ICSNet: A Hybrid-Interaction Honeynet for Industrial Control Systems

Luis Salazar
University of California, Santa Cruz
Santa Cruz, CA, USA
luedsala@ucsc.edu

Efrén López-Morales
Texas A&M University, Corpus Christi
Corpus Christi, TX, USA
elopezmorales@islander.tamucc.edu

Juan Lozano
University of California, Santa Cruz
Santa Cruz, CA, USA
juclozan@ucsc.edu

Carlos Rubio-Medrano
Texas A&M University, Corpus Christi
Corpus Christi, TX, USA
carlos.rubiomedrano@tamucc.edu

Álvaro A. Cárdenas
University of California, Santa Cruz
Santa Cruz, CA, USA
alvaro.cardenas@ucsc.edu

Abstract

Industrial Control Systems (ICS) manage several critical infrastructures such as the electrical grid and water treatment plants. ICS have been the target of cyberattacks designed to disrupt the operation of critical infrastructure, risking the safety of the system. Honeypots and honeynets are used to gather intelligence on novel threats against ICS and to help us prepare for future attacks. In this paper, we introduce ICSNet, a *hybrid-interaction* honeynet that improves on the state of the art of ICS honeynets by developing a new modular architecture that integrates high-fidelity physical process simulations, more industrial protocols, and high-fidelity device fingerprints. We evaluate ICSNet using multiple physical process scenarios and reconnaissance tools like Nmap and Nikto. We show that ICSNet can successfully represent different ICS environments while interacting with the industrial assets in the physical simulation, giving attackers a convincing view of an ICS.

CCS Concepts

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Security and privacy** → **Domain-specific security and privacy architectures**; • **Applied computing** → *Industry and manufacturing*.

Keywords

Cyber-Physical Systems, Industrial Control Systems, Cyber Deception, Honeypot, PLC

ACM Reference Format:
Luis Salazar, Efrén López-Morales, Juan Lozano, Carlos Rubio-Medrano, and Álvaro A. Cárdenas. 2024. ICSNet: A Hybrid-Interaction Honeynet for Industrial Control Systems. In *Proceedings of the Sixth Workshop on CPS&IoT Security and Privacy (CPSIoTSec'24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3690134.3694813

CPSIoTSec'24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-1244-9/24/10
https://doi.org/10.1145/3690134.3694813

1 Introduction

Industrial Control Systems (ICSs) consist of a combination of components (e.g., electrical, mechanical) that work together to achieve an industrial objective, such as manufacturing, energy management, water treatment, etc. [37]. ICSs underpin many Critical Infrastructures (CI) so vital to society that their incapacitation or destruction would have considerable economic and health consequences [4].

In recent years, cyberattacks against ICSs have increased. For instance, in 2022, a new version of the Industroyer malware was discovered. Its purpose was to attempt, yet again, to target Ukraine's power grid by attacking circuit breakers automatically [33].

In order to gather information about novel threats to ICS, honeypots can be used to identify new attack patterns to help us prepare better protections. Early ICS honeypots, such as the SCADA HoneyNet Project [1], were initial efforts to provide low-interaction simulations limited to one or two network protocols commonly used in Industrial Control. Newer ICS honeypots such as HoneyPLC [19] extended previous work to provide a high-interaction environment and support for more ICS protocols and devices.

The design of general-purpose ICS honeynets has many challenges, including the diversity of vendors, industrial protocols, physical processes, control devices, and functionalities. Previous work has faced challenges when trying to replicate the diversity of these systems while ensuring a high-fidelity environment. In this paper we introduce ICSNet, a *hybrid-interaction* honeynet that improves on the state of the art of ICS honeynets by developing a new modular architecture that integrates high-fidelity physical process simulations, more industrial protocols, and high-fidelity device fingerprints.

Our contributions are as follows:

(1) We design *ICSNet*, the first *hybrid-interaction* honeynet, which solves many of the limitations of related approaches.
(2) We integrate our implementation with *Factory IO*, a high-fidelity physical process simulation that allows ICSNet to interact with the industrial assets, which provides the most sophisticated physical simulation in any honeypot in the literature.
(3) Our implementation supports more ICS network protocols, ICS manufacturers, ICS devices, and physical processes than

previous work. In addition, we make our implementation open source to help increase the fidelity of future ICS honeynets. ICSNet is available online[1].

(4) We provide experimental evidence that shows ICSNet successfully interacts and deceives well-known tools for network reconnaissance.

2 Background

ICS is a term that encompasses multiple types of control systems, including supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and other control system devices such as Programmable Logic Controllers (PLC) [36]. ICS networks support the infrastructure of many of the basic services that we rely upon. For instance, energy transportation, communications, and clean water. In the United States, Presidential Policy Directive 21 defines sixteen critical infrastructure sectors, including communications, manufacturing, emergency services, and energy, among others [11].

2.1 Honeypots and Honeynets

Honeypots. A honeypot is a computer system that aims to be probed, attacked, or compromised. The value of a honeypot lies in the data it obtains from attackers. These data can be later analyzed to detect existing and new adversaries' techniques [29].

Honeynets. A honeynet is a network of at least two interconnected honeypots. This architecture provides a controlled network in which adversaries' interactions are monitored and recorded.

Low-Interaction. Low-interaction honeypots provide the least amount of functionality to adversaries. They are usually implemented via simple scripts. They have two main advantages. Low-interaction honeypots are more accessible to develop and maintain, and due to their limited functionality, they pose a low risk of an adversary taking over them. Their main drawback is that adversaries may not complete their attack and realize they are interacting with a decoy.

High-Interaction High-interaction honeypots provide interactions equal to or close to the same level as the natural system [7]. They are implemented using real devices or via software emulation, e.g., virtualization. In the context of ICS, high-interaction honeypots can simulate attacks in a more realistic setting than low-interaction honeypots thanks to a higher fidelity of the physical process simulation. High-interaction honeypots have two main disadvantages. First, they are very costly to deploy and maintain. Second, they pose a high risk of adversary takeover. High-interaction honeypots may provide too much freedom, which an adversary could exploit to take over the honeypot, lock out the honeypot administrator, and use the honeypot to establish a foothold in the network [43]. Even though high-interaction honeypots provide the best interaction capabilities, they should not always be used if they pose a high risk of an adversary takeover.

Hybrid-Interaction Hybrid-interaction honeynets [13, 14] combine low and high-interaction honeypots to take advantage of their combined benefits, thus getting the best of both worlds. One of the advantages of hybrid-interaction honeynets is their flexibility. They allow multiple types of honeypots to be integrated with the same honeynet. In essence, they can be *mixed 'n matched* to satisfy the honeynet requirements. Additionally, hybrid-interaction honeynets allow the honeynet administrator to control the amount of adversary takeover risk they are willing to take. If they want to reduce the takeover risk, they can replace high-interaction honeypots with low-interaction ones. RIoTPot is a hybrid-interaction honeypot capable of switching between low and high-interaction modes [35]. To the best of our knowledge, ICSNet is the first hybrid-interaction honeynet in the literature.

2.2 Evolution of ICS Honeypots

We now state and discuss the stages of evolution of ICS honeypots that we identified during our literature review.

1. Basic ICS Protocol Simulations.

The first generation of ICS honeypots is characterized by limited simulation of ICS devices and protocols. For example, the SCADA Honeynet Project [1], developed by Cisco Inc., is one of the very first ICS honeypots. It simulates a basic PLC with a Modbus TCP server with hardcoded registers. Future approaches such as S7commTrace [44], Conpot [41], and the Digital Bond's Honeynet [42] expanded the number of ICS devices and protocols, however, keeping with the basic simulation design. For example, Conpot allows users to customize their honeypots by using templates and S7commTrace provides basic simulation of the S7comm Siemens proprietary protocol. These ICS honeypots do not have any physical processes simulation capabilities.

2. Basic ICS Protocol and Physical Simulations

The second generation of ICS honeypots is characterized by the incorporation of basic physical simulation features. The work by Antonioli et al. [7] is one of the first examples of this trend. Their approach simulates the Ethernet/IP ICS protocol along with multiple PLCs and HMIs. Most importantly, they integrate MiniCPS with their honeypot which allows it to simulaute a water treatment physical process where each water tank has an inflow pipe, and an outflow pipe, that simulate hydraulics equations. HoneyPhy [18] is another example which simulates a thermostat physical process.

