# PLCpot: Application Dialogue Replay based Scalable PLC Honeypot for Industrial Control Systems

Syed Ali Qasim
qasims@gvsu.edu
Grand Valley State University
Allendale, Michigan, USA

Muhammad Taqi Raza
taqi@umass.edu
University of Massachusetts Amherst
Amherst, Massachusetts, USA

Irfan Ahmed
iahmed3@vcu.edu
Virginia Commonwealth University
Richmond, Virginia, USA

## ABSTRACT

Programmable Logic Controllers (PLCs) are essential components of industrial control systems (ICS), overseeing critical processes like manufacturing and power generation. As cyberattacks grow in sophistication, the security community uses PLC honeypots to gather threat intelligence on attackers' tools and strategies. Existing PLC honeypots, whether low or high interaction, often face challenges in maintaining realism or supporting complex interactions. This paper presents PLCpot, a protocol-agnostic and scalable PLC honeypot framework designed to emulate PLC communication by analyzing and replaying network traffic. By identifying dynamic fields and function codes within protocols and mapping them to application-level operations, PLCpot supports features such as control logic transfer, basic authentication, and operational modes to enhance attacker engagement.

We demonstrate PLCpot's emulation capabilities with multiple PLC types, evaluating its potential to replicate common functional and operational behaviors. Additionally, a case study involving a lab-based elevator model showcases PLCpot's ability to engage attackers and capture data for analysis. While PLCpot currently supports basic ICS protocols over the transport layer, this framework advances ICS threat intelligence by providing a versatile and scalable approach for emulating PLC behavior and collecting attack data to inform future security measures.

## CCS CONCEPTS

• **Security and privacy** → **Intrusion Detection Systems**; • **Computer systems organization** → *Embedded and cyber-physical systems security*.

## KEYWORDS

Critical Infrastructure, Industrial Control Systems, Honeypots

**ACM Reference Format:**
Syed Ali Qasim, Muhammad Taqi Raza, and Irfan Ahmed. 2025. PLCpot: Application Dialogue Replay based Scalable PLC Honeypot for Industrial Control Systems. In *ACM/IEEE 16th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2025) (ICCPS '25), May 6–9, 2025, Irvine, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3716550.3722032


*ICCPS '25, May 6–9, 2025, Irvine, CA, USA*

ACM ISBN 979-8-4007-1498-6/2025/05
https://doi.org/10.1145/3716550.3722032

## 1 INTRODUCTION

Programmable Logic Controllers (PLCs) are crucial to modern industrial control systems (ICS), directly monitoring and controlling various processes such as manufacturing, power generation, and chemical processing. They execute user-defined control logic based on sensor input signals and send output to actuators. PLCs also communicate with other systems like human-machine interfaces (HMIs) and supervisory control and data acquisition (SCADA) systems. Due to their critical role, PLCs have become prime targets for increasingly sophisticated cyberattacks[1, 2, 4, 5, 7, 9, 10, 16, 25, 29], requiring the security community to develop effective threat intelligence capabilities to stay ahead of attackers.

PLC honeypots are one such solution that can help security professionals gather valuable threat intelligence by attracting and monitoring attackers targeting ICS. While physical honeypots offer effective intelligence gathering, they come with high hardware and deployment costs[30]. Fortunately, many researchers have focused on developing virtual honeypots [6, 8, 11, 14, 17, 26–28] that can simulate the behavior of a real PLC. There are two types of virtual honeypots: low-interaction and high-interaction. Low-interaction honeypots simulate only parts of a real system's functionality, making them easy to set up and resource-efficient but limited in functionality and easily detectable by attackers. High-interaction honeypots aim to emulate the entire PLC system, including hardware, software, and network environments, offering more comprehensive data on attacker behavior but requiring greater resources and expertise to set up and maintain.

However, existing ICS honeypots have limitations that hinder effective engagement with attackers and useful threat intelligence collection. Low-interaction honeypots are unable to engage and attract sophisticated attackers, while high-interaction ones lack operational-level functionalities such as control logic transfer, PLC modes, and support for different PLC functions. Due to the lack of operational-level support, most high-interaction honeypots can be easily identified and may not be able to engage the attacker for longer sessions. To address this, more advanced honeypots with operational-level capabilities are needed for effective engagement.

In this paper, we present PLCpot, a protocol-agnostic, scalable PLC honeypot developed based on our ICS insights that communication between the PLC and any client is deterministic and can be replayed to mimic a real PLC. To achieve this, PLCpot analyzes multiple network dumps of a PLC, learning its protocol, the location of dynamic fields present in the protocol message (which need to be updated for successful replay), various function codes used by the PLC, and mapping these function codes with application-level operations in a template. Using this template, PLCpot can replay network dumps, acting as a real PLC on the network and providing

various application-level features such as control logic transfer, PLC authentication, and PLC modes.

Our contributions are threefold:

- We introduce PLCpot, a scalable and protocol-agnostic honeypot framework capable of providing application-level features.
- We demonstrate PLCpot's ability to mimic different PLCs and provide experimental evidence of its capabilities to replicate various functional and operational features.
- We present a case study using a lab model of an elevator with PLCpot, performing various ICS attacks on it, showcasing its capacity to engage the attacker and store attack data.

# 2 BACKGROUND AND RELATED WORK

## 2.1 Programmable Logic Controllers in ICS

Programmable Logic Controllers (PLCs) are embedded devices that control and monitor physical processes using input/output modules connected to sensors and actuators. The control logic, a user-defined program, manages input processing and output delivery. Users can configure, program, and maintain PLCs using proprietary engineering software, allowing them to write and read control logic to and from PLC memory[3, 18, 20–22],. PLCs also support network communication, enabling interaction with various ICS services such as engineering software and human-machine interfaces (HMIs), as shown in figure 1.

**PLC Operational and Functional Features** Along with transferring the control logic, the PLC also has several other operational and functional features:

- **Authentication:** PLCs offer password protection for login and state changes.
- **Report I/O data:** When connected to engineering software or HMI, PLCs reply to I/O-related requests to show the ICS state.
- **PLC modes:** PLCs have modes like "Run" and "Program". "Program" mode allows control logic writing, while "Run" mode executes control logic and blocks write memory operations.
- **Session Establishment:** PLCs accept remote connections and have session establishment protocols in their firmware.
- **Session Maintenance:** PLCs respond to incoming requests for operations like read, write memory, authentication, mode change, and reading I/O data.

