User invokes SERA Blue Team Agent (BTA), which scans the instantiated CPS to discover network topology and configuration, vulnerabilities, CPS failures etc. It then hands over the information to SERA's iEXAM module [15].
- (d) *Step 4* – iEXAM gathers together the collected information, generates a planning model for the CPS resiliency analysis and builds a Resiliency Graph [3]. At this stage, the Resiliency Graph shows how resiliency can potentially be compromised. iEXAM's LLM module generates explanations corresponding to this Resiliency Graph.
- (e) *Step 5* – User asks SERA's Red Team Agent (RTA) to validate the Resiliency Graph by generating the actual attacks corresponding to those identified in Step 4. The attacks' effects on the CPS are demonstrated on the digital twin in SERA.
- (f) *Step 6* – User performs what-if analysis by interacting with iEXAM. iEXAM determines a set of resilience paths based on criteria set forth by user.
- (g) *Step 7* – User asks SERA's Blue Team Agent to implement the resiliency solutions suggested in Step 6 for proper validation. Steps 5–7 can be iterated as desired by the user.

Note that SERA is equipped to save partial results from each of the steps 1–7 above for further data analysis and experiment repeatability.

#### 3.2 Overview of SERA Infrastructure

Figure 1 provides an architectural overview of the SERA infrastructure that supports CPS resiliency experimentation on SERA as discussed above. The major components (called nodes henceforth) are as follows:

- (1) **SERA Hardware** consists of (a) Two rack servers that are configured to serve the roles of the Compute Server, Backup Server and Data Management Server. One of the servers has 52 CPU cores at 2.1 GHz, 385 GB RAM and a RAID array with 6.5 TB of usable disk space. The other server has 64 CPU cores at 2.1 GHz, 512 GB RAM and a RAID array with 7 TB of usable disk space. The Compute Server is the main server that instantiates the SERA environment for resiliency experimentation. During experiments, it instantiates all virtual machines, the iEXAM framework for explainable resiliency analysis, the Blue Team Agents (BTAs) and the Red Team Agents (RTAs), and the Digital Twins. The

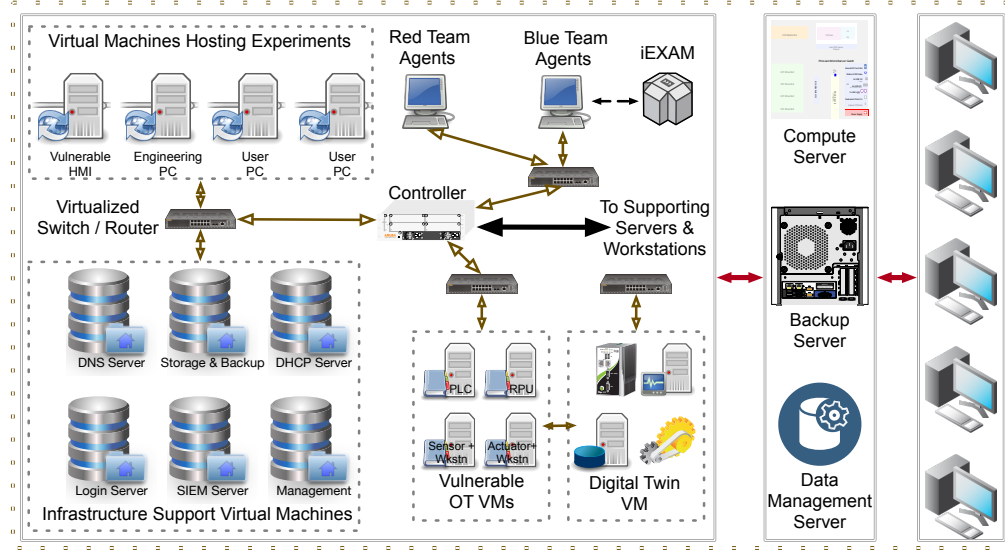


Figure 1: SERA Infrastructure Architecture

Data Management Server hosts various images of vulnerable machines, images of applications that are used in various virtual machines, such as, the infrastructure support virtual machines, 3D simulation libraries etc., and codes for the BTAs and RTAs, LLM agents used for explainability. The Backup server is used to back up all data produced or consumed in the testbed as well as snapshots of paused experiments. (b) Several user workstations that are used to access the SERA infrastructure to configure and run experiments.

- (2) **Virtual Machines (VMs)** SERA extensively uses virtualization technology for creating the CPS environment whose resiliency properties are being evaluated. Virtualization allows easy scalability, adaptability, diversity and cost-effective management of simulation experiments. Broadly, these VMs can be categorized into five types:

- VMs for hosting experiments. These machines host various IT components of the CPS, such as the vulnerable HMI, Engineering PCs, Users etc. Together with the Vulnerable OT VMs and the Digital Twin VMs, these form the core of the CPS being investigated.
- Vulnerable OT VMs. These are the machines that introduce the vulnerable PLCs, RPUs, Workstations that control the sensors, actuators etc., in the CPS and via which attacks can be launched on the CPS.
- Digital Twin VMs. These VMs provide the 3D simulation framework for the ICS. We use Factory I/O in SERA.
- Virtual machines to support the infrastructure needed for the experiments. These include servers such as DNS, DHCP, SIEM, Network management, User management etc. The BTAs and RTAs use these VMs to run experiments.
- Virtualized switches and routers for interconnecting different components. The Controller is a major component. It is responsible for managing resources for experiment

nodes. It also provides services to allocate network address spaces for the experiment nodes as required by the experiments requirements. The Controller is provided with overall knowledge of the availability of resources, user space, node visualization, shell management, etc., when the CPS is set up. Note that if SERA is used to host multiple CPS experiments on different types of CPS environments, then we recommend that the Controller be hosted on a separate physical machine so that the other nodes (some of which would intentionally be containing malware for experiment) cannot manipulate the controller's memory space thereby affecting fidelity and measurement precision of experiments.