3. Adavanced ICS Protocol Simulations

The third generation is characterized by the improvement of the simulated ICS devices and network protocols, which opened the door for previously unavailable interaction opportunities. HoneyPLC [19] introduced an advanced simulation of the S7comm protocol that not only responds to basic network queries but can interact with advanced tools such as PLCScan, Nmap and the real Step7 Manager used to configure real PLCs. Additonal examples include the work by Grigoriou eta. [15] and HoneyVP [45] which also include advances simulations of other ICS protocols such as IEC-104.

Eventhough this generation saw the development of more advanced ICS device and protocol simulations, the advancement of the physical process simulations remained largely stagnant.

4. Advanced ICS Simulations with Basic Physical Processes Simulation. This current generation of ICS honeypots is characterized by advanced simulations of ICS devices and network protocols and basic simulations of physical processes as seen in Gen 2. For example, ICSpot [10] builds on top of HoneyPLC[19] and adds a physical process simulation based on the SWaT water treatment

[1] https://anonymous.4open.science/r/ics-virtual-testbed-766D

Table 1: Comparison of Existing Honeypots/Honeynets and ICSNet.

	[1]	[44]	[42]	[41]	[35]	[10]	[19]	[20]	**ICSNet**
Supported ICS devices	1	1	1	2	⊠	2	5	7	12
Interaction level	L	H	H	L	Y	H	H	H	Y
Network protocols	3	1	1	3	4	4	3	2	5
Physical process simulation	○	○	○	○	○	●	○	●	●
Modularity / Extensibility	○	○	○	●	●	●	●	●	●
Honeynet	●	○	●	○	○	○	○	●	●
Supported manufacturers	1	1	1	2	⊠	3	3	3	6

○: Not supported ●: Supported ⊠: Not specified L: Low H: High Y: Hybrid

process. This simulation leverages Industrial Hacking Simulator (IHS) is a repository with ICS scenarios which is based on MiniCPS. Another example is HoneyICS[20], which also builds on top of HoneyPLC and integrates multiple functionalities including a simplified version of the Secure Water Treatment system (SWaT) implemented using Simulink.

We aim to push the state of the art by introducing ICSNet and a new generation of ICS honeypots that include both *advanced ICS device and protocol simulations* and *advanced physical processes simulations*.

Despite the benefits of the previously discussed honeypots, the present solutions fail to provide the necessary features to simulate ICS networks so that the latest and most sophisticated attacks can be understood. Specifically, their limitations are:

Limited ICS Device Support. Existing approaches provide shallow PLC simulations that do not account for the difference between PLC manufacturers and models. This limitation stops current approaches from extracting data from adversarial interactions and malware that varies depending on the PLC. MiniCPS, for example, simulates PLCs using Python scripts [8]. ICSNet addresses this limitation by providing support for 10 PLC models implemented using a high-fidelity fingerprint of ICS devices.

Limited ICS Network Protocol Support. Current approaches provide limited support for ICS network protocols. This limitation prevents other virtual testbeds from simulating different types of ICS networks. Most testbeds support only one protocol, namely, Modbus TCP.

Limited Interaction Support. Current honeynets do not use the hybrid-interaction model to provide low and high-interaction simulations. This limitation prevents other honeynets from simulating different types of ICS networks.

Limited Physical World Simulation Support. Most of the current solutions do not include support or consideration of the ICS physical simulation. This limitation hinders the ability of any ICS honeypot or honeynet to provide responsive and believable interactions to an attacker. As discussed in Sec.3.7, ICSNet addresses this limitation by introducing a high-fidelity physical process simulation that can simulate multiple physical processes, not just one.

We summarize these limitations in Table 1. The table shows that our solution supports more devices, vendors, protocols, and potentially more physical process simulations than other work.

3 ICSNet's Design

3.1 Threat Model

1) We assume that the attacker already has access to our honeynet. This could be achieved by scanning for ICS network protocol ports over the Internet or using a search engine such as Shodan[25]. 2) After selecting their target, the attacker starts a reconnaissance using port scan tools such as Nmap. The attacker's goal is to find and enumerate the number of hosts in the network (honeynet) and what are the exposed services, e.g., Modbus TCP port 502. 3) After mapping out the honeynet and finding target services, the attacker probes each service to obtain more information. For example, they could use Nmap's Modbus discovery scripts mentioned in Sec.5 to get device information, such as the device model and firmware version. 4) Finally, the attacker can send a malicious Modbus command to the PLC to change one of the registers. This would damage the industrial control system operation.

We design ICSNet to log all the attacker's interactions with the honeypots in the honeynet so that they can be later analyzed.

Based on the current state and limitations of existing solutions, we identified the following primary characteristics guiding the design of ICSNet: (1) Modular Architecture (Sec.3.2), (2) Hybrid-Interaction Architecture (Sec.3.3), (3) High fidelity ICS Device Profiles (Sec.3.4), (4) Support of representative ICS protocols (Sec.3.6), (5) High fidelity physical process simulations (Sec.3.7).

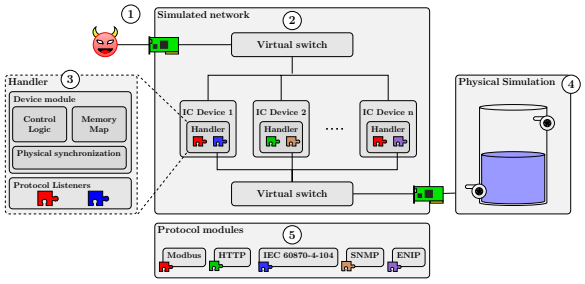
3.2 Modular Architecture

We leveraged and integrated two existing frameworks to implement the modular part of ICSNet: Honeyd and Mininet.

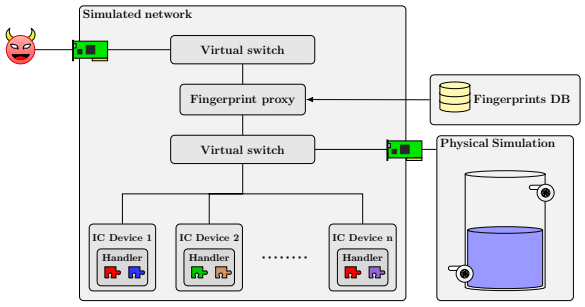
Honeyd. Honeyd is a virtual honeypot framework that simulates [29] virtual computer systems at the network level. Honeyd simulates the TCP/IP stack of different operating systems and devices to deceive network fingerprinting tools. It leverages a fingerprint database to store and match hundreds of devices, e.g., PLCs. Honeyd can be extended to include protocol simulations using scripts. For example, it can simulate a telnet server using a Python script. Additionally, Honeyd can be configured as a proxy to route a particular port to a different host. For example, it can send HTTP traffic to a web server. Honeyd is classified as a low-interaction honeypot as the simulations are limited to scripts and do not provide advanced simulations. We selected Honeyd because of two reasons. First, it is highly customizable, allowing new fingerprints for ICS devices. Second, its proxy capabilities allow low and high-interaction honeypot implementations to be routed to the correct TCP and UDP ports.

Mininet. Mininet [27] creates realistic virtual networks by taking advantage of Linux's kernel namespaces, similar to some container solutions. By leveraging this feature, it can simulate several hosts with "independent" kernels communicating with each other via virtual switches. Mininet uses the facilities provided by Open vSwitch [28] to create and manage the different virtual switches and any corresponding association with the virtual interfaces handled by each kernel namespace. Furthermore, since Mininet is ultimately a Software-Defined Network (SDN), it also supports adding an external SDN controller to handle and manage the network traffic.

The primary design goal of our solution, shown in Figure 1a, is to present the attacker with an entire virtual network through a single



(a) ICSNet's architecture



(b) simulation with device profiles

Figure 1: Extensible hybrid-interaction architecture.

network interface ①. We make no assumptions about the nature of the communications channel between this network interface and the attacker. In this network, the attacker must be able to interact with any simulation nodes they can reach, and this interaction must be convincing enough that the attacker believes it to be a legitimate industrial control network.

To that end, we attach the network interface of a simulation host, which the attacker can access, to a virtual switch ②. This interconnection allows the attacker to interact with any simulated hosts attached to this virtual switch. Each of these simulated hosts is, in the sense of the simulation, a node that carries out a specific role in the simulation, be it a PLC, an HMI, a node simulating a physical process, or any other relevant node in the simulated scenario.