## 2.2 Related Work on ICS Honeypots

Honeypots are decoy systems designed to mimic real systems, attracting and engaging attackers. They can be classified into low-interaction or high-interaction honeypots based on the functionality and level of engagement they provide. Numerous PLC honeypots have been developed for ICS[6, 8, 11, 14, 26–28]. Among the existing honeypots, HoneyPLC[14] and ICSpot[8] are closest to PLCpot in terms of the functionalities they provide. HoneyPLC supports control logic upload and download functionality for Siemens PLCs and can establish and maintain a session with STEP7 engineering software. ICSpot, based on HoneyPLC, provides all the features of HoneyPLC and includes dedicated modules for simulating I/O data from the physical process. Table 1 summarizes our assessment of various features and capabilities of existing honeypots, such as protocol coverage (the number of different protocols a honeypot supports), control logic transfer (whether the honeypot can upload and download control logic), device discovery (if the honeypot is identified as a real PLC by engineering software and other network scanning tools), and the operational and functional features the honeypot provides (application-level interaction).

## 2.3 Limitations of State-of-the-art Honeypots

State-of-the-art honeypots have limitations that reduce their ability to engage attackers and collect valuable threat intelligence data. Some of these limitations are:

- **Limited coverage (L1)**: Current honeypots, both low and high interaction, cover a limited number of PLCs. They rely on libraries and frameworks for specific PLCs, developed through reverse engineering ICS protocols. For example, HoneyPLC, CryPLH, and ICSpot use the Snam7[24] framework for S7comm server simulation, while many Modbus-based honeypots depend on open-source Modbus libraries such as libmodbus[23]. These dependencies restrict the supported PLC range.
- **PLC Memory Manipulation (L2)**: Major ICS attacks, like Stuxnet[13], aim to manipulate PLC memory. Attackers seek to write malicious control logic or malware to disrupt the physical process. However, only HoneyPLC and ICSpot offer control logic download capability. The absence of control logic capture in most honeypots limits their attack surface and threat intelligence generation, making them unable to detect various attack types.
- **Lack of Operational and Functional Features (L3)**: PLCs offer features like authentication, modes, session establishment, and maintenance, which enhance the attack surface and enable better attacker engagement. However, inaccurate emulation of these features may reveal honeypot identities. ICSpot and HoneyPLC support application-level sessions, CryPLH and S7CommTrace partially support Siemens PLCs, but none fully support PLC mode features. Only CryPLH offers authentication, while ICSpot (S7comm-based) and NeuPot (Modbus-based) provide I/O data exchange.

Given these limitations, there is a pressing need for a dynamic, scalable honeypot capable of replicating the diverse operational and functional features of real PLCs to improve engagement with attackers and threat intelligence collection.

# 3 PLCPOT: ENABLING APPLICATION-LEVEL PLC FUNCTIONALITIES AT SCALE

## 3.1 Challenges in Developing a Scalable Honeypot

Developing a scalable and dynamic honeypot for different PLC operational features presents various challenges.

***Diverse Application Logic (C1):*** The heterogeneity in PLC application logic complicates honeypot functionality. Differences in authentication protocols, such as those in Schneider Electric's Modicon M221 and Allen-Bradley MicroLogix 1400 as shown in figure2,

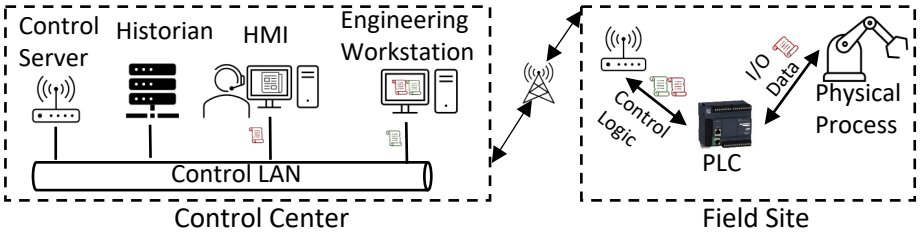


**Figure 1: An overview of Industrial Control Systems. The PLC communicates the actual state of the physical process (I/O data) to the engineering software and HMI. The engineering workstation runs the engineering software to control and configure the PLC.**

**Table 1: Summary of existing PLC honeypots in literature and their features ●= Complete Implementation ◐= Partial Implementation ○= No Implementation**

| Honeypot | ICS Protocols | | | Required ICS Protocol Libraries | Standard Operational and Functional Features | | | | | | | Device Discovery | | Additional IT Services | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Modbus | ENIP | S7comm | | Control Logic Upload | Control Logic Download | Authen-tication | Exchange I/O Data | PLC mode | Session Establisment | Session Maintaince | Engineering Software | Network Scanning | HTTP | FTP | SNMP |
| Conpot | ◐ | ◐ | ◐ | YES | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ◐ | ● | ○ | ● |
| HoneyNet | ◐ | ○ | ○ | YES | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ◐ | ● | ● | ● |
| ICSpot | ◐ | ○ | ● | YES | ● | ● | ○ | ● | ○ | ● | ● | ● | ● | ● | ○ | ● |
| NeuPot | ◐ | ○ | ○ | YES | ○ | ○ | ○ | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| HoneyPLC | ○ | ○ | ● | YES | ● | ● | ○ | ○ | ○ | ● | ● | ● | ● | ● | ○ | ● |
| CryPHL | ○ | ○ | ● | YES | ○ | ○ | ◐ | ○ | ○ | ◐ | ◐ | ◐ | ○ | ● | ○ | ● |
| s7CommTrace | ○ | ○ | ● | NO | ○ | ○ | ○ | ○ | ○ | ◐ | ○ | ○ | ● | ● | ○ | ● |

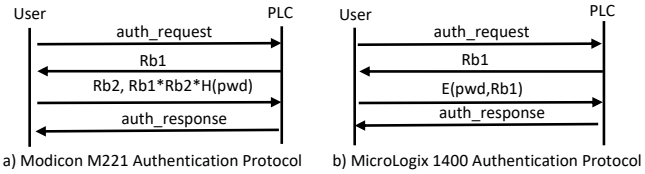


**Figure 2: Client Authentication Protocol used by different PLCs**

and variations in hashing algorithms [4] present challenges in replicating application logic.

***Proprietary PLC Protocols (C2):*** Proprietary ICS protocols used by PLCs for network communication pose challenges in replicating application-level functionalities. For example, M221 uses the UMAS protocol within Modbus, while MicroLogix 1400 employs the PCCC protocol encapsulated by the ENIP protocol. Understanding PLC protocol fields and semantics is a significant research challenge.

***Lack of PLC State Machine Knowledge (C3):*** PLCs have various programming modes or states, each with distinct state-switching mechanisms. For instance, MicroLogix 1400 directly enters "Run" mode after control logic download, while Modicon M221 requires a user command ('Start') in an intermediate state to initiate execution. Accurately implementing PLC state machines is challenging due to the lack of public documentation, requiring extensive manual experimentation.