## 4 Functional Components of SERA

SERA is an AI-driven, software-based simulation environment designed to run repeatable CPS resiliency experiments safely. The resiliency analysis comprises of automatically evaluating if the CPS is vulnerable to cascading failures triggered by cyber attacks. SERA establishes this first by enumerating *resilience paths* as defined in [3]. It then allows red teaming on the CPS to validate the outcomes of this analysis. SERA further provides suggestions on how to make the vulnerable CPS resilient and then allows blue teaming to implement those changes. The SERA user can iterate through this process until satisfied.

It supports realistic IT and OT components, runs a simulated physical process through Digital Twin (i.e., Factory I/O, Unity/Unreal Engine + GridLAB-D), enables Red Team and Blue Team actions using agents, collects telemetry, and produces explainable resiliency analysis through iEXAM and LLM-based narratives.

### 4.1 iEXAM Module for SERA

At the heart of SERA is the iEXAM module, which builds upon the Resiliency Graph framework presented in [3] and incorporates

LLMs for ease of interaction by the CPS operators for what-if analysis. A schematic architecture of iEXAM and the corresponding analysis workflow is given in Figure 2. iEXAM consists of a set of tools based on Machine Learning and AI Planning that enables operators to analyze how a cyber-attack can trigger safety events in CPS and/or cascading failures and then interactively determine potential approaches to mitigate threats. The Resiliency Graph framework is based on the notion of transition system [16] and is built through a composition of attack graphs representing cyber-attacks in the CPS and fault trees representing failures. The mathematical basis that allows attack graphs to be composed with fault trees is the observation that each of these can be expressed in terms of a set of related state transitions. A state transition is effectuated by an action on an initial state and culminates in a resulting state. An initial state is a set of pre-conditions that specify where the action can be executed; the resulting state is a set of post-conditions that specify the result of the action. Such a composed transition system can capture the different ways a cyber-attack can cause safety issues or cascading failures. The transition system is represented as an AI Planning problem in the Planning Domain Definition Language (PDDL) [15]. The reasoning engine is based on AI Planning that generates *plans* that identify the various attack events in the CPS, which can lead to cascading failures and safety issues. An LLM module in iEXAM takes these plans and converts them to natural language text for easy interpretation and understanding by the CPS operator. iEXAM allows the operator to interact with the AI planner to query the underlying transition system and perform what-if analysis. This analysis provides actionable suggestions or recommendations [15] from the tool including insights into potential attack vectors and help prioritize efforts to secure critical assets effectively. These suggestions comprise of a diverse set of solutions each of which can potentially take the CPS to a safe and secure state. To help the operator decide which of the actionable suggestions to implement, the toolset provides explanations in natural language, also generated by the LLM module.

## 4.2 Vulnerable IT VMs

In SERA, vulnerable IT virtual machines are created as controlled experiment assets rather than random outdated systems. The main idea is to build machines that are repeatable, isolated, measurable, and easy to reset so that the same attack chain can be reproduced across multiple runs. Each vulnerable IT VM described in Table 1, is intentionally prepared from a known baseline (a known CVE or known attack), and frozen into a snapshot or template (here as an .OVA file) so the Controller can clone fresh instances for every experiment run. We create vulnerable IT VMs using three categories of weaknesses:

**4.2.1 Vulnerable applications (app-layer CVEs).** The first category is vulnerable applications at the application layer, where exploitation happens through web apps or higher-level software logic. This includes scenarios like a software (i.e., WordPress, Drupal, Joomla, Web server, ScadaBR (HMI)) with a vulnerable plugin version, an outdated web admin portal, or a service with a known application CVE. These VMs are built by installing the application stack and its dependencies, pinning the vulnerable versions, disabling automatic updates, SSH, Web logs, application logs, and authentication

logs are enabled so the compromise path produces evidence for Blue Team Agents that can later be explained.

**4.2.2 Vulnerable services (protocol/service CVEs).** The second category is vulnerable services at the protocol or daemon level. These VMs represent weaknesses in services such as HTTP, SMB, FTP, SSH, databases, or other network-facing software where scanning and service exploitation is a realistic entry point. The creation process focuses on installing a known vulnerable version confirming that the service is reachable and behaves as expected, and opening only the required ports. Since these vulnerabilities often drive early attack graph steps like discovery and exploitation, we make sure the service banner, network behavior, and authentication events are stable and observable in telemetry.

**4.2.3 Misconfiguration vulnerabilities (non-CVE weaknesses).** The third category is misconfiguration vulnerabilities that are not tied to a CVE. These represent the kinds of enterprise weaknesses that frequently enable lateral movement and privilege escalation, such as weak passwords, exposed admin interfaces, over-permissive shares, plaintext credentials left on disk, unsafe sudo rules, excessive local admin privileges, or overly permissive firewall rules inside a host. They are seeded in a way that makes them actionable in an attack chain, for example by placing a credential artifact that enables pivoting to another IT or OT node.