The overall design principle of our solution is focused on flexibility; each simulated node instantiates a device handler, managing the device module and the industrial protocol listeners ③. The device module provides the specific control logic for the scenario, a memory mapping resembling an actual device, and a synchronization mechanism used to interact with the physical process simulation ④. The handler bridges this device simulation with the industrial control protocol modules, allowing any actor within the same network to interact with the simulated device via the appropriate protocols ⑤.

Our solution enables the simulation of various industrial control scenarios. Each scenario uses a configuration file specifying the network topology, device information, and appropriate modules. For each device, it is necessary to define 1) network links, 2) the virtual device information (e.g., manufacturer, model, device name), 3) the industrial control module it will instantiate, and 4) the industrial control protocols it will support. The downside of this scheme is that if a malicious actor performs any reconnaissance activities over the simulated network, they would still recognize the simulated devices as the host operating system.

3.3 Hybrid-Interaction Architecture

ICSNet's modular design also allows us to mix low and high-interaction honeypots in our honeynet and achieve a hybrid-interaction architecture. We decided to have a hybrid-interaction architecture because it allows ICSNet to incorporate multiple honeypots without the need to limit ICSNet to be either low or high-interaction. As discussed in Sec. 2.2, this freedom allows us to reduce the risk of adversarial takeover when a high-interaction implementation is not required. For example, a hybrid-interaction architecture is ideal for ICS networks. This is because ICS networks are made of multiple heterogeneous devices and protocols. Developing high-interaction honeypots for all ICS devices in an ICS honeynet might be impossible.

3.4 Personality Engine

To fool an attacker and overcome Nmap's OS detection discussed in Section 5, we must first create a suitable network profile for any device we intend to simulate. There are two primary aspects to these device profiles: the open ports with their corresponding protocols, e.g., HTTP's TCP port 80, and the overall network behavior of the device, e.g., TCP options. We introduce a fingerprint proxy (Figure 1b) to present these aspects of the simulated devices to the attacker.

This proxy mimics the network stack behavior of a specified device as long as its fingerprint is in the database and acts as a barrier between the attacker and the simulated devices. We also configure the proxy to redirect any incoming connections to specific industrial control ports toward the appropriate emulated device, providing a reasonable interaction with the simulated physical process.

To generate the fingerprints, we use Nmap, which offers a scan option that enables a fingerprinting process for any given host. A host scan with NMap reveals which ports a particular device opens for any communication with the SCADA servers, most of which are well-known and standardized (e.g., Modbus' TCP port 502 or IEC104's TCP port 2404).

The second aspect, however, has a caveat regarding how Nmap performs its detection. Nmap uses several techniques to measure specific TCP/IP parameters and behaviors to determine the overall network fingerprint of a given device. NMap then compares these results with its internal database to estimate the operating system running in the targeted device.

We performed several Nmap scans on the same device and noticed that some parameters within the fingerprint vary between scans, specifically what Nmap calls the "SP" and the "ISR" values [5]. Fingerprint 1 depicts an example with these single values.

After several scans, we noticed that these values, while not fixed, were always within a specific range for a given device. To methodically acquire the proper fingerprints, we used *python-nmap* [2] to write a script to systematically scan a given target and determine the range for these varying values. After a reasonable number of scans (in our case, we noticed that after 20 scans, the results were

Fingerprint (1) Sample fingerprint with single values.

```
SEQ(SP=9A%GCD=1%ISR=9F%TI=I%CI=I%TS=1)
OPS(O1=M5B4NW0NNT11%O2=M5B4NW0NNT11%O3=M5B4NW0NNT11%
O4=M5B4NW0NNT11%O5=M5B4NW0NNT11%O6=M5B4NNT11)
WIN(W1=2000%W2=2000%W3=2000%W4=2000%W5=2000%W6=2000)
ECN(R=Y%DF=Y%T=40%W=2000%O=M5B4NW0%CC=N%Q=)
T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=N)
T4(R=Y%DF=N%T=40%W=2000%S=A%A=Z%F=R%O=%RD=0%Q=)
T5(R=Y%DF=N%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=N%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T7(R=Y%DF=N%T=40%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
U1(R=Y%DF=N%T=40%IPL=38%UN=0%RIPL=G%RID=G%RIPCK=Z%RU
CK=0%RUD=G)
IE(R=N)
```

Fingerprint (2) Sample fingerprint with relevant ranges.

```
SEQ(SP=98-9C%GCD=1%ISR=9F-A1%TI=I%CI=I%TS=1)
OPS(O1=M5B4NW0NNT11%O2=M5B4NW0NNT11%O3=M5B4NW0NNT11%
O4=M5B4NW0NNT11%O5=M5B4NW0NNT11%O6=M5B4NNT11)
WIN(W1=2000%W2=2000%W3=2000%W4=2000%W5=2000%W6=2000)
ECN(R=Y%DF=Y%T=40%W=2000%O=M5B4NW0%CC=N%Q=)
T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=N)
T4(R=Y%DF=N%T=40%W=2000%S=A%A=Z%F=R%O=%RD=0%Q=)
T5(R=Y%DF=N%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=N%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T7(R=Y%DF=N%T=40%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
U1(R=Y%DF=N%T=40%IPL=38%UN=0%RIPL=G%RID=G%RIPCK=Z%RU
CK=0%RUD=G)
IE(R=N)
```

always the same), we determined the appropriate fingerprint for each device we needed to simulate. By using ranges as the values for these parameters (Fingerprint 2), the fingerprints capture the overall behavior of the device, thus providing a more reliable fingerprint for it.

Aside from the industrial control protocols, some devices offer a web interface to present diagnostics information or an application with some configuration options. In the case of PLCs, these web interfaces usually show information about the overall device, such as the network interface information, system uptime, device model, version, and firmware revision. In this sense, the web interface is another way to acquire the same information available via other industrial control protocols such as Modbus. In other cases, the manufacturer allows the administrator to upload custom web pages that may interact with the inputs and outputs of the device, offering the possibility of creating crude HMI web pages displaying the current state of the physical system. In these instances, the device presents the administrator with a web application in which a login is necessary to make any changes. These functionalities usually depend on the licensing scheme purchased with the device.

To emulate the internal web server of the devices that provide this capability, e.g., PLCs, we need to offer a similar functionality within our simulated device. We first mirror the devices' websites into static files using *wget* [31]. We then implement a simple HTTP server with configurable headers that serve the static files to any incoming connection. Figure 3 shows an example front page of one of the devices offered by this simulated web server.

3.5 Devices Supported in ICSNet

Besides the granularity addressed by a device fingerprint and its Web Page extraction, we went further and included a variety of models from three vendors; also, to add more realistic features, we explored an industrial networking switch and a security appliance. The total number of ICS device models is twelve, listed below:

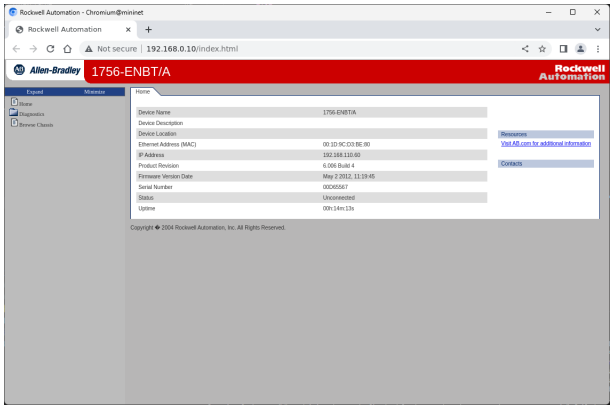


Figure 3: Home page of emulated PLC.

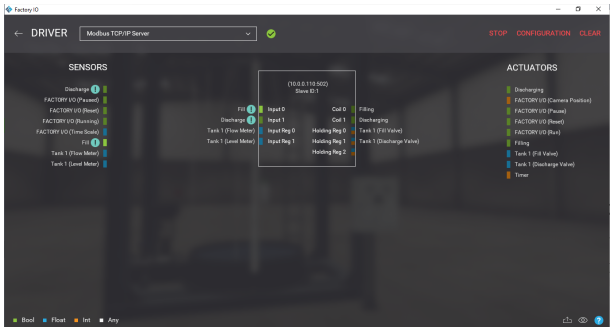


Figure 4: Factory I/O Modbus configuration.

