2023-12

IMPLEMENTING A HIGH-INTERACTION HYBRID HONEYPOT FOR FACILITY AUTOMATION SYSTEMS

Colvin, Scott D.

Monterey, CA; Naval Postgraduate School

https://hdl.handle.net/10945/72508

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IMPLEMENTING A HIGH-INTERACTION HYBRID HONEYPOT FOR FACILITY AUTOMATION SYSTEMS

by

Scott D. Colvin

December 2023

Thesis Advisor: Thuy D. Nguyen
Co-Advisor: Neil C. Rowe

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis
4. TITLE AND SUBTITLE IMPLEMENTING A HIGH-INTERACTION HYBRID HONEYPOT FOR FACILITY AUTOMATION SYSTEMS		**5. FUNDING NUMBERS**
6. AUTHOR(S) Scott D. Colvin		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.		
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.		**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Operational technology includes environments such as industrial control systems, building-automation systems, and transportation systems. With the rising trend of cyberattacks against these systems, operational technology needs better methods to increase security without costly redesigns of existing systems. We developed a high-interaction hybrid honeypot that uses reverse-proxy technology with commercial building-automation software and equipment to deceive attackers with real (not simulated) data. Our Web proxy monitors and intercepts malicious requests to manipulate target equipment, and deploys deceptive tactics such as sending fake HTTP acknowledgments and modifying webpages to include misleading information. Our results showed the effectiveness of this method in a controlled environment. This deception technique offers a new low-cost approach to defend building-automation systems in industries and the United States government, including the Department of Defense, from evolving cyber threats.

14. SUBJECT TERMS cyber deception, industrial control system, honeypot, facility automation, decoy, Web proxy, BACnet			**15. NUMBER OF PAGES** 55
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

IMPLEMENTING A HIGH-INTERACTION HYBRID HONEYPOT FOR FACILITY AUTOMATION SYSTEMS

Scott D. Colvin
Lieutenant, United States Navy
BS, University of Maryland, University College, 2017

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

from the

NAVAL POSTGRADUATE SCHOOL
December 2023

Approved by: Thuy D. Nguyen
Advisor

Neil C. Rowe
Co-Advisor

Alex Bordetsky
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Operational technology includes environments such as industrial control systems, building-automation systems, and transportation systems. With the rising trend of cyberattacks against these systems, operational technology needs better methods to increase security without costly redesigns of existing systems. We developed a high-interaction hybrid honeypot that uses reverse-proxy technology with commercial building-automation software and equipment to deceive attackers with real (not simulated) data. Our Web proxy monitors and intercepts malicious requests to manipulate target equipment, and deploys deceptive tactics such as sending fake HTTP acknowledgments and modifying webpages to include misleading information. Our results showed the effectiveness of this method in a controlled environment. This deception technique offers a new low-cost approach to defend building-automation systems in industries and the United States government, including the Department of Defense, from evolving cyber threats.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

BAS	Building-Automation System
FCU	Fan-Coil Unit
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HVAC	Heating, Ventilation, and Air Conditioning
ICS	Industrial Control System
IT	Information Technology
MITM	Man-in-the-Middle
OT	Operational Technology
TCP	Transmission Control Protocol
XML	Extensible Markup Language

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I sincerely appreciate the mentorship provided by Professor Nguyen and Professor Rowe during my thesis journey. Their support was crucial in navigating the challenges that significantly contributed to my understanding of scientific research and writing.

I would like to send a heartfelt thank you to my family and friends for their unwavering support. Their encouragement was my constant source of strength during this rewarding academic endeavor.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This chapter explains the motivation for this research, research methods, and thesis outline.

A. MOTIVATION

Operational technology (OT) is the term for programable systems and devices interacting with a physical environment (NIST, 2023). These environments include industrial control systems (ICS), building-automation systems (BAS), and transportation systems. Historically, these systems focused on availability and safety rather than security (Colbert et al., 2016). However, cyberattacks such as the 2015 BlackEnergy3 attack against Ukraine's power grid left a quarter-million people without power (Assante et al., 2016). The 2021 ransomware attack against the Colonial Pipeline caused fuel shortages and panic buying (Beerman et al., 2023).