These vulnerabilities are deployed across diverse operating systems, including Windows workstations and servers (Windows 8/8.1/10, Windows Server 2008/2008 R2), multiple Linux distributions (Ubuntu 20.04/22.04 LTS, Debian 8/9/10, Fedora 35/36/38, CentOS Linux 5/6/7, Red Hat Linux 7/8/9), and legacy operating systems. This OS diversity is intentional because real IT environments contain mixed platforms and older systems that cannot be upgraded easily. Automatic updates are disabled to prevent vulnerabilities from silently disappearing, installers and packages are stored locally within the lab if needed, and these VMs are placed behind strict routing and firewall controls so they cannot reach sensitive management components or escape the experiment boundaries.

Some of the vulnerable IT VMs also host vulnerabilities through Docker<sup>2</sup>. In this mode, the VM acts as a stable host platform, while multiple vulnerable applications and services run as containers. This lets SERA pack several targets into a single VM, pin versions cleanly using container tags, and reset quickly by restarting containers and resetting volumes. Docker can host both application-layer CVEs and protocol/service vulnerabilities. This is useful because it mirrors how modern enterprises deploy services while still allowing realistic attacker paths.

## 4.3 Vulnerable OT VMs

Vulnerable OT VMs in SERA represent the operational technology side of the CPS, and they are the components through which cyber actions can directly influence the simulated physical process. These machines introduce OT assets such as PLCs, RPUs, and OT workstations that interact with sensors and actuators, making them the primary bridge between attacker activity and process-level consequences. In our setup, these OT VMs are intentionally designed to

<sup>2</sup><https://github.com/vulhub/vulhub>: Vulhub is an open-source collection of pre-built, ready-to-use vulnerable Docker environments

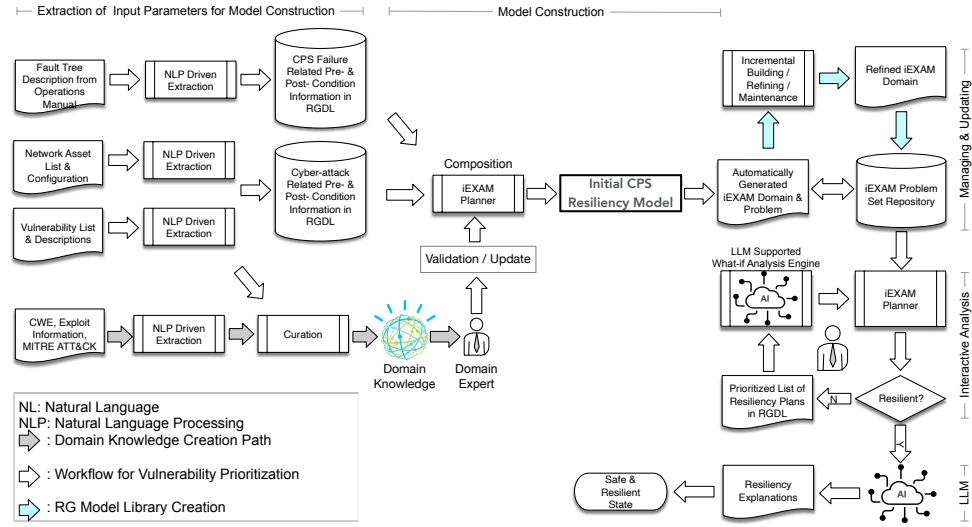


Figure 2: iEXAM Schematic Architecture

Table 1: IT VM Stack in SERA

| SL No. | VM Name      | OS                     | Computation                  | CVE/Weakness          | Vul Type                           |
|--------|--------------|------------------------|------------------------------|-----------------------|------------------------------------|
| 1      | Lab-Host-001 | Ubuntu Server (20.04)  | RAM:2GB, 1CPU, Storage: 10GB | CVE-2011-2523         | Service: vsftpd 2.3.4              |
| 2      | Lab-Host-002 | Debian 10              | RAM:2GB, 1CPU, Storage: 10GB | Multiple <sup>1</sup> | Service: httpd-2.4.49              |
| 3      | Lab-Host-003 | Linux 2.4              | RAM:1GB, 1CPU, Storage: 5GB  | CVE-2012-1823         | Application: PHP-CGI               |
| 4      | Lab-Host-004 | Red Hat Linux          | RAM:4GB, 2CPU, Storage: 20GB | CVE-2026-1400         | Application: WordPress AI Plugin   |
| 5      | Lab-Host-005 | Ubuntu Server (20.04)  | RAM:2GB, 2CPU, Storage: 10GB | CVE-2017-5638         | Service: apache-struts2 via docker |
| ⋮      | ⋮            | ⋮                      | ⋮                            | ⋮                     | ⋮                                  |
| 40     | Lab-Host-040 | Windows 2008 R2 Server | RAM:2GB, 2CPU, Storage: 30GB | CVE-2017-0144         | Service: SMBv1                     |

<sup>1</sup><https://www.cvedetails.com/version/668692/Apache-Http-Server-2.4.49.html>

be exploitable so that attacks can be launched against control logic, command channels, and operator workflows, and the resulting impact can be observed in the CPS simulation.

A major OT element in our environment is built around Open-PLC Editor<sup>3</sup> using a deliberately vulnerable version. OpenPLC is used to author and manage PLC logic, and by running it in a controlled VM we obtain a realistic control-programming surface that attackers can target. This VM captures the PLC engineering workflow, including storing and editing ladder logic or structured control programs, deploying logic to a PLC runtime, and interacting with OT protocols used to communicate with the process.