## 3.2 PLC Communication Insights

We built PLCpot based on observations from real PLC interactions with engineering software: 1) deterministic communication, and 2) consistent binary chunk size during uploading and downloading.
**Deterministic Behavior:** Communication between the PLC and network entities involves requests and responses. Apart from messages requesting the current state of the physical process, all other messages are deterministic if the PLC configuration remains unchanged. Predictable communication patterns exist during sessions.

**Memory Read/Write Operations:** The engineering software converts control logic to machine-level binary, divides it into chunks, and writes these chunks onto PLC memory using "write" requests. To read control logic, the software sends "read" requests for different memory regions. Our analysis revealed that the software consistently divides machine-level binary into the same chunks and uses the same memory locations during upload and download operations as shown in figure 3. These messages can be exploited for replay attacks to replicate control logic operations on the PLC.

## 3.3 PLCpot Framework

To address honeypot limitations and challenges in 2.3 , we developed PLCpot, a scalable, protocol-agnostic honeypot. It uses packet replay techniques and network dumps from real PLCs to learn session-dependent dynamic fields. By abstracting operations and mapping function codes to functionalities like control logic transfer and PLC authentication, it creates a *PLC template* for each PLC. With this template, PLCpot replays network dumps, mimicking a real PLC on the network.

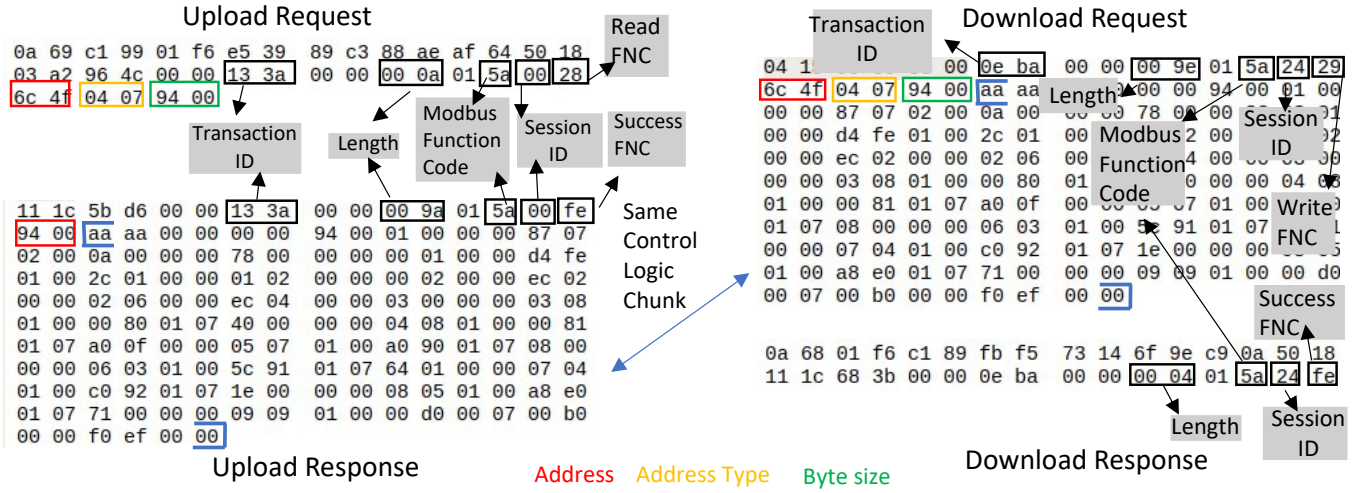


**Figure 3: Request-Response message to read and write a control logic on M221 memory**

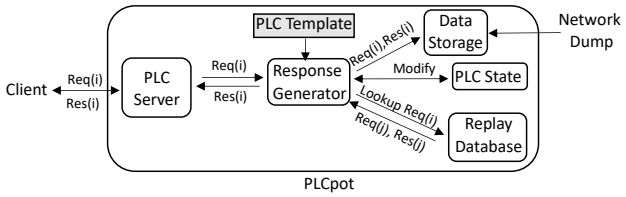


**Figure 4: Overview of PLCpot. The PLCpot requires network dump and PLC template to communicate with the client**

*3.3.1* ***PLC Template Generation.*** The creation of a generic honeypot with application-level functionalities is challenged by diverse PLC protocols and manufacturers' implementations. To address this, we developed the PLC Template for PLCpot, using benign network dumps from communication between engineering software and real PLCs to learn dynamic fields, function code locations, and application-level request-response messages.

### I - Uncovering Dynamic Fields in Proprietary ICS Protocols (C2)

The proprietary nature of ICS protocols poses challenges in designing dynamic honeypots. By replaying old network dumps, we simulate real PLC communication. Identifying and updating dynamic fields is crucial for this simulation. We use differential analysis on analogous messages from different sessions to uncover dynamic field locations and semantics in PLC protocols, as shown in Figure 5. The process explained below involves extracting request-response pairs from network dumps, grouping and comparing them to identify dynamic fields, and mapping request and response message values to verify usability.

**Message Identification & Pairing:** PLCpot collects multiple network dumps from PLC communication performing the same operations. It identifies request and response messages using IP address and port (Modicon M221 PLC uses port 502, with destination port 502 messages as requests and source port 502 messages as responses), working on transport layer payloads. For each request-response pair (Req_i, Res_i) in one dump, it finds a similar pair (Req_j, Res_j) in another dump, generating tuples (Req_i, Res_i, Req_j, Res_j) of the same message in different sessions as described in algorithm 1.

**Algorithm 1** Message Identification and Pairing

**Require:** $\text{Req}_i, \text{Res}_i \in$ Network Dump $i$
**Require:** $\text{Req}_j, \text{Res}_j \in$ Network Dump $j$
1: **for** each $\text{Req}_i$ **do**
2: **for** each $\text{Req}_j$ **do**
3: **if** $\text{len}(\text{Req}_i) == \text{len}(\text{Req}_j)$ and $\text{sim}(\text{Req}_i, \text{Req}_j)$ is max **then**
4: Pair $(\text{Req}_i, \text{Req}_j, \text{Res}_i, \text{Res}_j)$
5: **end if**
6: **end for**
7: **end for**

**Finding Session Dependant Fields:** PLCpot identifies various session-dependent fields in each tuple. For every tuple T, differential analysis is conducted between the respective request and response messages by employing algorithm 2. Figure 6 shows the dynamic field identified by comparing two Modicon M221 messages from different sessions.