- **Siemens Simatic ET 200SP**: This model, in Figure 5a, comes from a line of devices that include distributed CPU, interface modules, and signal models, all of which can support dynamic industrial settings and include multiple protocols.
- **Siemens ET 200s**: This device, shown on Figure 5b, is a solution for a distributed I/O system. The vendor optimized it for a ProfiNet Bus on a Decentralized Periphery Master-Slave architecture.
- **Siemens Simatic S7-1200**: These devices are listed as low to medium-power applications and are part of a line that includes central processing, interface modules, and signal modules. Figure 5c shows this model.
- **Siemens Simatic S7-1500**: This is a high-end line of devices. Figure 5d shows that it has a built-in display, and the vendor also highlights safety and security integration and enhanced processing.
- **Siemens S7-300**: This device was available officially until October 2023, although support will continue for ten more years. With newer models, its use is reserved for low-power activities. Figure 5e is a photo of this device.
- **Allen-Bradley 1756 Enbt/A**: This Allen-Bradley module, in Figure 5j it is integrated to a chassis. This module supports EtherNet/IP communication.
- **Allen-Bradley Micrologix 1400**: This Allen-Bradley controller, pictured in Fig.5g, is a compact device with the option to add expansion modules.

- **National Instruments cRIO-9024**: The CompactRIO controller has two ethernet ports, USB, and serial. It has an 800 MHz CPU, 512 MB RAM, and a 4 GB storage unit. Displayed on Figure 5k
- **National Instruments cRIO-9068**: This device is an eight-slot cRIO controller with a 667 MHz CPU, 512 MB RAM, and 1 GB storage unit. Displayed on Figure 5l
- **ABB PM554-TP-ETH**: This PLC has eight digital inputs, six digital outputs, Ethernet, and has the option of plugging input/output terminal modules. We show this device in Figure 5f.
- **Moxa EDS-405A Switch**: Designed for industrial applications, it allows useful management functions, such as ring coupling, port mirroring, and virtual local networks. It has Ethernet I/P and Profinet standard implementations. This device has a web interface for management. It is displayed at the right of Figure 5h.
- **Tofino Mguard RS4004**: A security Router, its characteristics include 10/100 Mbps, VPN, firewall, and four-port managed switch, with DMZ and the possibility to be extensible. Figure 5h at the left shows this device.

3.6 Support of Representative ICS protocols

We must implement multiple network protocols commonly found in ICS to support the protocol modules defined in Fig. 1a.

We considered two criteria to identify what ICS protocols should be supported in our honeynet. First, we considered the protocols for the physical ICS devices procured for this research. Second, we considered ICS protocols based on their popularity.

To identify the protocols running on a particular device, we used Nmap's port scan. For example, to scan Allen -Bradley's MicroLogix 1400 PLC shown in Fig. 5g, we used the command *nmap* -p 1-10000, 45000-55000 -oN micrologix1400 192.168.103.15*. The output indicated that this particular PLC was running three services. A web server on TCP port 80, Modbus on TCP port 502, and EtherNet/IP on TCP port 2222. We scanned all the devices and supported the following five ICS network protocols: Modbus TCP, HTTP, IEC-104, SNMP, and ENIP. We left out some ICS protocols, such as DNP3, because they are not present in any of the ICS devices included in this work, and our goal is to match devices with the associated protocols they support.

Different ICS devices make use of a combination of the above protocols. For example, a PLC might support Modbus TCP's communication with other PLCs. It might support HTML to expose a web server that shows diagnostics. It might support SNMP in exposing an SNMP agent that shares information such as temperature. On the other hand, ICS devices, such as industrial routers, only support HTTP to expose a web configuration interface. Supporting these protocols improves ICSNet's fidelity when an attacker interacts with it. To simulate IEC-104, we use the work by *Salazar et al.* [34]. We used their work because it is open-source and publicly available, which allowed us to integrate it into ICSNet. To simulate Modbus TCP, we implemented our simulation using Scapy [32]. Although some Modbus implementations exist such as *pymodbus* [16], we wrote our custom implementation because it was easier to integrate into our honeynet. To simulate ENIP, we leveraged CPPPO [17], which is publicly available and open-source. We use snmpsim [12], also publicly available and open-source, to simulate SNMP.

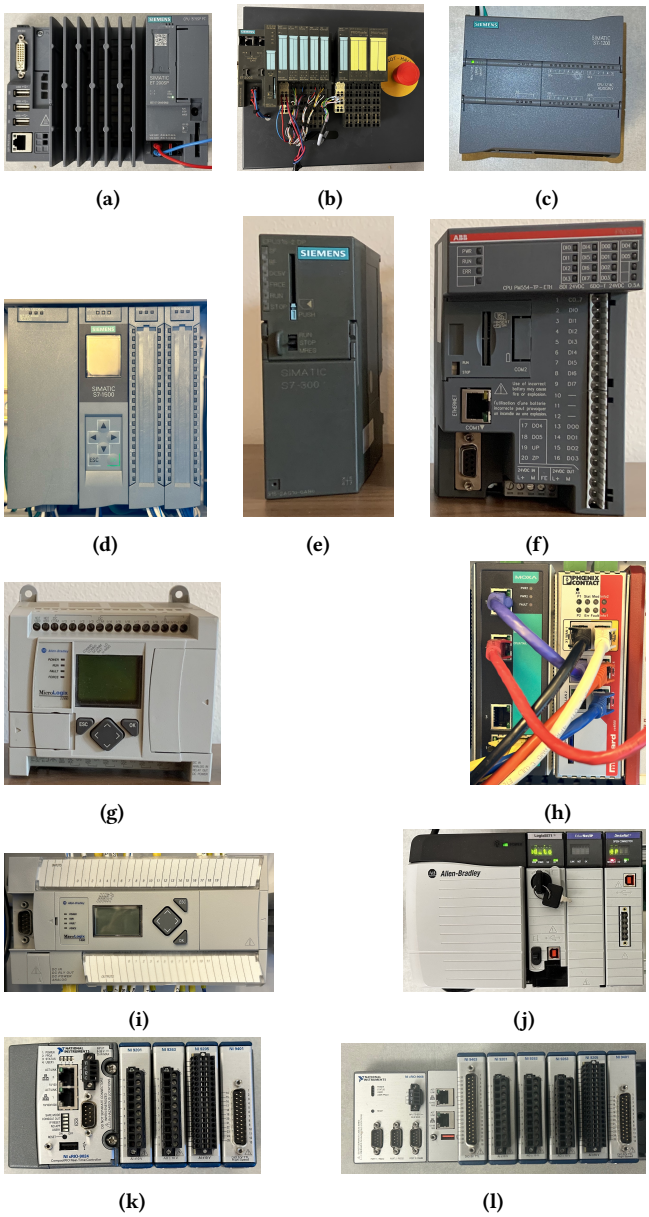


Figure 5: (5a): Siemens ET 200. (5b): Siemens ET 200s. (5c): Siemens S7-1200. (5g): Allen-Bradley Micrologix 1100. (5d): Siemens S7-1500. (5e): Siemens S7-300. (5f): ABB PM554-TP-ETH. (5h): Moxa EDS-405A Switch (left), Tofino Mguard RS4004 (right). (5i): Allen-Bradley MicroLogix 1400. (5j): Allen-Bradley ENBT. (5k): National Instruments cRIO-9024. (5l): National Instruments cRIO-9068.

3.7 High Fidelity Physical Process Simulations

For the physical process simulation, we support two alternative mechanisms. The first is to include a virtual device for this purpose and configure the virtual network topology so that this node is isolated from any possible interaction from the attacker. The second one relies upon an external simulator and isolates a secondary network interface in the host in place of the simulation node, a network interface with access to the external simulator.

Since we already added a fingerprint proxy, we can either add a physical simulation node to the network or link a secondary network interface to the internal switch, to which the attacker has no direct access.

As an external simulator we considered PLC trainers. These systems provide pre-built scenarios that simulate the operation of a PLC alongside a particular control logic program. This way, field engineers can learn how to use control systems utilized in the industry with just a PC, the PLC control logic, and an optional input/output interface [3].