Alongside the PLC programming environment, SERA includes OT workstations that are responsible for controlling and supervising PLC logic and operational behavior. These workstations act as the “brain” of the control plane in the OT zone, performing tasks such as loading logic projects, pushing updates to PLC runtimes, and interfacing with control network services. They are modeled as separate assets because in real deployments these systems are frequently distinct from the PLC device itself, and they often become the attacker’s preferred target due to their richer and larger attack surface. Similar to the vulnerable IT VM methodology, OT VMs incorporate three categories of weaknesses so that we can study different classes of attack paths and their physical outcomes.

<sup>3</sup><https://autonomylogic.com/>: We are using vulnerable version v2.0, v2.1 etc.

#### 4.4 Digital Twin VMs

Digital Twin VMs provide the physical-process simulation layer in SERA. They host domain-specific simulators that generate time-series “ground truth” signals such as sensor readings, actuator states, and system performance variables. This allows SERA to study cyber-to-physical impact, without requiring physical hardware testbeds, and to align cyber events with measurable process deviations for resiliency analysis and explainability.

For factory and industrial ICS scenarios, SERA uses Factory I/O VM or Unity VM to model 3D production environments and PLC-driven automation loops [7]. For power-system scenarios, SERA instantiates grid simulation VMs using OpenDSS, pandapower, and GridLAB-D to capture electrical network behavior such as power, voltage profiles, and stability metrics. When experiments require multi-simulator interaction, SERA uses HELICS as a co-simulation to synchronize time and exchange signals across digital twins.

For visualization and operator-facing experiments, SERA supports 3D rendering platforms including Unity and Unreal Engine, as well as web-based 3D visualization stacks such as CesiumJS, Babylon.js, and Three.js. Together, these Digital Twin VMs enable SERA to generalize across CPS domains, support PowerGrid ICS experimentation, and provide intuitive visual and data-driven evidence of cyber-induced physical impact.

## 4.5 Infrastructure Support VMs

Infrastructure Support VMs provide the shared services required to run SERA experiments in a realistic and manageable way. These virtual machines host core enterprise-style capabilities such as DNS (CoreDNS, Pi-Hole, BIND 9, dnsmasq) and DHCP (currently using VirtualBox, VMware, Proxmox inbuilt DHCP server) for naming and address assignment, SIEM tool such as Wazuh, Security Onion, OpenSearch, Graylog for centralized log ingestion and alerting, network management services for monitoring and controlling segmentation and routing policies, and user management services for accounts, authentication, and role-based access. We are using various open-source firewall such as OPNsense, pfSense, OpenWrt for each subnet (IT-DMZ, IT-Trusted, OT-DMZ, OT-Trusted). Instead of embedding these functions inside experiment nodes, SERA keeps them as dedicated support VMs so experiments remain scalable, and easier to reset without losing global observability.

## 4.6 AI Agents for SERA

In SERA, we are incorporating cutting edge AI technology that could elevate security analysis of CPS specially ICS to a level where it could help system administrators to make informed decision and reduces risk of a security escalation. Let it be the Red Team Agents (RTAs) that can validate and test security vulnerability automatically or be the Blue Team Agents (BTAs) with AI planning powered iEXAM tool that can analyze the security loopholes and comes up with solutions to stop any security breaches.

**4.6.1 Blue Team Agent.** BTA is the defensive “operator” in our security experiment. In SERA, it is an agentic system (LLM + MCP-tools) as illustrated in Figure 3 that watches the environment, detects suspicious activity, understands what is happening, relays that information to iEXAM for actionable solutions and executes/implements actions to reduce risk and restore safe operation.

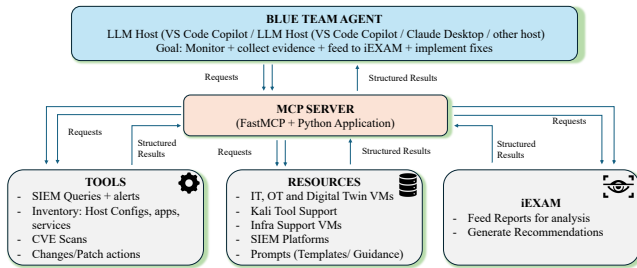


Figure 3: Architecture for Blue Team AI Agent

As illustrated in Algo 1, the job of the BTA is to continuously monitor all IT, OT, and Digital Twin VMs, collect and correlate evidence from telemetry sources (SIEM alerts, host logs, network & application logs, and process traces), and turn that evidence into actionable security insight. Practically, it builds and maintains an up-to-date picture of each VM’s state, including host configuration, running services, installed applications, exposed ports, and known weaknesses (CVE-based vulnerabilities). BTA has the permission to access and modify the state of any machine in SERA.