**Dynamic field Mapping:** Upon identification of dynamic field locations in the request and response messages, the next step is determining their relationship. This is crucial for the PLCpot because to replay an old network dump, the PLCpot must update the dynamic fields in accordance with the new session. The process starts by identifying common dynamic fields in both messages and comparing their values. If they share the same values, PLCpot maps the dynamic fields in the response with the indices in the request message, enabling it to use the bytes present at the dynamic field

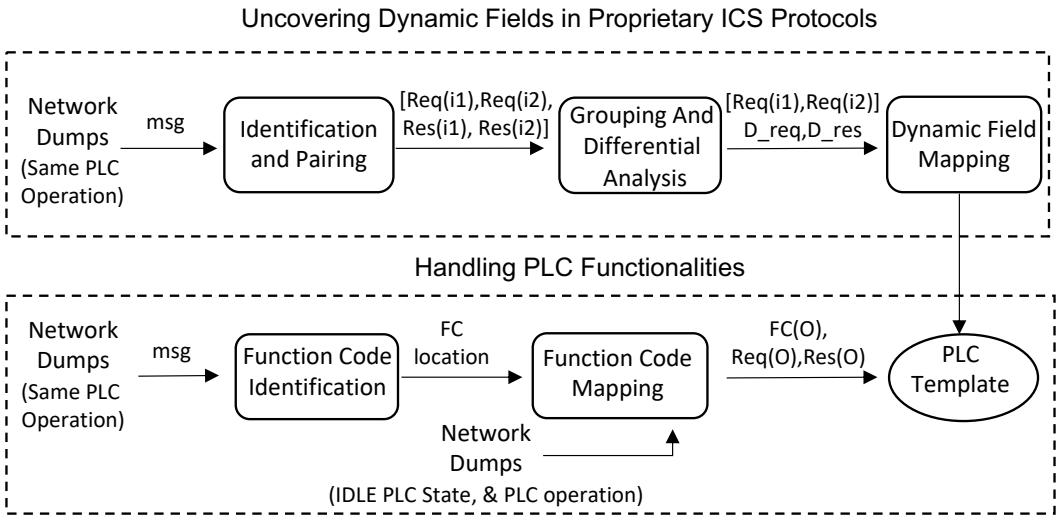


**Figure 5: Overview of PLC template generation**

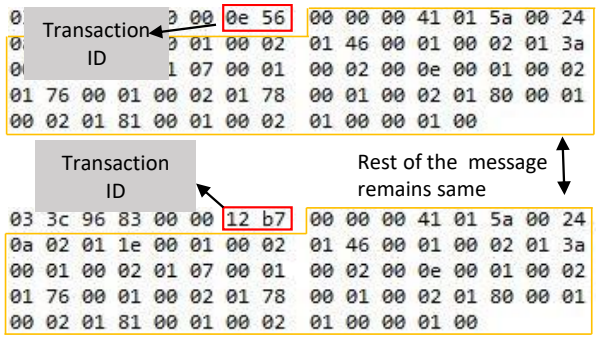


**Figure 6: Same message in two different sessions of Modicon M221 PLC**

**Algorithm 2** Dynamic Fields Detection

```
Initialize D_req ← ∅, D_res ← ∅
for T = [Req(i1), Req(j1), Res(i1), Res(j1)] do
    for a = 1 to |Req(i1)| do
        if Req(i1)[a] ≠ Req(j1)[a] then
            D_req ← D_req ∪ {a}
        end if
    end for
    for b = 1 to |Res(i1)| do
        if Res(i1)[b] ≠ Res(j1)[b] then
            D_res ← D_res ∪ {b}
        end if
    end for
end for
```

location to update the response message during the replay of any new request. This mapping, along with the location of dynamic fields, is stored in the PLC template.

### II - Handling PLC Functionalities (C1 & C3)

PLCpot provides an abstraction layer to address heterogeneous application logic implementations across PLCs. It delivers appropriate responses to operational and functional feature requests without executing corresponding actions, relying on observations to mimic various features by replaying suitable response messages.

**Function Code:** Function codes are specialized fields in ICS protocols for issuing commands, with manufacturers assigning distinct codes for operations like reading or writing data, starting or stopping PLCs, and running diagnostics.

**Function Code Identification:** Function code fields lack fixed patterns, making identification challenging. We developed heuristics based on ICS domain knowledge to identify function code locations by examining the limited number of unique function code values supported by each PLC.

**Heuristic:** Our heuristic identifies a field in an ICS message as a function code if it has a limited range of unique values in all request messages and at most two unique values in all response messages. We compare messages byte-by-byte within groups, and an index with variation within a threshold in request messages and at most two values in response messages indicates the function code's location.

**Function Code Mapping:** After identifying the function code location, we map function codes to application-level functionalities. We manually perform operations, capture network communication, extract function codes, and compare them with benign session function codes to map the remaining function codes to application-level functionality.

*3.3.2* ***PLCpot Modules.*** Figure 4 shows the PLCpot modules and their functions:

**Replay Database:** Organizes messages into request-response pairs for replaying network dumps. It uses fixed ports and a protocol-agnostic approach to pair messages based on source-destination port numbers and IP addresses, storing payloads in a hash table.

**PLC State:** Stores the initial state of the PLCpot's, including PLC mode and password. This module adjusts the state upon client request, allowing users to modify PLC features remotely to deceive attackers convincingly.

**Data Storage:** Preserves all communication between the PLCpot and clients for threat intelligence generation and forensic analysis.

**PLC Server:** Facilitates network communication by opening a TCP/UDP socket on the same port used by actual PLCs. This module serves as the gateway for client interaction, forwarding messages to the response generator module and relaying generated responses to clients.

**Response Generator:** Processes request messages and searches for similar requests in the replay database using string similarity and message length. It identifies two types of request messages: those that alter PLC memory (e.g., control logic download, PLC mode change, password updates) and those that do not (e.g., control logic upload, echo/ping messages, session establishment/maintenance). The response generator updates the PLC state for memory change operations, updates session-dependent dynamic fields in the response message, and forwards the response to the communicating entity.

# 4 EVALUATION

Existing PLC honeypots lack application-level interaction and operational features of real PLCs, which are crucial for expanding the honeypot's attack surface and gathering meaningful information. To evaluate PLCpot's capabilities, we address the following research questions:

- Q1 Can PLCpot be recognized as a real PLC by widely used engineering software across different PLC vendors?
- Q2 Does PLCpot accurately emulate critical operational and functional features such as connection establishment, control logic upload/download, authentication, and PLC modes?

## 4.1 Experimental Setup and Methodology

Our experimental setup included Allen-Bradley Micrologix 1400, Micrologix 1100, and Schneider Electric Modicon M221 PLCs. We used SoMachineBasic and RsLogix 500 engineering software, running on a Windows 10 VM (engineering workstation), to configure and program the Schneider Electric and Allen-Bradley PLCs. The PLCpot ran on an Ubuntu 18 VM, and all devices were on the same network.

To evaluate PLCpot's various functions, we first executed a targeted function on the real PLC using the engineering software and captured the network traffic. Then, we provided the network dump to PLCpot and performed the same function on it. Finally, we assessed if PLCpot could deliver the same features and functionalities as the real PLC.