For instance, Factory I/O is a Next Generation PLC trainer and provides industrial scenarios and customization capabilities [6]. EasyPLC is another tool that allows PLC interaction on a 3D custom environment [3]. Both 3D PLC trainers rely upon additional software to establish the ladder logic to be implemented; those tools can be open access and easily integrated.

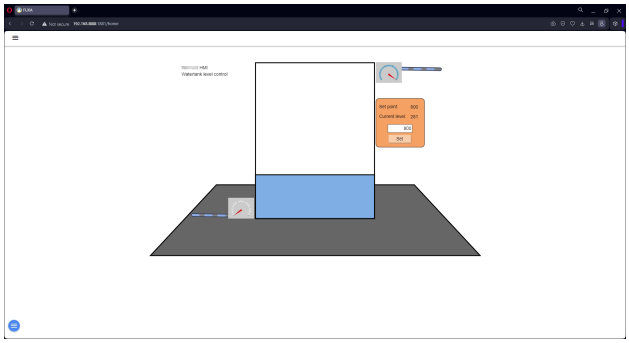
We found that Factory I/O provides more extensive documentation than EasyPLC. We also could use Factory IO before committing to a license purchase with full features during a one-month trial. For those reasons, we decided to go further and use it as our physical environment simulation software. Available industrial scenarios relevant to this research include a water tank with sensors and actuators to control liquid levels, conveyors with box detection sensors, and an automated warehouse.

Factory I/O simulates a physical process configured in what they define as a "scene", and provides access to virtual sensors and actuators that affect the simulation. To interface with this simulator, we used its built-in Modbus support to acquire or set the sensor or actuator values (Figure 4).

For the control logic of the different scenarios, we implemented a configurable simulated PLC that interacts with Factory I/O via Modbus. This simulated device constantly reads the current status of the simulation and executes the appropriate control logic consistent with the physics of the simulated system. The local state within the device resides in a simulated memory mapping consistent with what would be suitable for standard industrial control protocols: four 64K addressable memory blocks comprised of pairs of boolean and 16-bit data values organized in read-only and read-write blocks. That is 64K read-only boolean values, 64K read-write boolean values, 64K read-only 16-bit data, and 64K read-write 16-bit data.

We designed this memory mapping to use standard industrial control protocols seamlessly. For instance, if the PLC we want to simulate uses Modbus, then the memory mapping would align with Modbus' Discrete Inputs (boolean read-only), Coils (boolean read-write), Input Registers (16-bit read-only), and Holding Registers (16-bit read-write). The exact mapping is also compatible with, for instance, IEC-104. The difference is that instead of memory addresses, IEC-104 uses Information Object Addresses (IOA), and

(a) Factory I/O



(b) FUXA HMI

Figure 6: Water tank physical process and HMI.

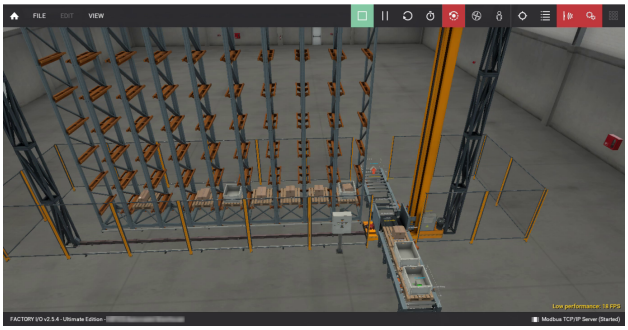
the device determines the specific memory mapping it needs to access by the given IEC-104 message and IOA. For instance, IEC-104's "Single/Double command" would access boolean read-write memory regions, whereas "Single-point information" would access either the read-only or read-write boolean mappings.

Aside from the physical simulation, we add another component to the network in the shape of an HMI for each simulated scenario. To deploy these HMIs, we use FUXA, an open-source HMI and SCADA dashboard. This HMI communicates with our simulated devices identically to how it would communicate with actual industrial control devices. With the HMI's added functionality, each scenario provides a complete overview of the simulated process.

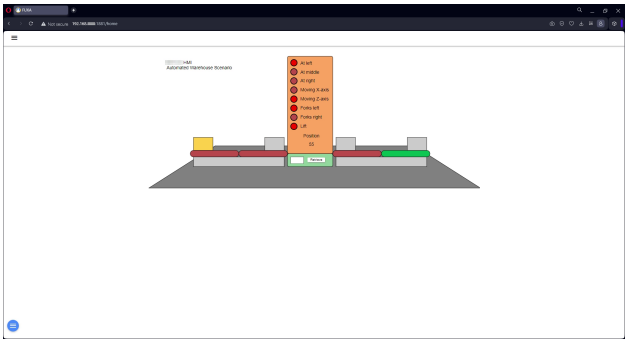
We first tested our simulation with a simple water tank as a first physical process. Based on Factory I/O, this process provides a fixed-volume tank with a water input controlled by a valve and an output sink controlled by a valve. It also offers a single sensor providing the liquid's height within the tank (Figure 6a). The overall idea of the control process is to maintain a constant flow of water while keeping a specific height given by a set point.

Finally, we implemented a simple HMI using FUXA (Figure 6b). This HMI presents a web interface in which an operator can monitor the current state of the simulated PLC. It communicates with the simulated PLC using Modbus and allows the operator to change the process' set point to any desired value.

As a second scenario, we tested an automated warehouse process (Figure 7a). Unlike the water tank, a state machine defines the control logic for this process instead of a physical model. Moreover,



(a) Factory I/O



(b) FUXA HMI

Figure 7: Automated warehouse physical process and HMI.

the particularities of the scenario warrant the use of two control devices.

The overall functionality of the process is to receive packages in an input conveyor belt, handle them with a forklift crane, and store them in a warehouse space organized into a defined grid. As for the retrieval, an operator can input the numerical value corresponding to a warehouse space in the grid, and the control system should automatically operate the crane to retrieve the requested package and deliver it to an output conveyor belt.

The crane offers a set of sensors that indicate its movement status and the position of the forklift. Similarly, the conveyor belts offer sensors that detect packages in both the input and output positions. Thus, this scenario has two main functions: one is to operate the conveyor belts, and another is to operate the cranes. Each function is dependent on the current state of the components.

As long as the warehouse has available spots, the system will automatically receive new packages and store them inside the first available spot. If there are no available spaces in the warehouse, the system will idle until the operator requests a package and empties a spot. Any pending packages will remain in the input conveyor belt until a free spot is available.

We also implemented another HMI using FUXA (Figure 7b) that presents the system's current state and enables an operator to issue a retrieve command for any particular package spot stored within the warehouse.

Table 2: OS Detection comparison

Device	% OS detection Real	% OS detection ICSNet
Allen-Bradley enbt/a	100	40
Micrologix 1400	36	100
Mguard RS4004	100	100
MOXA EDS-405A	86	100
NI-Crio-9024	100	100
NI-Crio-9068	100	100
Siemens 200sp	10	80
Siemens S7-1500	100	100
Siemens S7-1200	100	100

4 Evaluation

4.1 Device Fingerprint evaluation

ICSNet's simulation architecture, depicted in Fig 1a, includes a TCP/IP stack simulation by leveraging Honeyd [29], discussed in Sec.3.2. To evaluate our TCP/IP stack simulation, we use Nmap, discussed in Sec. 5.

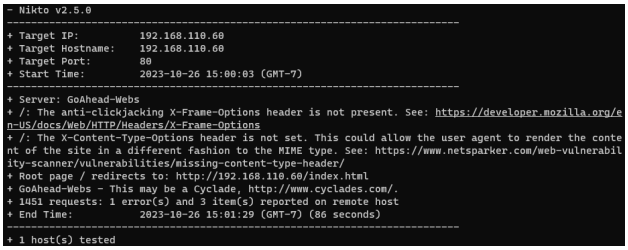
Experiment Environment. Our environment includes two hosts running Ubuntu 20.04. The first host runs ICSNet, and the second host runs Nmap. Both hosts are connected to the name network without any router in between.

Experiment Data. The experiment data consists of multiple Nmap OS detection scan reports for each ICS device listed in Fig. 5. We use these to determine if a well-known scanning tool, i.e., Nmap, can identify, with high confidence, ICS devices in cases where their fingerprint profile is not part of released repositories. We aim to extract high-quality fingerprints from the *real* ICS devices procured for this study. We then included these fingerprints in the Nmap fingerprint database and ran fifty scans for each ICS device and their counterpart of the ICSNet approach. Since we added information to the fingerprint database, we plan to submit that information to NMAP.