After iEXAM produces recommendations, the BTA implements the changes when permitted. These changes may include patching

### ALGORITHM 1: Working Algorithm for BTA

```

Input : Login_info_VMs, vm_list, Prompts, LLMs, MCPTools, userRequests,
        iEXAM_recommendations
Output: Reports

for each request ∈ userRequests do
    if request == VMs_state_Check then
        for each vm ∈ vm_list do
            vmState[vm] ← invoke(MCPTools, CheckState(vm));
        end
        Reports ← generateReports(vmState);
    end
    else if request == patch_vms then
        changePlan ← process(MCPTools, iEXAM_recommendations);
        for each vm ∈ vm_list do
            session[vm] ← doSSH(MCPTools, Login_info_VMs, vm);
            vmState[vm] ← applyChanges(MCPTools, session[vm],
                                      changeState(vm, changePlan));
        end
        Reports ← generateReports(vmState, changePlan);
    end
end
return Reports;

```

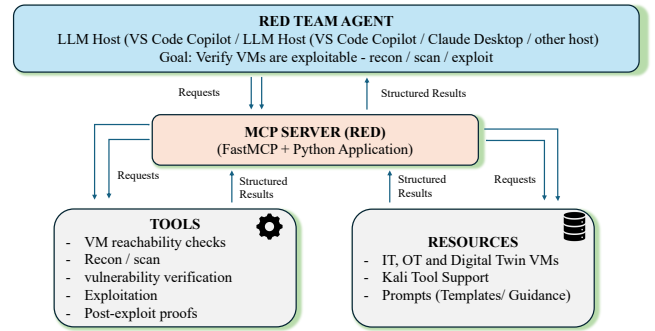


Figure 4: Architecture for Red Team AI Agent

or upgrading vulnerable applications, disabling risky services or plugins, blocking ports, updating firewall rules, or rolling back to a known-good snapshot. Finally, it validates that the mitigation worked by checking system health, reduced exposure, and confirming through telemetry that alerts drop and behavior returns to expected levels, while recording artifacts and generating a clear timeline and report of what happened and what was done.

**4.6.2 Red Team Agent.** RTA is the offensive “attacker role” in our security experiment. In SERA, it is an agentic system (LLM running in a host like VS Code Copilot or Claude Desktop, connected to tools through an MCP server) as shown in Figure 4 that tries to compromise the target IT and OT VMs in a controlled way so we can measure how exploitable the environment is.

As described in Algo. 2, the job of the RTA is to verify exploitability and demonstrate realistic attack paths end to end. It performs reconnaissance to discover reachable hosts and services, scans for exposed ports and applications, enumerates versions and configurations, and maps those findings to potential weaknesses, including application-layer CVEs, service/protocol CVEs, and misconfiguration-based weaknesses. When an exploit path is possible, it attempts exploitation using approved tooling, then collects proof of compromise such as a shell, privileged access, sensitive file access, or the ability to modify control logic in OT.

**ALGORITHM 2: Working Algorithm for RTA**

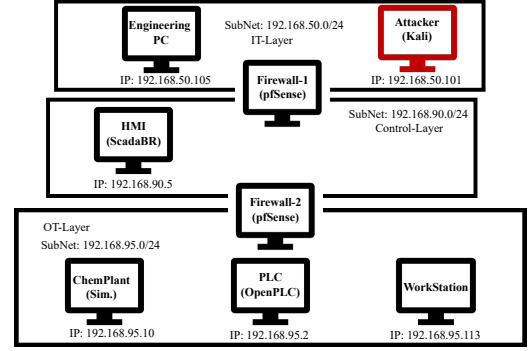

---

```

Input :vm_list, Prompts, LLMs, MCPTools, userRequests, exploitProfiles
Output:Reports
agentAction ← recon_scan, verify_exploit, exploit_vms;
for each request ∈ userRequests do
  if agentAction == recon_scan then
    for each vm ∈ vm_list do
      netInfo[vm] ← invoke(MCPTools, DiscoverHost(vm));
      ports[vm] ← invoke(MCPTools, PortScan(vm));
      services[vm] ← invoke(MCPTools, ServiceEnum(vm, ports[vm]));
      webInfo[vm] ← invoke(MCPTools, WebEnum(vm));
    end
    findings ← correlate(netInfo, ports, services, webInfo);
    report ← generateReports(findings);
  end
  if agentAction == verify_exploit then
    for each vm ∈ vm_list do
      candidates[vm] ← invoke(MCPTools, FindExploitCandidates(vm,
        services[vm], webInfo[vm]));
      verified[vm] ← invoke(MCPTools, VerifyVuln(vm,
        candidates[vm]));
    end
    report ← generateReports(verified);
  end
  if agentAction == exploit_vms then
    for each vm ∈ vm_list do
      profile ← selectProfile(exploitProfiles, vm);
      result[vm] ← invoke(MCPTools, RunExploit(vm, profile));
      proof[vm] ← invoke(MCPTools, CollectProof(vm));
      artifacts[vm] ← invoke(MCPTools, ExportArtifacts(vm));
    end
    report ← generateReports(result, proof, artifacts);
  end
end
return Reports;

```

---

**Figure 5: Simulated Testbed Generated by Controller****Figure 6: Simulated Tennessee Eastman Process Plant**

In the SERA workflow, the RTAs work before and after the BTAs and iEXAM. It helps to answer questions like which VMs are actually exploitable, what the minimal attack chain is, and which vulnerabilities or misconfigurations most strongly enable escalation from IT toward OT and ultimately physical impact in the digital twin. And after BTA applies iEXAM’s solution, RTA re-verifies whether exploitation still exists.

#### 4.7 Controller UI

The Controller User Interface (UI) is the single dashboard for operating SERA. It does not act like the RTA or BTA. Its job is to let user sees and interacts with everything in one place. It visualizes the full experiment topology across IT, OT, digital twin, and infrastructure services. Each VM is shown with key details like role, IP, OS, zone, and live status so one can quickly understand connectivity and where activity is happening. It also manages experiment lifecycle actions such as starting and stopping runs, cloning VMs, taking snapshots, rollback, and resetting the environment. This is what makes experiments repeatable and prevents drift after attacks.