With the rising number of cyberattacks, the operational technology industry needs fast and innovative methods to increase security without costly redesigns of current systems. Traditional software defense measures such as firewalls, intrusion-detection systems, and intrusion-prevention systems are insufficient in detecting and combating evolving ICS threats (Koay et al., 2023).

Hybrid honeypots with a deceptive proxy can significantly aid in operational-technology defense. Honeypots are security tools that attract attackers away from real systems, wasting their time and collecting data on their actions (Bythwood et al., 2023). Hybrid honeypots use at least some physical operational-technology equipment instead of simulated components, providing more realistic data. By adding a hybrid honeypot, the system can provide increased security to the honeypot and further waste an attacker's time.

B. RESEARCH METHODS

This thesis addresses the following hypothesis: Using building-automation equipment and its control software, a realistic decoy for an ICS device can attract malicious cyber actors while keeping them from changing the equipment. This thesis also addresses these research questions:

- RQ1: Can communication protocols between a human interface, deception proxy server, and building-control equipment be manipulated without causing system errors?
- RQ2: Can such a deception system be implemented and be undetected by an attacker?
- RQ3: Can such a deception system simulate enough functions to keep an attacker engaged for a while?

Our approach created a hybrid honeypot for a fan-coil unit (FCU) system that is a frequent part of commercial building-automation equipment. It used a transparent reverse proxy as an invisible network intermediary that intercepted and altered data packets between a client's Web browser and the ICS Web control server. It provided false information to attackers while leaving the genuine equipment unaffected.

We first examined the communication protocols between the ICS Web server and the human-machine interface (HMI) Web browser. We then designed and implemented a server-side proxy to track the connections and alter these messages. We tested the honeypot with simulated attacks to see if an attacker could detect the deception.

C. THESIS OUTLINE

Chapter II discusses ICSs and active defense using cyberdeception methods such as decoys and honeypots for an early warning of attacks. Chapter III describes the methods and design used in the research, including an analysis of communication protocols and the design of the proxy and honeypot. Chapter IV describes our implementation of the proxy's capabilities and testing of the hybrid honeypot. Chapter V reports the test results and assesses the ability of the proxy to alter data and remain undetected. Chapter VI summarizes the results and outlines future work.

II. BACKGROUND

This chapter provides background information on industrial control systems (ICSs), active defense, and cyberdeception.

A. INDUSTRIAL CONTROL SYSTEMS

ICSs are computer systems that monitor, manage, and control industrial processes in manufacturing, energy, transportation, and medical facilities. They differ from information technology (IT) systems in that real-world physical equipment is monitored by sensors and manipulated by controllers to activate physical "actuators." A category within ICSs is building-automation systems (BAS), which oversee the building environment, fire suppression, plumbing, elevators, and surveillance.

A frequently seen building-automation system is a heating, ventilation, and air conditioning (HVAC) system. It has at least these components:

1. Temperature Sensors: Thermostats read building temperatures, provide a human interface to change temperatures, and are often integrated with a controller that sends signals to activate heating or cooling units.
2. Air Conditioners: These systems have an evaporator, condenser, and compressor. They use a heat-transfer medium and circulate the heated or chilled medium to the coil unit where a fan blows air over it.
3. Fans: They draw air into the system, propel air across the coil, and subsequently distribute the cooled or heated air back to the building.
4. Controllers: These devices regulate and manage the system's operations to maintain temperature and airflow settings.

This research focused on the fan-coil unit (FCU), which combines the components above in a consolidated unit for an HVAC system. HVACs are not just needed for human comfort; they are needed in computer server rooms to prevent overheating or underheating, which can cause system instability, hardware damage, data loss, and building fires.

B. SERVER-SIDE PROXIES

Server-side proxies aid network security by acting as intermediaries between clients and servers, by balancing loads, caching content, and filtering content (DeJonghe, 2022). Well-known proxies are NGINX and Apache due to their reliable performance, open-source nature, and extensive documentation. MITMproxy is another open-source server-side proxy designed to intercept traffic and alter data packets, making it ideal for simulating and studying man-in-the-middle attacks (Cortesi et al., 2023). MITMproxy's flexible scripting and interception features enable deceptive environments that mimic client and server interactions, enabling research on cyberdeception techniques.