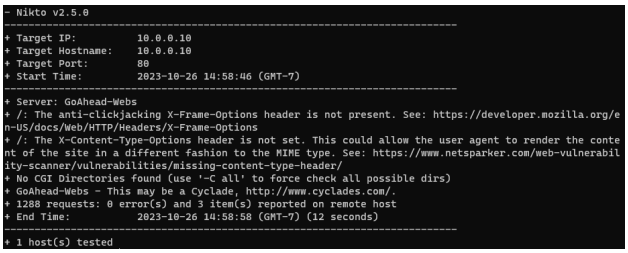
Each scan output consists of a predicted Operative System and a confidence percentage. Since there are fifty scans per device, we also used the frequency for each pair (OS, Confidence) for both the set of real devices and the devices from our ICSNet proposal. We want to evaluate the accuracy of standard scan tools to identify ICS devices given fingerprint extraction on our own; we used the frequency of the correct OS detection per device and normalized it.

Table 2 summarizes our results. OS detection performs well for the actual scanned devices, achieving high percentages in seven of the nine ICS equipment evaluated; furthermore, performance improves for our ICSNet, getting over 80% OS prediction for eight of the nine devices. Our approach identifies Industrial Devices as frequently as a person with access to the actual device would find for all the studied real scanned ICS devices, except Allen-Bradley enbt/a.

There are two real devices with low detection values: Siemens 200sp and Allen-Bradley Micrologix 1400; similarly, our simulated Allen-Bradley enbt/a scan identification results are below 50%. In a perfect world, scan fingerprinting would be a one-to-one function allowing exact host identification; nonetheless, in practice, there are multiple fingerprints mapping to a single device, or it could be a single fingerprint that maps to numerous OS. Our central



(a) Device



(b) Honeypot

Figure 8: Nikto web scan.

hypothesis for the low identification of Siemens 200sp, Micrologix 1400, and virtual Allen-Bradley enbt/a is that the values that we found in our extracted fingerprints are very close to the ones of other OS; we are in the error scenario where one fingerprint maps multiple devices, in practice Nmap bets high for the different devices given the clashing fingerprints; in fact, embedded systems are prone to have this type of error since third party vendors usually sell their system architecture and firmware to various companies.

4.2 Web Evaluation

Emulating a web page for any device is not limited to serving duplicate files using the same protocol. Websites have additional inherent information that complements the characterization of a particular device or application, such as the actual web server and any configured headers. Our simulation must provide matching files and match the appropriate headers to fully mimic the original device.

To test this, we veered off from using Nmap. We instead used Nikto [38], which we discussed in Sec.5. We chose this tool not only for its wide adoption but also due to its versatility and scan speed. Nikto can perform generic and specific server software checks to identify web technologies used and reveal any known vulnerabilities in a web application.

We ran Nikto against the original device (Figure 8a) and our simulated web server (Figure 8b) and compared the results. While the exact number of requests varies, which is to be expected as ICSNet cannot behave precisely as all the different web servers, the actual header identification and web server fingerprinting that Nikto executes concludes that our simulated device is running the "same" web server and discloses the same known vulnerabilities as the original devices.

The known vulnerabilities that the devices presented relate to misconfigured HTTP headers in the responses provided by the different devices. The vulnerabilities Nikto identified involve either missing or improper configurations of the "cross-origin resource sharing (CORS)" policies within the HTTP headers, which could allow attacks such as cross-site request forgery or clickjacking. Aside from this comparison, we also took note of the web server Nikto designates for each device and matched it to the web server detected in the simulation. We show these comparisons in Table 3.

Table 3: Nikto scan results

Device	Requests		Server match	Vulnerable headers
	real	simulated		
Allen-Bradley enbt/a	1451	1288	yes	2/2
Micrologix 1400	1435	1376	yes	2/2
Siemens S7-1500	1383	1245	yes	3/3
MOXA switch	1426	1335	yes	1/1
mGuard RS4004	1512	1368	yes	2/2

Table 4: ICS Protocol Evaluation Results

ICS Protocol	Implementation	Evaluation tool	Result
Modbus	ICSNet custom	nmap script	✓
IEC-104	NEFICS	nmap script	✓
ENIP	cpppo	nmap script	✓
SNMP	snmpsim	nmap script	✓
HTTP	Python HTTPServer	Nikto	✓

As for the difference in the number of requests shown in Table 3, even though we do not notice any evident discrepancies in the content presented by the device and the simulation, executing a thorough web scan with a tool such as Nikto can reveal additional resources or responses leading to further requests for any given test. Since these behaviors are not easily predictable and can vary between web server implementations, we expected to have a reasonable amount of discrepancies between the scans of the actual device and the simulation. Even though there are several discrepancies with the simulations, the overall discrepancies do not exceed 20% of the total number of requests between the two samples.

Furthermore, the overall result provided by Nikto reaches the same conclusion from the perspective of a potential attacker. Assuming the attacker does not have a specific profile of the devices s/he is attacking, which is a reasonable scenario, it is doubtful that s/he would flag the simulation as a fake service, given the overall similarities with the actual device.

4.3 ICS protocol evaluation

We use the publicly available Nmap scripts discussed in Sec. 3.6 to evaluate the supported ICS protocols discussed in Sec. 5. Nmap is a well-known reconnaissance tool, and if our honeynet is able to deceive Nmap, then it means that our honeynet meets the High-Fidelity design objective discussed in Sec. 3.4.

Experimental Environment. Our environment includes two hosts running Ubuntu 20.04 and Python 3.10 and the required dependencies, for example, mininet and scapy. The first host runs ICSNet, and the second host runs a reconnaissance tool, in this case, Nmap. Both hosts are connected to the name network without any router in between.

Experimental Methodology. We run ICSNet to simulate one honeynet containing one device for each ICS protocol to be tested. We then use Nmap's scripting engine to test whether or not our ICS protocol simulation can successfully interact with Nmap. Specifically, we use Nmap commands such as *nmap -p 2404 -v -v -v -v -n -Pn –script=iec-identify 10.0.0.10*.

Experimental Results. Table 4 shows our experiment results. Our ICS protocol simulations can respond to Nmap's scripts successfully and provide accurate data. Fig. 3 shows the IEC-104 script output when ICSNet simulates a PLC running IEC-104.

Fingerprint 3: Nmap IEC-104 script output.

```
PORT     STATE SERVICE REASON
2404/tcp open  iec-104 syn-ack ttl 128
| iec-identify:
|   ASDU address: 10
|_  Information objects: 5
```

4.4 Physical process evaluation

Cyber attacks against industrial control systems fundamentally differ from those against standard corporate information technology. Since the primary goal of most of these attacks is to alter a physical process or state, these attacks usually do not exploit a coding vulnerability in some software but rather use legitimate networking protocols to modify the behavior of the target. While attackers may need to exploit vulnerabilities in the traditional sense to gain access and privileges to tamper with the industrial control network, the attack against the physical process seldom uses a particular vulnerability but instead takes advantage of any acquired privileges.

As a result of these attacks, the attacker declares a successful attack when s/he observes changes in the state of the physical process. In an actual control system, the attacker uses industrial control protocols to send specific commands to the devices handling the process and retrieves sensor data to ascertain whether the attack was successful.

Because ICSNet simulates not only the network interaction of the devices but also the physical process behind them, any changes to the read-write values of the industrial control devices will, in turn, affect the physical simulation and present reasonable changes consistent with the physical process, visible in the read-only values of the simulated devices.

We evaluated this physical interaction by executing a plausible attack against the simulated processes. To test the feasibility of attacks against different control logic schemes, we devised the two scenarios proposed in section 3.7; one has a physical model controller, and the other has a state-based controller. From the available personalities, we randomly chose the "Allen-Bradley 1756 ENBT/A" as the personality for the required PLC devices with Modbus as the primary industrial control protocol.

We configured ICSNet with an appropriate network topology to follow the defined threat model, in which the attacker can communicate with the target PLC devices via the fingerprint proxy. This configuration lets the attacker interact with the simulated PLC devices as if s/he were in the same virtual switch (Figure 9).