The UI provides unified access to machines through SSH, RDP, web consoles, or hypervisor console links, so user does not have to manually track credentials and endpoints across dozens of VMs. Finally, the Controller UI centralizes observability and agent supervision. It links to SIEM dashboards, logs, alerts, packet captures, and digital twin traces, and it shows RTA and BTA’s actions, and returned outputs so user can audit what happened during the run.

## 5 Resiliency Analysis using SERA

In this section, we describe how resiliency analysis is performed using SERA and how we evaluate SERA through controlled experiments. The workflow begins with the user specifying CPS components, including the IT & OT assets, the digital twin simulator, and the desired network segmentation. This specification serves as the experiment blueprint. At present, the blueprint is provided through command-line arguments and an interactive agent that configures SERA based on user inputs. We are currently designing a dedicated UI so that users can visually configure and edit the testbench. Figure 5 shows the graphical view of the testbench and the VMs instantiated by SERA. In our setup, the Engineering PC is lab-host-040 from the IT VM stack, the HMI is an OT VM running ScadaBR, the PLC is an OT VM running OpenPLC v2, and the OT workstation is an Ubuntu-based OT VM that configures and manages PLC logic. Firewalls are instantiated as infrastructure support VMs such as VMs with pfSense to enforce isolation between layers. The chemical plant is represented by a Digital Twin VM that provides a 3D visualization of the simulated Tennessee Eastman process plant, as shown in Figure 6. Each OT digital twin has a pre-defined Fault Tree that is later used during resiliency analysis.

Once the testbench (TB-1) is up and running, the user invokes the BTA and uses a natural-language prompts, for example, “*Run a comprehensive scan (use collect all evidence tool) on TB-1 and collect all the data. You have all the permissions and use SSH login details if necessary*”. We are using Copilot with VS Code for LLM model.

**Table 2: Red Team Agent Trace of CVE-2017-0144 Exploitation**

| Stage         | Results                                                                                                                                                           |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MCP Tool Call | <b>nmap_target_scan</b> (target='192.168.50.105', options='-sC -sV -Pn')                                                                                          |
| LLM Response  | 135/tcp, open, msrpc, Microsoft Windows RPC ...445/tcp, open, microsoft-ds, Windows Server 2008 R2 Enterprise 7601 Service Pack 1 microsoft-ds.                   |
| MCP Tool Call | <b>metasploit_search_cve</b> (target='192.168.50.105', cve='CVE-2017-0144')                                                                                       |
| LLM Response  | Exploit Name: exploit/windows/smb/ms17_010_eternalblue                                                                                                            |
| MCP Tool Call | <b>metasploit_set_exploit</b> (name='exploit/windows/smb/ms17_010_eternalblue', target='192.168.50.105', attacker='192.168.50.101')                               |
| LLM Response  | used exploit/windows/smb/ms17_010_eternalblue, set rhosts to 192.168.50.105, set lhost to 192.168.50.101, default payload is windows/x64/meterpreter/reverse_tcp. |
| MCP Tool Call | <b>metasploit_check</b> ()                                                                                                                                        |
| LLM Response  | 192.168.50.105:445 - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Enterprise 7601 Service Pack 1 x64 (64-bit).                                 |
| MCP Tool Call | <b>metasploit_set_payload</b> (payload='windows/x64/shell_reverse_tcp')                                                                                           |
| LLM Response  | Exploited successfully. Command shell session 2 opened (192.168.50.101:4444 -> 192.168.50.105:49160) at 2026-03-10 03:26:15 -0400.                                |

**Table 3: Red Team Agent Trace of Buffer Overflow Attack on PLC.**

| Stage         | Results                                                                                                                                                                                                             |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MCP Tool Call | <b>wireshark_target_scan</b> (source='192.168.90.5', destination='192.168.95.2', protocol='Modbus')                                                                                                                 |
| LLM Response  | 0... .... = Exception: No, .000 0001 = Function Code: Read Coils (1), Reference Number: 40, Bit Count: 1.                                                                                                           |
| MCP Tool Call | <b>metasploit_search_exploit</b> (type='modbus')                                                                                                                                                                    |
| LLM Response  | auxiliary/analyze/modbus_zip, auxiliary/scanner/scada/modbus_banner_grabbin, auxiliary/scanner/scada/modbusclient, ...                                                                                              |
| MCP Tool Call | <b>metasploit_modbus_info</b> (type='auxiliary/scanner/scada/modbusclient')                                                                                                                                         |
| LLM Response  | As the source is Scada and client is PLC using modbus, auxiliary/scanner/scada/modbusclient is used. Required DATA_ADDRESS, DATA_COILS, NUMBER, RHOSTS and UNIT_NUMBERS.                                            |
| MCP Tool Call | <b>metasploit_modbus_exploit</b> (destination='192.168.95.2', action='WRITE_COILS', data_address='40', number='1')                                                                                                  |
| LLM Response  | Exploited successfully. Running module against 192.168.95.2, 192.168.95.2:502 - Sending WRITE COILS..., 192.168.95.2:502 - Values 1 successfully written from coil address 40, Auxiliary module execution completed |

BTA collects all the network configurations (by *nmap*), host information (by SSH), CVE data (by *nuclei*), connectivity data (through *traces*) to create a comprehensive report. We have created automated scripts that parse this report along with Fault Tree to convert it into PDDL files (domain.pddl, problem.pddl) which iEXAM uses to analyze and find a resilience plan or path as shown in below. This resilience path explains if there is a possible attack path whether a path from IT to OT exists for a specific fault or goal condition, i.e., reactor/chemical plant shut down (fault-rex-sht-dwn).