C. CYBERDEFENSE

Passive cyberdefense monitors and analyzes network activities to identify potential threats without engaging adversaries. Common tools include intrusion-detection systems, intrusion-prevention systems, firewalls, security-information event management (SIEM), encryption, and vulnerability scans (Senol, 2022). These analyze incoming traffic and events to detect anomalies or attack patterns. Since passive strategies only collect data for analysts to make decisions, they cannot support quick responses to new threats. Moreover, these tools risk misinterpreting new activity as a compromise, making them susceptible to deception or false positives.

Active cyberdefense is more proactive as it involves engagement with threats, aiming to find, disrupt, or deny their activities. Active defense includes threat hunting (patrolling a network), threat-intelligence sharing, dynamic access controls, and deception technologies (Senol, 2022). Active-defense methods allow defenders to respond more quickly, minimizing potential damage and shortening the attacker's window of opportunity.

D. CYBERDECEPTION

Cyberdeception can be used either offensively (to attack someone) or defensively (to defend against attacks) (Rowe & Rrushi, 2016). Defensive cyberdeception can mislead malicious actors with false data or false environments. Examples are decoys, honeypots, and honeynets, which can divert attackers from real systems. Dropping or altering network data can confuse attackers while gathering intelligence about them and wasting their time. A decoy can also be a false file, misleading database record, fake user credential, or dummy device.

Cyberdeception can be made more elaborate with "deceptive software engineering." This concept focuses on delivering customized feedback to a malicious cyber actor. Common techniques include deceptive firewalls, software wrappers that intercept software calls to enable deceptive responses, and honeypots or honeynets that emulate systems or networks. Finally, custom-coded deception software can deliver unique information to minimize the attacker's ability to detect the deception.

E. HONEYPOTS

Honeypots are one good way to implement many deception techniques. They mimic real systems or services within a network but have no purpose other than to collect attack data (Franco et al., 2021). Honeypots can be isolated from a network by being placed in front of firewalls or hidden on a network as legitimate services (Razali et al., 2018).

Low-interaction honeypots are simple to set up, easy to maintain, require few network resources, and can provide early detection and alerting of threats. However, these honeypots are easy to detect, do not waste attackers time, and do not provide research's much intelligence gathering on the attacks. High-interaction honeypots emulate complex systems and provide more features and deeper interaction options. Such honeypots let researchers gather more data on adversary activities for forensic analysis and intelligence.

A relatively new concept is the hybrid honeypot, which uses physical components and commercial software within the honeypot, instead of relying entirely on simulation or virtualization (Franco et al., 2021). A hybrid technique generates realistic data and reduces potential signatures that tend to give away the presence of deterministic honeypots. A hybrid honeypot can provide real-world feedback about the environment by allowing the attacker to interact with physical equipment, as by turning off real fans and then seeing a temperature rise that triggers the opening of cooling valves.

F. COMMUNICATION PROTOCOLS AND LANGUAGES

The Hypertext Transfer Protocol (HTTP) communicates between Web browsers and servers (Fielding et al., 2022). It transfers text and pictures across the Internet. When a user requests a Web page with a uniform resource locator (URL) such as https://nps.edu, the browser sends a GET request to the hosting server and receives an acknowledgement and some HTML data. The common commands are GET, POST, HEAD, and PUT.

HTML, the primary language for Web pages, uses tags to delimit elements such as links, images, headings, and text (Berners-Lee et al., 1995). HTML structure includes a root <html> tag, a <head> component, and a <body> component. The <head> component contains metadata like titles and resource links, while the <body> component identifies visible content for the user interface.

XML (Extensible Markup Language) is a language for structuring data (Hollenbeck et al., 2003). Although visually resembling HTML, XML uses user-defined tags instead of predefined tags as in HMTL. User-defined tags organize data for specific applications.

The "publisher and subscriber" model enables persistent communication between two entities. Asynchronous messaging enables a process to send a message to a message broker to disseminate it to subscribed entities seeking that data (Saxena et al., 2018). If subscribers are temporarily inaccessible, the message broker stores it until successful delivery is possible. This strategy can relieve network congestion by combining messages and disseminating the combinations to all subscribers.

G. RELATED WORK ON HYBRID HONEYPOTS

The 2015 GasPot Experiment analyzed the security of gas-tank monitoring systems by setting up honeypots, called GasPot, to attract cyber-attacks (Wilhoit et al., 2015). Attacks revealed security vulnerabilities that could lead to fuel theft or fuel-supply disruption.