## 4.2 Device Discovery - Q1

To address a limitation of existing PLC honeypots, which often lack realistic device discovery capabilities, we evaluate PLCpot's ability to be detected and identified as a real PLC. Device discovery is essential for establishing communication, as it allows the honeypot to be scannable and identifiable on the network, thereby enhancing its engagement potential. We assessed PLCpot's discoverability using widely-used engineering software such as RSLogix 500 and SoMachineBasic.

**Methodology:** In this experiment, we first used RSLogix and SoMachineBasic to discover a real PLC and captured the network communication between the PLC and the engineering software. We configured a driver in RSLinx Classic (provided by the PLC vendor) to connect to the PLC via its IP address. Using this network dump, we configured PLCpotwith the PLC template and activated the PLCpotserver. Finally, we used the discovery function in the engineering software to verify if PLCpotwas identifiable on the network by its IP address.

**Evaluation Criteria:** To validate PLCpot's realism, we set criteria that it should be identified by RSLogix 500 as a MicroLogix 1400 and 1100 PLC and by SoMachineBasic as a Modicon M221 PLC.

**Results:** The PLCpot was identified by RSLogix as a real MicroLogix 1400 and MicroLogix 1100 PLC, whereas SoMachineBasic identified it as a Modicon M221 PLC, as shown in Figure 7.

## 4.3 Operational and Functional Features - Q2

**1- Session Establishment and Maintenance**

PLCs, acting like servers, must establish and maintain communication sessions. After verifying that PLCpot is a genuine PLC, we tested its ability to establish and maintain communication sessions. During a session, the engineering software periodically sends various ping/ messages to the PLC. If the PLCpot does not respond correctly to these messages, the engineering software terminates the connection with an error.

**Methodology:** We used network dumps captured from a real PLC to initialize and run PLCpot for each PLC. Then, using engineering software, we attempted to discover and establish communication sessions with PLCpot. We ran multiple experiments with varying session durations, recording request-response messages.

**Evaluation Criteria:** We set the following criteria: 1) PLCpot should handle various message types; 2) PLCpot should send appropriate responses for each request; 3) PLCpot should maintain sessions of different lengths (5, 15, 30 mins); 4) No timeouts or disconnects should occur.

**Results:** PLCpot successfully established and maintained communication with engineering software in all experiments without any errors or connection terminations from the engineering software. As shown in Table 2, PLCpot managed communication sessions for set durations. For all three PLCs, PLCpot successfully responded to hundreds of requests in 5, 15, and 30-minute sessions, handling 9 unique function codes for the Modicon M221 and 2 unique function codes for both the MicroLogix PLCs. The number of unique function codes remained the same for all sessions because the engineering software sends the same request messages repeatedly in idle conditions, inquiring about the PLC state and reading IO values.

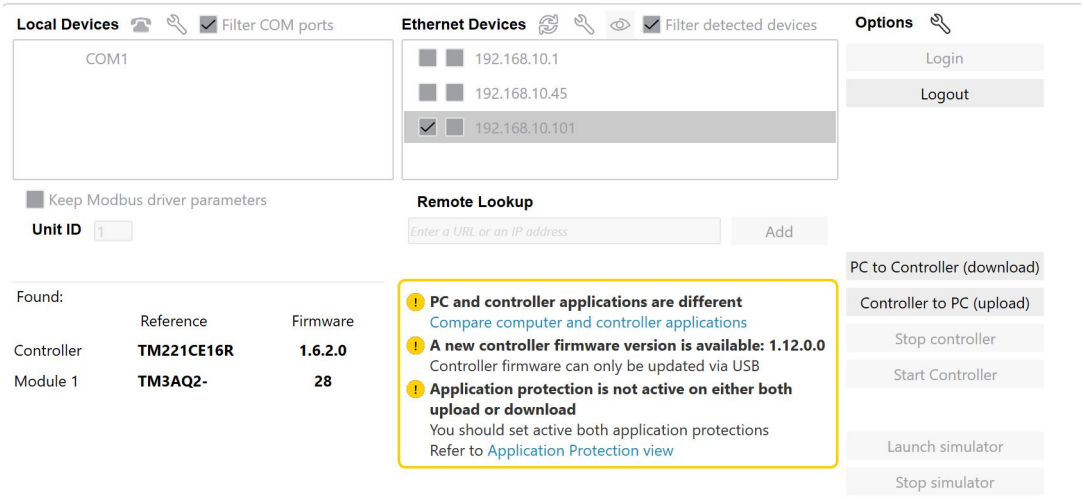


**Figure 7: PLCpot identified as a real Modicon M221 PLC in SoMachineBasic**

**Table 2: Summary of messages exchanged between a `plcpot` and the engineering software for various sessions.**

| PLC | Session Length (min) | # of req msg | # of res msg | # of unique FC | Session Timeout/ Disconnects |
|---|---|---|---|---|---|
| MicroLogix 1100 | 5 | 8153 | 8153 | 2 | 0 |
| | 15 | 24460 | 24460 | 2 | 0 |
| | 30 | 48926 | 48926 | 2 | 0 |
| MicroLogix 1400 | 5 | 10569 | 10569 | 2 | 0 |
| | 15 | 31720 | 31720 | 2 | 0 |
| | 30 | 63460 | 63460 | 2 | 0 |
| Modicon M221 | 5 | 2678 | 2678 | 9 | 0 |
| | 15 | 8098 | 8098 | 9 | 0 |
| | 30 | 16126 | 16126 | 9 | 0 |

**2- Authentication**

Many PLCs offer an authentication feature, allowing users to set a password. When establishing a communication session or performing critical operations, the PLC verifies the password. As shown in figure 2, authentication typically involves a challenge-based mechanism, where the PLC sends a challenge, the user or engineering software responds with an encrypted password, and the PLC validates the credentials before granting access. However, different PLCs implement authentication using proprietary algorithms and mechanisms, making a universal implementation challenging.

To address this complexity, PLCpot provides an authentication abstraction, allowing administrators to define authentication behavior within the PLC template. Instead of performing real password verification, PLCpot replays previously captured authentication exchanges, making it appear as though the system is validating credentials. The user can configure the PLC template to approve or deny authentication attempts based on predefined settings, ensuring flexibility while maintaining deception. This approach ensures that attackers receive responses mimicking real authentication interactions, reducing the likelihood of easy detection.

**Methodology:** To implement authentication mimicry, we first set up password authentication on a real Modicon M221 PLC using its engineering software. We then captured authentication exchanges for both successful and failed login attempts. By analyzing these network interactions, we identified authentication-related messages, including challenge-response messages, function codes, and response patterns.