In both scenarios, the attack is quite similar. Since the attacker can communicate with the simulated devices, s/he has to issue

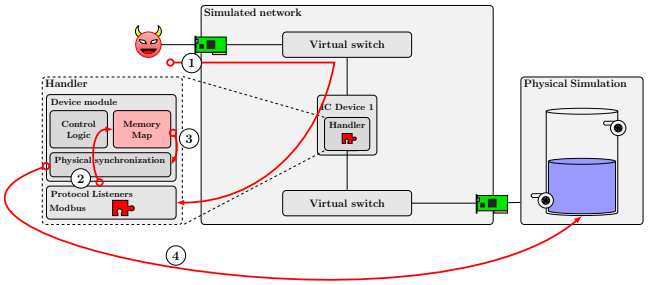


Figure 9: Water tank attack: scenario.

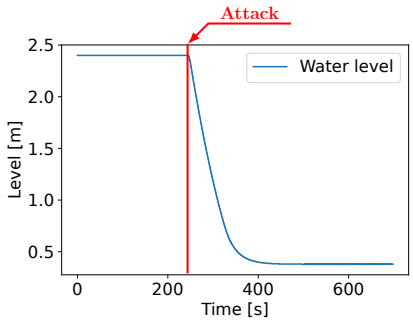


Figure 10: Water tank attack: Water level tampering.

an appropriate command to modify the physical synchronization component, constantly assert the sensor data from the physical simulation, updating the memory map sensor values. The new actuator values from the memory map are updated to the simulated holding register, where the PLC stores the target value.

From the viewpoint of ICSNet's implementation, the protocol listener handles the Modbus connection, accepts the command if the address is within the read-write memory mapping, and modifies the value as requested. Concurrently, the control logic constantly runs a control loop that reads the sensor data, runs the control logic, and adjusts the actuator values as needed. In addition, the physical synchronization component constantly reads the sensor data from the physical simulation, updates the memory map sensor values, asserts the new actuator values from the memory map, and updates them within the physical simulation.

Figure 9 shows the attack flow. ① The attacker sends the Modbus command to the simulated PLC. ② The Protocol Listener receives the command, updating the memory map and forcing the control logic to make the necessary changes. ③ The Physical Synchronization component reads the new actuator values due to the memory map tampering and, subsequently, ④ sends them to the physical simulation, which is ultimately affected by the attack.

To corroborate whether the attack was successful, the attacker must acquire some relevant sensor data from a physical source that should be affected by the intended attack. Since sensor data in a PLC is read-only, the attacker must send a command to retrieve this information from either a discrete input, in the case of boolean sensors, or an input register. Since the corresponding sensor values

in our chosen scenarios are not discrete, the attacker sends the Modbus command *Read input register* targeting the sensor address that stores the relevant value. In this case, the Protocol Listener accepts the command and returns the current values from the memory map to the attacker.

Our first attack aims to disrupt the process that requires a specific water level in the tank by attempting to empty the tank. The attacker must send a zero value to the holding register that stores the set point to accomplish this goal. After executing the attack as described above, we observed the behavior of the water level in the tank, shown in Figure 10. Before the attack, the control logic was kept stable, as the process required. When the malicious command modifies the Set Point in the memory map, the level decreases to a value close to zero, disrupting the process.

We now focus on attacking the warehouse model to test the versatility of our interaction with different physical processes. In this scenario, we are assuming that the attacker wants to retrieve a package from the warehouse's output without being able to interact with the warehouse's HMI. Suppose the attacker knows the ID of the warehouse spot in which the desired package resides. In that case, the attacker can send the Modbus command to the PLC, which stores the retrieval position in a specific holding register. As a proof of concept, we sent the command with an occupied position, which caused the warehouse to retrieve the package and place it in the output conveyor belt.

In summary, our high-fidelity physical process integration with Factory I/O allows an attacker to see the changes to the physical system under control. Our modular design can enable future researchers to extend our simulations to several other physical processes. Factory I/O allows designers to create essentially digital twins based on the sensors and actuators it supports.

5 Conclusions

As far as we know, our honeynet has the most extensive set of simulated devices, protocols, and physical processes; however, they still do not capture all possible configurations of ICS systems. While extending our honeynet to model more devices is likely, we believe these extensions are straightforward, given our modular design. This also applies to any additional physical process simulations.

One open limitation is integrating with **OpenPLC or CODESYS**. While ICSNet supports the most significant number of ICS devices in the literature, it does not currently support virtualized PLCs such as OpenPLC or CODESYS. These PLC frameworks have gained popularity in academic research [23, 30, 46] and industry. Integrating them into ICSNet would add even more ICS device support and increase our tool's fidelity. Thanks to ICSNet's modular design, it would be possible to extend ICSNet to include one of the above platforms.

Modern ICS Protocol Support. Although ICSNet already supports representative ICS protocols found in my many ICS, we would explore the simulation of integration of additional protocols in future work. Specifically, protocols such as OPC-UA [40] and MQTT [26]. These protocols are becoming more common thanks to their integration into new "IoT-ready" ICS devices [9, 24].

Acknowledgments

This work was supported in part by the NSF CPS program through CNS-1929410 by the US Department of Transportation (USDOT) Tier-1 University Transportation Center (UTC) Transportation Cybersecurity Center for Advanced Research and Education (CYBERCARE). (Grant No. 69A3552348332).

References

[1] 2005. SCADA HoneyNet Project: Building Honeypots for Industrial Networks. http://scadahoneynet.sourceforge.net/. Accessed: 2023-10-27.
[2] 2021. python-nmap. https://pypi.org/project/python-nmap/. Accessed: 2023-10-27.
[3] 2023. First All-In-One PLC Simulation Software. https://www.nirtec.com/. Accessed: 2024-04-15.
[4] 2023. Industrial Control Systems | Cybersecurity and Infrastructure Security Agency. https://www.cisa.gov/topics/industrial-control-systems
[5] 2023. TCP/IP Fingerprinting Methods Supported by Nmap. https://nmap.org/book/osdetect-methods.html. Accessed: 2023-10-27.
[6] 2024. Next-Gen PLC Training - Factory IO. https://factoryio.com/. Accessed: 2024-04-15.
[7] Daniele Antonioli, Anand Agrawal, and Nils Ole Tippenhauer. 2016. Towards high-interaction virtual ICS honeypots-in-a-box. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy (co-located with CCS)*. ACM, 13–22.
[8] Daniele Antonioli and Nils Ole Tippenhauer. 2015. MiniCPS: A Toolkit for Security Research on CPS Networks. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy* (Denver, Colorado, USA) *(CPS-SPC '15)*. Association for Computing Machinery, New York, NY, USA, 91–100. https://doi.org/10.1145/2808705.2808715
[9] Cloud Connectivity – IoT PLC: Controllers with MQTT 2024. Cloud Connectivity – IoT PLC: Controllers with MQTT. https://www.wago.com/us/open-automation/cloud-connectivity/iot-plc-controller-with-mqtt. Accessed: 27-09-2023.
[10] Mauro Conti, Francesco Trolese, and Federico Turrin. 2022. ICSpot: A High-Interaction Honeypot for Industrial Control Systems. In *2022 International Symposium on Networks, Computers and Communications (ISNCC)*. 1–4. https://doi.org/10.1109/ISNCC55209.2022.9851732
[11] Presidential Policy Directive. 2013. Critical infrastructure security and resilience. PPD-21, Released February 12, 2013.
[12] Ilya Etingof. 2020. SNMP Simulator. https://github.com/etingof/snmpsim. Accessed: 2023-10-27.
[13] Wenjun Fan, Zhihui Du, and David Fernández. 2015. Taxonomy of honeynet solutions. In *2015 SAI Intelligent Systems Conference (IntelliSys)*. IEEE, 1002–1009.
[14] Javier Franco, Ahmet Aris, Berk Canberk, and A Selcuk Uluagac. 2021. A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems. *IEEE Communications Surveys & Tutorials* 23, 4 (2021), 2351–2383.
[15] Elisavet Grigoriou, Athanasios Liatifis, Panagiotis Radoglou Grammatikis, Thomas Lagkas, Ioannis Moscholios, Evangelos Markakis, and Panagiotis Sarigiannidis. 2022. Protecting IEC 60870-5-104 ICS/SCADA Systems with Honeypots. In *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*. 345–350. https://doi.org/10.1109/CSR54599.2022.9850329
[16] Jan Iversen. 2023. pymodbus-dev/pymodbus: A full modbus protocol written in python. https://github.com/pymodbus-dev/pymodbus. Accessed: 2023-10-31.
[17] Perry Kundert. 2023. Comm. Protocol Python Parser and Originator. https://github.com/pjkundert/cpppo. Accessed: 2023-10-27.
[18] Samuel Litchfield, David Formby, Jonathan Rogers, Sakis Meliopoulos, and Raheem Beyah. 2016. Rethinking the Honeypot for Cyber-Physical Systems. *IEEE Internet Computing* 20, 5 (2016), 9–17. https://doi.org/10.1109/MIC.2016.103
[19] Efrén López-Morales, Carlos Rubio-Medrano, Adam Doupé, Yan Shoshitaishvili, Ruoyu Wang, Tiffany Bao, and Gail-Joon Ahn. 2020. HoneyPLC: A Next-Generation Honeypot for Industrial Control Systems. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, Virtual Event, USA, 279–291. https://doi.org/10.1145/3372297.3423356
[20] Marco Lucchese, Francesco Lupia, Massimo Merro, Federica Paci, Nicola Zannone, and Angelo Furfaro. 2023. HoneyICS: A High-interaction Physics-aware Honeynet for Industrial Control Systems. (2023).
[21] Gordon Lyon. 2023. Nmap Scripting Engine (NSE). https://nmap.org/book/man-nse.html. Accessed: 2023-10-30.
[22] Gordon Lyon. 2023. NSEDoc Reference Portal: NSE Libraries - Nmap Scripting Engine documentation. https://nmap.org/nsedoc/lib/. Accessed: 2023-10-30.
[23] Efrén López-Morales, Jacob Hopkins, Alvaro A. Cardenas, Ali Abbasi, and Carlos Rubio-Medrano. 2024. By the Numbers: Towards Standard Evaluation Metrics for Programmable Logic Controllers' Defenses. In *Proceedings of the 2nd International*