A research project proposed a framework to create honeypots that can model physical processes and devices called HoneyPhy (Litchfield, 2017). They created a prototype of an HVAC honeypot with simulated devices running in Docker containers (Docker, n.d.). No integration with real components occurred as in our research here.

Another project used hybrid honeypots and associated deception technologies to gather threat intelligence (Haney, 2019). They proposed criteria for assessing a honeypot including realism, scalability, and detection resistance. They sought more authentic honeypot data by including real programmable logic controllers (PLCs) within a simulated ICS environment. Within 19 minutes of its online deployment their honeynet was probed by attackers. By day two of deployment, Shodan had scanned their honeynet's ports, performed fingerprinting, and concluded it had an ICS port open but not that it was a honeypot. This showed that attackers could quickly find industrial control systems without performing active reconnaissance. The primary limitation of this work was its simulated environment; however, acquiring equipment to create a hybrid environment can be costly.

Another hybrid honeypot project was the HoneyVP project (You et al., 2021). It proposed evaluation criteria of fidelity, scalability, and cost; fidelity is the degree of realism shown by the honeypot environment. This work addressed costs with a semi-virtual and semi-physical ICS honeypot architecture. In this architecture, attackers predominantly work with the virtual environment. The honeypot primarily relied on recorded cyberattack traffic, with the physical components invoked only during unfamiliar interactions or requests. While interactions within the virtual environment yield swift responses, the transition to the physical device through a reverse SSH tunnel could introduce a noticeable delay in response time that could be suspicious to an attacker. This approach differs from ours because we used physical equipment and commercial software without virtualization.

Another group of researchers studied the data collection capability of cloud-based honeypots compared to local honeypots (Ivanova et al., 2023). In a 65-day data collection period, they discovered that local deployments collected more relevant data than cloud-based deployments.

A recent hybrid honeypot detection project studied preserving honeypot resources by screening incoming connections (Bythwood et al., 2023). A screening proxy distinguished botnet attacks from advanced attacks, and routed the connection to low-interaction or high-interaction honeypots. The proxy used scoring information such as ports, operating system, IP address reputation, and HASSH values; HASSH identifies server secure shell (SSH) implementations (Salesforce, n.d.). Incoming connections are compared to the blocking list, and the likelihood of the connection being a botnet, spyware, malware, or a Web-crawling spider is determined. Like our project, they implemented a screening proxy in front of the honeypot using an IP address list. Their proxy redirects connections to either low-interaction or high-interaction honeypots, while our proxy screens all connections and only routes legitimate connections to the real FCU system.

III. METHODS AND DESIGN

This chapter discusses the hardware and software used to implement the hybrid honeypot, the high-level design of the honeypot, and the control-unit proxy requirements.

A. SYSTEM UNDER TEST

1. Physical Environment

We used a fan-coil unit (FCU) prototype from prior work. It includes an Automated Logic BACnet/IP router (Automated Logic, n.d. a), an Automated Logic BACnet controller (Automated Logic, n.d. b), an actuator, a fan motor, a temperature sensor, and a thermostat (Figure 1). The motor, actuator, temperature probe, and thermostat connect to the controller and communicate using analog signals. The controller compiles data from these devices and sends the data to the router, which then forwards them to the building-automation Web server using BACnet-over-IP protocols (ANSI, 2020). Figure 2 shows the FCU logical layout.