Next, we integrated this authentication data into PLCpot's template. Instead of verifying passwords, PLCpot replays captured authentication request-response messages to simulate real authentication behavior. The user can configure PLCpot to control authentication outcomes (e.g., always approve, always deny, or randomly allow access), adding further variability to deceive attackers. Finally, we ran PLCpot and made some authentication attempts.

**Evaluation Criteria:** For this experiment, we set the criteria that PLCpot should respond to authentication requests and approve or disapprove them according to the PLC template.

**Results:** In our experiments, PLCpot successfully managed authentication requests according to the PLC template. Additionally, it captured the attacker's password hash, which can be used for forensic analysis.

**3- PLC Modes**

PLC mode, an operational feature altered using engineering software, can be exploited to disrupt physical processes, as noted by Syed et al. [19]. It is crucial for PLCpot to replicate PLC modes to effectively engage attackers. We evaluated this on Micrologix 1400, 1100, and Modicon M221 PLCs.

**Table 3: Summary of control logic download & upload for M221 PLC**

| File Size (KB) | # of Programs | Total Rungs | | Total Instructions | | Download Operations | | Upload Operations | | No. of Connection Timeouts |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Download | Upload | Download | Upload | Total Req | # Write Req. | Total Req | # Read Req. | |
| 60-80 | 24 | 66 | 66 | 258 | 258 | 9050 | 1126 | 5124 | 1228 | 0 |
| 81-90 | 5 | 19 | 19 | 51 | 51 | 1888 | 255 | 917 | 265 | 0 |
| 91-100 | 5 | 45 | 45 | 129 | 129 | 1996 | 245 | 894 | 249 | 0 |
| 101-120 | 3 | 32 | 32 | 80 | 80 | 1178 | 151 | 516 | 141 | 0 |
| 120+ | 3 | 52 | 52 | 233 | 233 | 1176 | 173 | 509 | 168 | 0 |
| Total | 40 | 214 | 214 | 751 | 751 | 15288 | 1950 | 7960 | 2051 | 0 |

**Methodology:** We established sessions with real PLCs and changed their modes using the engineering software i.e "Run" to "program" (for Micrologix 1400 and 1100) or "start" to "stop" and vice versa for Modicon M221. We captured the network communication, identified functions and messages for PLC modes, and updated the PLC template. Finally, we tested mode changes on PLCpot multiple times.

**Evaluation Criteria:** 1) PLCpot should respond to mode change requests and support all modes. 2) PLCpot should update and return its mode based on user requests.

**Result:** PLCpot successfully changed its mode in each cycle and maintained connection without disruption, indicating its ability to engage attackers targeting PLC modes. Additionally, PLCpot logged communication for potential threat intelligence extraction.

**4- Control Logic Download**
Control logic download is a prime target in PLC cyber attacks, where attackers aim to disrupt physical processes by injecting malicious control logic. Thus, it is crucial for PLCpot to allow users to download malicious control logic and store it for threat intelligence gathering. We tested PLCpot's capability to handle control logic downloads.

**Methodology:** We initiated PLCpot with the Modicon M221 PLC template and established a communication session using SoMachineBasic software. We then downloaded control logic programs of varying sizes and complexities onto PLCpot using the engineering software, capturing the network communication for evaluating the upload functionality.

**Evaluation Criteria:** The evaluation criteria required PLCpot to handle all control logic download messages without timeouts or session disconnects, ensuring the user receives confirmation of control logic download from the engineering software.

**Results:** Our experiments demonstrated that PLCpot successfully handled all download requests. As seen in Table 3, we downloaded 40 control logic programs to the PLCpot. During these operations, it received 15,288 request messages, including 1,950 messages to write control logic on PLC memory. PLCpot effectively responded to all messages and stored the communication, which can be used to extract the control logic binary.

**5- Control Logic Upload**
Control logic upload is an essential feature for PLCpot to support, as it is often used in cyber attacks [12, 15]. Attackers perform reconnaissance by uploading or reading control logic from the PLC memory, gaining insights into the physical process's inputs, outputs, and current state. This allows them to create efficient and impactful malicious control logic. Additionally, after downloading malicious control logic to the PLCpot, attackers may perform upload operations to verify that the desired malicious control logic is running on the PLC.

**Methodology:** To evaluate PLCpot's control logic upload functionality, we began by running PLCpot with the M221 template and providing it with network dumps captured during the control logic download experiments. We then used the engineering software's upload functionality to read the control logic that was previously downloaded to the PLC. Finally, we compared the downloaded and uploaded control logic to assess the transfer accuracy. For each control logic, we manually compared individual rungs and instructions.

**Evaluation Criteria:** The evaluation criteria for this experiment are: 1)PLCpot must handle all read control logic requests, 2) engineering software must successfully upload the entire control logic from PLCpot without errors or session disconnects, and 3) the uploaded control logic should match the one previously downloaded, with the same rungs and instructions in order.

**Results:** As shown in Table 3 , our evaluation demonstrates that PLCpot successfully uploaded all 40 control logic files using the network dump. During our experiments, the PLCpot received over 7960 messages, including 2052 control logic read requests. Additionally, we compared the uploaded control logic program with the original program, and they had identical rungs and instructions in the same order, indicating that the PLCpot can effectively replicate the real PLC.

# 5 CASE STUDY: PLCPOT FOR ELEVATOR SYSTEM

To assess PLCpot's ability to engage with attacks and capture attack artifacts, we conducted a case study using a lab model elevator system controlled by a Modicon M221 PLC. The goal was to evaluate how well PLCpotcould mimic a real PLC in an ICS threat scenario.

To achieve this, we first downloaded the elevator control logic onto a real M221 PLC and captured the resulting network communication. This network dump, which reflects interactions between the PLC and the engineering software, was then fed into PLCpot. By replaying this captured traffic, PLCpotwas able to simulate the behavior of the core control application.

Next, we generated a PLC template using network dumps from 40 different control logic program downloads on a real M221 PLC, covering various configurations and sizes. Additionally, we captured authentication messages by performing both successful and unsuccessful authentication attempts on the real M221 PLC. These authentication interactions were also incorporated into the PLC template. The study setup included a Windows 10 VM for running the engineering software and two Ubuntu 20 VMs—one for running PLCpotand another for executing attacks. Multiple ICS attacks were conducted on the elevator system, and the resulting network communication was captured and analyzed.

## 5.1 Adversary Model and Attack Scenario

In our case study, we envision an adversary who has gained network access and can connect to the PLC, sending and receiving messages. This adversary has a moderate understanding of PLC operations and control logic but lacks insider knowledge of the specific physical process or detailed input/output (I/O) data. The attacker can perform the following actions:

- Upload the control logic from PLCpot to analyze and infer details about the physical process.
- Write and download malicious control logic to PLCpot's memory.
- Execute common PLC functions without a deep understanding of the proprietary protocols used by the PLC.