Workshop on Re-design Industrial Control Systems with Security (RICSS) (co-located with CCS). ACM.
[24] Efrén López-Morales, Ulysse Planta, Carlos Rubio-Medrano, Ali Abbasi, and Alvaro A. Cardenas. 2024. SoK: Security of Programmable Logic Controllers. In *33rd USENIX Security Symposium (USENIX Security 24)*.
[25] John Matherly. 2023. Shodan Search Engine. https://www.shodan.io/dashboard. Accessed: 2023-11-02.
[26] MQTT: The Standard for IoT Messaging 2022. MQTT: The Standard for IoT Messaging. https://mqtt.org/. Accessed: 03-11-2023.
[27] Mininet project. 2022. Mininet: An Instant Virtual Network on your Laptop (or other PC). http://mininet.org/.
[28] Linux Foundation Collaborative Projects. 2016. Open vSwitch. https://www.openvswitch.org/.
[29] Niels Provos et al. 2004. A Virtual Honeypot Framework.. In *USENIX Security Symposium*, Vol. 173. 1–14.
[30] Prashant Hari Narayan Rajput, Esha Sarkar, Dimitrios Tychalas, and Michail Maniatakos. 2021. Remote Non-Intrusive Malware Detection for PLCs based on Chain of Trust Rooted in Hardware. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 369–384.
[31] Ubuntu Manpage Repository. 2019. Wget - The non-interactive network downloader. https://manpages.ubuntu.com/manpages/lunar/en/man1/wget.1.html. Accessed: 2023-10-27.
[32] Rohith Raj S, Rohith R, Minal Moharir, and Shobha G. 2018. SCAPY- A powerful interactive packet manipulation program. In *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*. 1–5. https://doi.org/10.1109/ICNEWS.2018.8903954
[33] L. Salazar, S. Castro, J. Lozano, K. Koneru, E. Zambon, B. Huang, R. Baldick, M. Krotofil, A. Rojas, and A. Cardenas. 2024. A Tale of Two Industroyers: It was the Season of Darkness. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 161–161. https://doi.org/10.1109/SP54263.2024.00162
[34] Luis Salazar, Neil Ortiz, Xi Qin, and Alvaro A. Cardenas. 2020. Towards a High-Fidelity Network Emulation of IEC 104 SCADA Systems. In *Proceedings of the 2020 Joint Workshop on CPS&IoT Security and Privacy* (Virtual Event, USA) *(CPSIOTSEC'20)*. Association for Computing Machinery, New York, NY, USA, 3–12. https://doi.org/10.1145/3411498.3419969
[35] Shreyas Srinivasa, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis. 2022. Interaction Matters: A Comprehensive Analysis and a Dataset of Hybrid IoT/OT Honeypots. In *Proceedings of the 38th Annual Computer Security Applications Conference* (Austin, TX, USA) *(ACSAC '22)*. Association for Computing Machinery, Austin, TX, USA, 742–755. https://doi.org/10.1145/3564625.3564645
[36] Keith Stouffer, Joe Falco, Karen Scarfone, et al. 2011. Guide to industrial control systems (ICS) security. *NIST special publication* 800, 82 (2011), 16–16.
[37] Keith Stouffer, Michael Pease, C Tang, Timothy Zimmerman, Victoria Pillitteri, and Suzanne Lightman. 2023. Guide to Operational Technology (OT) Security. *National Institute of Standards and Technology: Gaithersburg, MD, USA* NIST SP 800-82 Rev. 3 (Sept. 2023). https://doi.org/10.6028/NIST.SP.800-82r3
[38] Chris Sullo. 2023. Nikto. https://github.com/sullo/nikto. Accessed: 2023-11-01.
[39] The MITRE Corporation. 2020. Reconnaissance, Tactic TA0043 - Enterprise | MITRE ATT&CK®. (oct 2020). https://attack.mitre.org/tactics/TA0043/ Accessed: 29-09-2023..
[40] Unified Architecture 2019. Unified Architecture - OPC Foundation. https://opcfoundation.org/about/opc-technologies/opc-ua/. Accessed: 03-10-2023.
[41] Johnny Vestergaard. 2023. Conpot. https://github.com/mushorg/conpot. Accessed: 2023-11-01.
[42] Susan Marie Wade. 2011. SCADA Honeynets: The attractiveness of honeypots as critical infrastructure security tools for the detection and analysis of advanced threats. (2011).
[43] Gérard Wagener, Radu State, Thomas Engel, and Alexandre Dulaunoy. 2011. Adaptive and self-configurable honeypots. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. 345–352. https://doi.org/10.1109/INM.2011.5990710
[44] Feng Xiao, Enhong Chen, and Qiang Xu. 2017. S7commTrace: A High Interactive Honeypot for Industrial Control System Based on S7 Protocol. In *Int. Conference on Information and Communications Security*. Springer, 412–423.
[45] Jianzhou You, Shichao Lv, Yue Sun, Hui Wen, and Limin Sun. 2021. HoneyVP: A Cost-Effective Hybrid Honeypot Architecture for Industrial Control Systems. In *ICC 2021 - IEEE International Conference on Communications*. 1–6. https://doi.org/10.1109/ICC42927.2021.9500567
[46] Wenhui Zhang, Yizheng Jiao, Dazhong Wu, Srivatsa Srinivasa, Asmit De, Swaroop Ghosh, and Peng Liu. 2019. Armor PLC: A Platform for Cyber Security Threats Assessments for PLCs. *Procedia Manufacturing* 39 (2019), 270–278.

Appendix

Reconnaissance Tools

Reconnaissance is a technique that involves actively or passively gathering information that can be used to support targeting. Such information may include details of the victim's organization and infrastructure [39]. In the context of ICS, the following tools help implement this technique.

Nmap. Nmap is a free and open-source utility for network exploration and security auditing. Nmap uses raw IP packets to determine what hosts are reachable over the network, what services these hosts offer, e.g., telnet, and what operating systems they are running. Nmap includes an extensive database to identify systems based on their response to TCP/IP probes. Nmap allows users to write scripts that extend Nmap's features [21]. A diverse catalog of scripts, including ICS-related scripts, is suited for different purposes. For example, the *modbus-discover* script enumerates Modbus devices and collects their device information [22].

Nikto. Nikto [38] is a command-line vulnerability scanner that scans web servers for dangerous files. It is written in Perl using LibWhisker to perform fast security or informational checks.