#### Resilience Plan/Path (IT → OT)

```
(attacker-exploit-CVE-2017-0144 it-endpoint-user-192-dot-168-dot-50-dot-105)
(it-endpoint-user-compromised it-endpoint-user-192-dot-168-dot-50-dot-105)
(attacker-discover-hmi it-endpoint-user 192-dot-168-dot-90-dot-5)
(attacker-discover-plc 192-dot-168-dot-90-dot-5 192-dot-168-dot-95-dot-2)
(attacker-connected-to-plc 192-dot-168-dot-95-dot-2)
(attacker-reset-the-reactor-pressure 192-dot-168-dot-95-dot-2)
(cause-fault-rex-press-dcr )
(cause-fault-rex-sht-dwn )
```

iEXAM explains the resilience path in natural language using LLM (we are using NVIDIA llama-3.1-nemotron-70b-instruct [6]) as shown below.

#### LLM Generated Explanations

```
step 1: The attacker exploited CVE-2017-0144 on the user IT endpoint at 192.168.50.105.
step 2: The attacker compromised the user IT endpoint at 192.168.50.105.
step 3: From the compromised IT endpoint, the attacker discovered the HMI at 192.168.90.5.
```

```
step 4: Through the HMI, the attacker discovered the PLC at 192.168.95.2.
step 5: The attacker established a connection to the PLC at 192.168.95.2.
step 6: The attacker reset the reactor pressure through the PLC at 192.168.95.2.
step 7: This action caused the reactor pressure to decrease.
step 8: The pressure fault ultimately caused the reactor to shut down.
```

Next, the user invokes the RTA to validate exploitability of the plan, whether the predicted IT to OT attack path can be executed in the live testbench. Once RTA's activation confirmed, LLM creates a plan of action and start exploitation step by step according to the resilience path. Table 2 shows the actions performed and response of LLM while compromising target 192.168.50.105 (*step 1*).

Then it confirmed the exploitation by creating an admin user in the target machine (*step 2*) and find the connected HMI's gateway (*step 3*). After that RTA finds the gateway, it performs an arp-scan on HMI (sudo arpspoof 192.168.90.5) and discover the PLC VM (*step 4*). RLA creates another virtual router for 192.168.95.0/24 (*step 5*) and then starts exploiting PLC (192.168.95.2) which is described in Table 3 (*step 6*). Once the exploitation is successful, it resets the data coil (the OpenPLC v2 target is vulnerable to a buffer overflow exploit) and the pressure of the main tank drastically reduces which forces a shutdown of reactor as shown in Figure 7.

iEXAM not only finds paths, it also provides diverse set of mitigation options on how to stop an attack to restore resiliency. For the above plan, iEXAM generates a solution such as “*update Microsoft Windows SMB Server*”. Once the RTA validates the exploitability, BTA parses the solution, SSH into target machine (192.168.50.105) using MCP tool, and installed the updated SMB server. For experiment, we kept the updating file (.exe) in ‘*Desktop/solution*’ folder and observe whether the BTA is able to install it. Once the BTA



**Figure 7: TE Process Plant’s Shutdown where Cyber-attack Triggers Pressure Sensor Failure (Fault) of Main Tank.**

applies the solution, we invoke the RTA again, and find that the exploitation fails for ‘lab-host-040’ which restores resiliency of the testbench.

## 6 Conclusions and Future Work

In this paper, we present SERA, an AI-driven, software-based simulation environment for explainable resiliency analysis in CPS. SERA can be programmed to simulate different types of CPS environments so long as the physical processes of the ICS can be built as digital twins in Factory I/O or Unity, which is a 3D simulation framework for ICS. Once a CPS is programmed into SERA, AI agents can automatically scan for vulnerabilities and create the CPS Resiliency Graph as described by Bashir et al. [3]. SERA extensively uses agentic AI for easy configuration and setup and AI planning for resiliency analysis. State of the art LLMs are used for generating explanations in a natural language. By using these techniques, SERA evaluates how cyber attacks can trigger cascading failures in the CPS and suggests potential changes to make the CPS more resilient. SERA also allows the results of this evaluation to be validated via recreating cyber attacks and implementing defenses.

There are several dimensions along which we plan to extend SERA. First, we have not yet explored techniques to make SERA a multi-user / multi-experiment testbed. The main challenge in this aspect is how to ensure how one user/experiment’s activities do not impact another one. Isolation of user spaces/experiments is very important. We are exploring how containerization principles can be used in this context. This, in turn opens up our second planned extension of SERA, which is how to enable collaboration among different containerized experiments without affecting fidelity and measurement accuracy of each individual experiment. This will be very important in order for SERA to support resiliency experiments where cyber attack on one CPS (for example, a smart grid) causes attack and failure to cascade to a different CPS (for example, water treatment plant). Finally, to keep all options open, we plan to explore how a hardware-in-the-loop setup can be integrated with SERA.