This adversary scenario is realistic for an attacker with network access but limited proprietary knowledge or advanced tools. As PLCpot does not provide real-time physical system data, the adversary cannot fully verify I/O consistency. Additionally, PLCpot is designed for interactions with standard engineering software rather than sophisticated scanning tools like Shodan, which could expose its limitations. The adversary's goal is to gain insights into control logic and possibly alter PLC operations to disrupt the process.

## 5.2 Cyber Attacks On PLCpot

We executed multiple ICS attacks on the PLCpot:

**Control Logic Injection Attack:** The control logic injection attack [29] involves downloading harmful control logic onto a PLC's memory, disrupting the physical process. The attacker uses engineering software or custom scripts to send "write" requests with malicious payloads.

**Control Logic Theft Attack (Reconnaissance):** This attack [12, 15] involves gathering information about a physical process by analyzing the control logic on a PLC. The attacker aims to develop customized attacks targeting the specific process. The reconnaissance phase can lead to sophisticated attacks, causing significant damage.

**Control Engine Attack:** A Control Engine Attack [19] targets the control engine executing the control logic on a PLC. The attacker sends a command to change the state of the PLC, disrupting the physical process.

**Brute Force Authentication:** We tested the data collection capability by performing a brute force authentication attempt using engineering software or Python scripts. The objective was to evaluate PLCpot's ability to detect, respond to, and log malicious access attempts.

## 5.3 Analysing the Forensic Artifacts

After executing attacks on PLCpot and capturing the communication, we analyzed the network dumps to identify the attacker's footprints. This analysis helped determine the attacker's methods and actions during the attack. The information can be used to reconstruct the attack and assess its impact on the system.

**Identifying Control Logic Injection and Reconnaissance Attacks**

We analyzed network dumps from the control logic injection and reconnaissance attacks using Eupheus, a control logic decompiler by Sushma et al. [12]. Eupheus converts control logic binaries to Instruction List format. We identified write request messages with function code '29' to extract the control logic binary from the network dump. To recognize control logic theft attacks, we looked for read function code '28' messages. Our analysis found 61 unique read messages, suggesting the attacker attempted to access the PLC's control logic.

**Detecting Control Engine Attack:** To detect control engine attacks, we developed a Python script that searches the PLCpot network dump for request messages where the attacker attempts to change the M221 PLC mode from start to stop. In the M221 PLC, function code '40' is used to start the PLC and '41' to stop the PLC. The script filters request messages accordingly.

$$\forall r \in Reqs, \quad \begin{cases} \text{PLC START}, & \text{if } r.\text{tcppayload}[9] = 40 \\ \text{PLC STOP}, & \text{if } r.\text{tcppayload}[9] = 41 \\ \text{Continue}, & \text{otherwise} \end{cases}$$

**Detecting Authentication Attempts:** We used a custom Python script to detect and analyze password authentication attempts on the PLCpot. During the function code extraction phase, we identified messages for M221 authentication. Authentication in M221 occurs in two steps: requesting a seed (function code '03') and sending the computed password hash(function code '6d'). We programmed the PLCpot to engage the attacker, responding to the initial authentication request with an old message and rejecting the authentication when the attacker sends the password hash(function code 'fd').

**Algorithm 3** Algorithm to detect PLC Authentication Attempt

```
r ← Reqs
if r.tcppayload[9] == "03" then
    Authentication Attempt
end if
if r.tcppayload[9] == "6D" then
    Password Hash
end if
```

## 6 LIMITATION AND FUTURE WORK

Although PLCpot is protocol-agnostic and offers many PLC operational and functional features, it has some limitations.

First, the diversity of network dumps determines the quality of dynamic fields identified, as PLCpot performs differential analysis on messages from various dumps. If a dynamic field's value doesn't change between two different dumps, it won't be labeled as dynamic. Therefore, to identify complete dynamic fields, diverse training data is essential.

Another limitation relates to the network dump used for replay. During template generation, PLCpot identifies function codes, their purposes, and associated request and response messages, storing this information in the PLC template. PLCpot can only send a successful response if the required information is present in the template and network dump being replayed. Thus, it is crucial to populate the template with as many function codes as possible and ensure the replayed network dump contains all required response messages.

Furthermore, while PLCpot employs several heuristics to ensure replies are as accurate as possible, there remains a possibility for sophisticated attackers to detect the honeypot. An adversary familiar with PLCpot's operation and communication protocol could craft specific requests designed to reveal discrepancies. For instance, an attacker might introduce a persistent change expecting it to manifest in the subsequent reply. While PLCpot covers control logic upload/download, authentication, and several PLC modes, an attacker could still attempt to detect the honeypot by making subtle changes that should appear in the next response. If PLCpot fails to accurately reflect these changes, it could expose its presence as a honeypot. However, if we capture these crafted messages and their responses from a real PLC, PLCpot can incorporate those messages into its database, preventing detection in future interactions.

Lastly, since PLCpot is not connected to a real physical system, it lacks continuous data from sensors or actuators like a real PLC. A highly knowledgeable attacker with an in-depth understanding of both the PLC protocol and the physical process—including the type of process, normal sensor data, actuator (I/O) data, etc might be able to detect PLCpot. This detection could occur because PLCpot only replies from its database of already captured messages, which may not fully mimic the real-time operational dynamics of an actual PLC system. To address this, PLCpot could be enhanced with an additional module that generates I/O data similar to a real physical process, which we plan to explore in future work.

## 7 CONCLUSION

As attacks on ICS systems continue to grow in both frequency and sophistication, it is imperative for the security community to understand attacker behavior and capabilities. In this paper, we introduce PLCpot, a scalable, protocol-agnostic honeypot. Our experimental results show that PLCpot outperforms existing state-of-the-art honeypots by providing application-level functionalities. Additionally, the PLC template generation feature of PLCpot underscores its scalability, allowing the security community to configure it to operate as various (out-of-the-box) PLCs. To illustrate PLCpot's effective engagement with attackers, we conducted a case study using a lab model of an elevator. Throughout this case study, we launched several attacks on the PLC, demonstrating that PLCpot engages effectively with attackers and collects data that can support forensic analysis and enhance threat insights.

## ACKNOWLEDGMENT

This work is partially supported by the National Science Foundation under Grant Award Numbers 2345563 and 2212424.