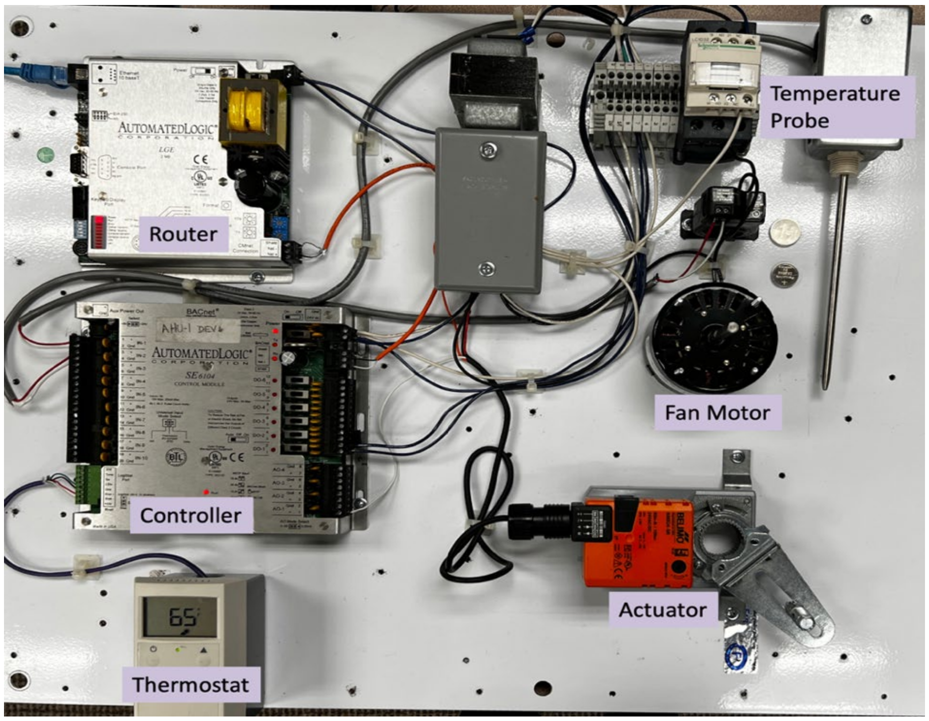


Figure 1. Hybrid honeypot components

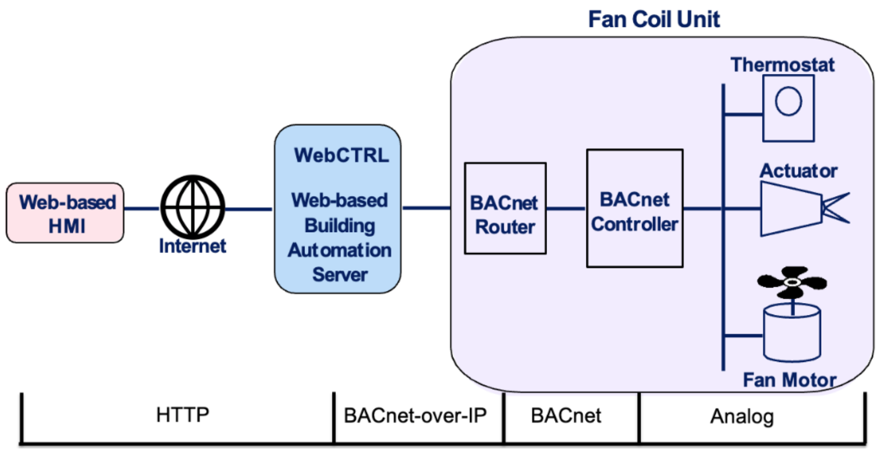


Figure 2. Fan-coil unit logical layout

2. WebCTRL

The FCU equipment is managed by the Automated Logic WebCTRL building-automation software (Automated Logic, 2019), which includes a Web server that allows remote users to monitor and control equipment using a Web browser. A sample view of the graphical user interface showing the status of the FCU equipment is in Figure 3.