## Acknowledgment

This work was made possible in part by funding from the U.S. Office of Naval Research under contract N000142612041. The opinions, results, conclusions, and suggestions presented in this work

are those of the authors and do not necessarily represent those of the Office of Naval Research or other organizations and agencies.

## References

- [1] Aditya Ashok, Adam Hahn, and Manimaran Govindarasu. 2011. A cyber-physical security testbed for smart grid: system architecture and studies. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, 1–1.
- [2] Ayan Banerjee, Krishna K Venkatasubramanian, Tridib Mukherjee, and Sandeep Kumar S Gupta. 2011. Ensuring safety, security, and sustainability of mission-critical cyber-physical systems. *Proceedings of the IEEE*, 100, 1, 283–299.
- [3] Shadaab Kawnain Bashir, Rakesh Podder, Sarath Sreedharan, Indrakshi Ray, and Indrajit Ray. 2024. Resiliency graphs: modelling the interplay between cyber attacks and system failures through ai planning. In *2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*, 292–302. doi:10.1109/TPS-ISA62245.2024.00041.
- [4] Marcelo Chamana, Adam Hahn, et al. 2023. An integrated testbed for power system cyber-physical security. *Applied Sciences*, 13, 16.
- [5] Yung-Chang Chang, Li-Ren Huang, Hsing-Chuang Liu, Chih-Jen Yang, and Ching-Te Chiu. 2014. Assessing Automotive Functional Safety Microprocessor with ISO 26262 Hardware Requirements. In *Technical papers of 2014 international symposium on VLSI design, automation and test*. IEEE, 1–4.
- [6] Abhimanyu Dubey et al. 2024. The llama 3 herd of models. *arXiv e-prints*, arXiv-2407.
- [7] David Formby, Milad Rad, and Raheem Beyah. 2018. Lowering the barriers to industrial control system security with {grfics}. In *2018 USENIX Workshop on Advances in Security Education (ASE 18)*.
- [8] Stefano Guarino et al. 2021. A hardware-in-the-loop water distribution testbed dataset for cyber-physical security testing. *IEEE Access*.
- [9] S. Hosseinzadeh et al. 2024. Design and development considerations of a cyber-physical testbed for transportation critical infrastructure. *Sensors*.
- [10] Seungmin Kim, Gyunyoung Heo, Enrico Zio, Jinsoo Shin, and Jae-gu Song. 2020. Cyber attack taxonomy for digital environment in nuclear power plants. *Nuclear Engineering and Technology*, 52, 5, 995–1001.
- [11] Maria B Line, Inger Anne Tøndel, and Martin G Jaatun. 2011. Cyber security challenges in smart grids. In *2011 2nd IEEE PES international conference and exhibition on innovative smart grid technologies*. IEEE, 1–8.
- [12] Aditya Mathur and Nils Ole Tippenhauer. 2016. Swat: a water treatment testbed for research and training on ics security. In *Proceedings of the International Workshop on Cyber-Physical Systems for Smart Water Networks*.
- [13] Ibukun A Oyewumi, Ananth A Jillepalli, Philip Richardson, Mohammad Ashrafuzzaman, Brian K Johnson, Yacine Chakhchoukh, Michael A Haney, Frederick T Sheldon, and Daniel Conte de Leon. 2019. Isaac: the idaho cps smart grid cybersecurity testbed. In *2019 IEEE Texas Power and Energy Conference (TPEC)*. IEEE, 1–6.
- [14] Paul Pfister, David Nicol, et al. 2019. Extending the isea testbed for cyber-physical security experiments. In *USENIX Workshop on Cyber Security Experimentation and Test (CSET)*.
- [15] Rakesh Podder, Turgay Caglar, Shadaab Kawnain Bashir, Sarath Sreedharan, Indrajit Ray, and Indrakshi Ray. 2025. Spear: security posture evaluation using ai planner-reasoning on attack-connectivity hypergraphs. In *Proceedings of the 30th ACM Symposium on Access Control Models and Technologies*, 62–73.
- [16] Indrajit Ray, Sarath Sreedharan, Rakesh Podder, Shadaab Kawnain Bashir, and Indrakshi Ray. 2023. Explainable ai for prioritizing and deploying defenses for cyber-physical system resiliency. In *2023 5th IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 184–192.
- [17] Abhijeet Sahu, Patrick Wlazlo, Zeyu Mao, Hao Huang, Ana Goulart, Katherine Davis, and Saman Zonouz. 2021. Design and evaluation of a cyber-physical testbed for improving attack resilience of power systems. *IET Cyber-Physical Systems: Theory & Applications*, 6, 4, 208–227.
- [18] Jianhua Shi, Jiafu Wan, Hehua Yan, and Hui Suo. 2011. A survey of cyber-physical systems. In *2011 international conference on wireless communications and signal processing (WCSP)*. IEEE, 1–6.
- [19] Christos Siaterlis, Andres Perez Garcia, and Béla Genge. 2012. On the use of emulab testbeds for scientifically rigorous experiments. *IEEE Communications Surveys & Tutorials*, 15, 2, 929–942.
- [20] Christos Siaterlis, Bela Genge, and Marc Hohenadel. 2013. Epic: a testbed for scientifically rigorous cyber-physical security experimentation. *IEEE Transactions on Emerging Topics in Computing*, 1, 2, 319–330.
- [21] Siddharth Sridhar, Adam Hahn, and Manimaran Govindarasu. 2011. Cyber-physical system security for the electric power grid. *Proceedings of the IEEE*, 100, 1, 210–224.