## REFERENCES

[1] Adeen Ayub. 2024. Stealthy Control Logic Attacks and Defense in Industrial Control Systems. *VCU Scholars Compass* (Aug 2024). https://scholarscompass.vcu.edu/etd/7817/

[2] Adeen Ayub, Wooyeon Jo, and Irfan Ahmed. 2024. Charlie, Charlie, Charlie on Industrial Control Systems: PLC Control Logic Attacks by Design, Not by Chance. In *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 182–193. https://doi.org/10.1109/HOST55342.2024.10545392

[3] Adeen Ayub, Wooyeon Jo, Syed Ali Qasim, and Irfan Ahmed. 2023. How Are Industrial Control Systems Insecure by Design? A Deeper Insight Into Real-World Programmable Logic Controllers. *IEEE Security Privacy* 21, 4 (2023), 10–19. https://doi.org/10.1109/MSEC.2023.3271273

[4] Adeen Ayub, Hyunguk Yoo, and Irfan Ahmed. 2021. Empirical Study of PLC Authentication Protocols in Industrial Control Systems. In *2021 IEEE Security and Privacy Workshops (SPW)*. 383–397. https://doi.org/10.1109/SPW53761.2021.00058

[5] Adeen Ayub, Nauman Zubair, Hyunguk Yoo, Wooyeon Jo, and Irfan Ahmed. 2023. Gadgets of Gadgets in Industrial Control Systems: Return Oriented Programming Attacks on PLCs. In *2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 215–226. https://doi.org/10.1109/HOST55118.2023.10132957

[6] Daniela Buza, Ferenc Juhász, György Miru, Márk Félegyházi, and Tamás Holczer. 2014. CryPLH: Protecting Smart Energy Systems from Targeted Attacks with a PLC Honeypot. In *International Workshop on Smart Grid Security*.

[7] T. M. Chen and S. Abu-Nimeh. 2011. Lessons from Stuxnet. *Computer* 44, 4 (2011), 91–93.

[8] 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

[9] Dragos and Dragos. 2022. CRASHOVERRIDE: Analyzing the malware that attacks power grids. https://www.dragos.com/resource/crashoverride-analyzing-the-malware-that-attacks-power-grids/

[10] Kevin E. Hemsley and Dr. Ronald E. Fisher. 2018. History of industrial control system cyber incidents. https://www.osti.gov/servlets/purl/1505628

[11] Arthur Jicha, Mark Patton, and Hsinchun Chen. 2016. SCADA honeypots: An in-depth analysis of Conpot. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*. 196–198. https://doi.org/10.1109/ISI.2016.7745468

[12] Sushma Kalle, Nehal Ameen, Hyunguk Yoo, and Irfan Ahmed. 2019. CLIK on PLCs! Attacking Control Logic with Decompilation and Virtual PLC. https://doi.org/10.14722/bar.2019.23074

[13] Ralph Langner. 2011. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Security Privacy* 9, 3 (2011), 49–51. https://doi.org/10.1109/MSP.2011.67

[14] 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, New York, NY, USA, 279–291. https://doi.org/10.1145/3372297.3423356

[15] Stephen McLaughlin and Patrick McDaniel. 2012. SABOT: Specification-Based Payload Generation for Programmable Logic Controllers *(CCS '12)*. Association for Computing Machinery, New York, NY, USA, 439–449. https://doi.org/10.1145/2382196.2382244

[16] Yassine Mekdad, Giuseppe Bernieri, Mauro Conti, and Abdeslam El Fergougui. 2021. A Threat Model Method for ICS Malware: The TRISIS Case. In *Proceedings of the 18th ACM International Conference on Computing Frontiers* (Virtual Event, Italy) *(CF '21)*. Association for Computing Machinery, New York, NY, USA, 221–228. https://doi.org/10.1145/3457388.3458868

[17] Syed Qasim, Muhammad Taqi Raza, and Irfan Ahmed. 2023. *vPLC: A scalable PLC testbed for IIoT security research*. https://www.acsac.org/2023/workshops/icss/syed-ali-qasim-paper.pdf

[18] Syed Ali Qasim. 2023. VIRTUAL PLC PLATFORM FOR SECURITY AND FORENSICS OF INDUSTRIAL CONTROL SYSTEMS. *VCU Scholars Compass* (2023). https://doi.org/10.25772/HWDW-CV61

[19] Syed Ali Qasim, Adeen Ayub, Jordan Johnson, and Irfan Ahmed. 2022. Attacking the IEC 61131 Logic Engine in Programmable Logic Controllers. In *Critical Infrastructure Protection XV*, Jason Staggs and Sujeet Shenoi (Eds.). Springer International Publishing, Cham, 73–95.

[20] Syed Ali Qasim, Wooyeon Jo, and Irfan Ahmed. 2023. PREE: Heuristic builder for reverse engineering of network protocols in industrial control systems. *Forensic Science International: Digital Investigation* 45 (2023), 301565. https://doi.org/10.1016/j.fsidi.2023.301565

[21] Syed Ali Qasim, Juan Lopez, and Irfan Ahmed. 2019. Automated Reconstruction of Control Logic for Programmable Logic Controller Forensics. In *Information Security*, Zhiqiang Lin, Charalampos Papamanthou, and Michalis Polychronakis (Eds.). Springer International Publishing, Cham, 402–422.

[22] Syed Ali Qasim, Jared M. Smith, and Irfan Ahmed. 2020. Control Logic Forensics Framework using Built-in Decompiler of Engineering Software in Industrial Control Systems. *Forensic Science International: Digital Investigation* 33 (2020), 301013. https://doi.org/10.1016/j.fsidi.2020.301013

[23] Stéphane Raimbault. [n. d.]. Libmodbus. https://libmodbus.org/

[24] Stéphane Raimbault. [n. d.]. Step7 Open Source Ethernet Communication Suite. https://snap7.sourceforge.net/

[25] Saranyan Senthivel, Shrey Dhungana, Hyunguk Yoo, Irfan Ahmed, and Vassil Roussev. 2018. Denial of Engineering Operations Attacks in Industrial Control Systems *(CODASPY '18)*. Association for Computing Machinery, New York, NY, USA, 319–329. https://doi.org/10.1145/3176258.3176319

[26] Yao Shan, Yu Yao, Tong Zhao, and Wei Yang. 2023. NeuPot: A Neural Network-Based Honeypot for Detecting Cyber Threats in Industrial Control Systems. *IEEE Transactions on Industrial Informatics* (2023), 1–10. https://doi.org/10.1109/TII.2023.3240739

[27] Susan Wade. 2011. *SCADA Honeynets: The attractiveness of honeypots as critical infrastructure security tools for the detection and analysis of advanced threats*. Ph. D. Dissertation.

[28] Feng Xiao, Enhong Chen, and Qiang Xu. 2017. S7commTrace: A High Interactive Honeypot for Industrial Control System Based on S7 Protocol. In *International Conference on Information, Communications and Signal Processing*.

[29] Hyunguk Yoo and Irfan Ahmed. 2019. Control Logic Injection Attacks on Industrial Control Systems. In *ICT Systems Security and Privacy Protection*, Gurpreet Dhillon, Fredrik Karlsson, Karin Hedström, and André Zúquete (Eds.). Springer International Publishing, Cham, 33–48.

[30] 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

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009