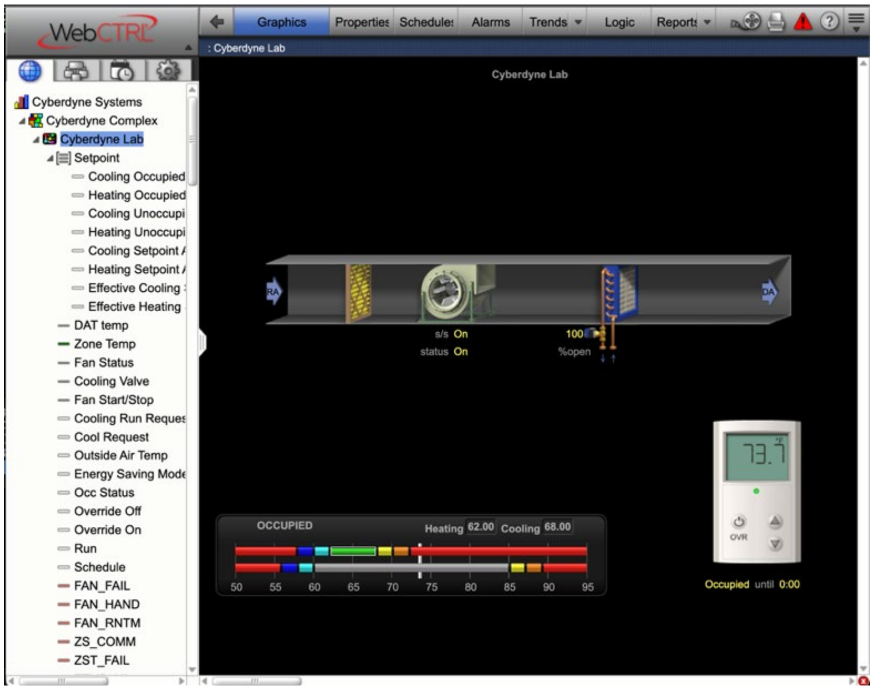
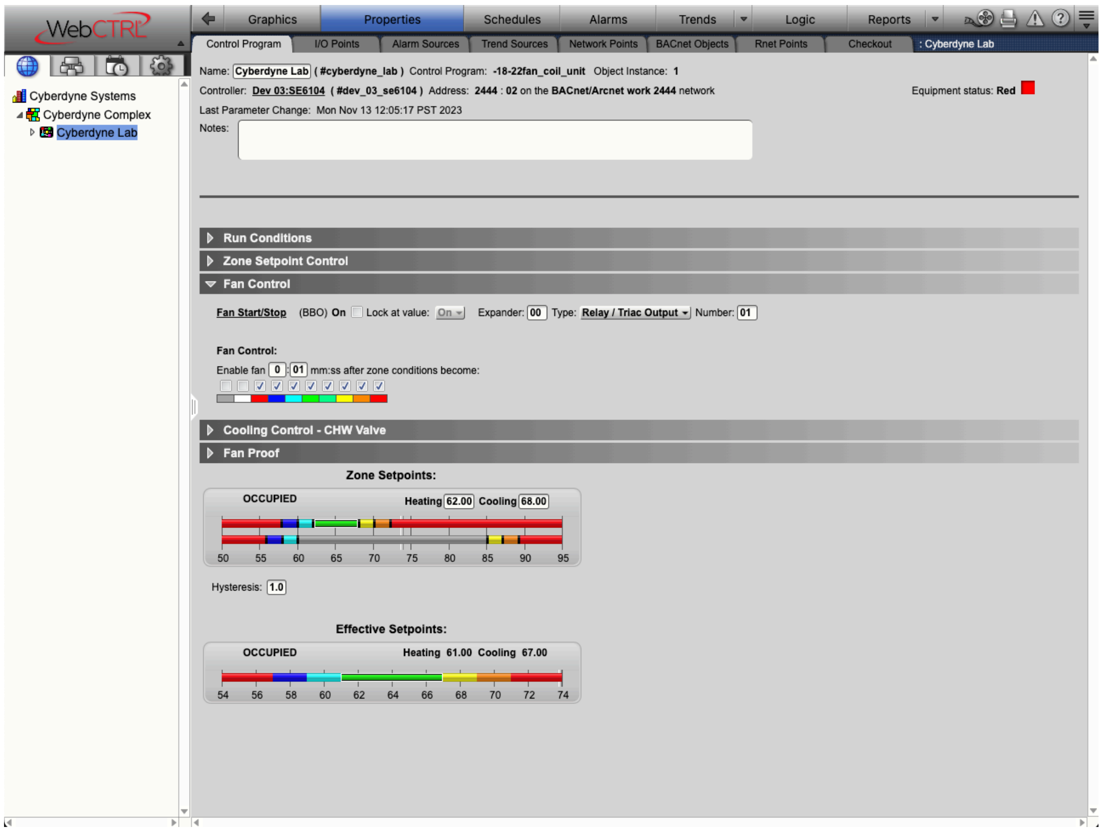


Figure 3. Sample WebCTRL graphics Web page

WebCTRL provides several webpages to control the FCU equipment. This research focused on the “properties control” and “equipment logic” categories that control physical equipment. Figure 4 shows the “control program” page in the “properties control” category. This webpage allows changes to the running conditions, temperature setpoints, fan power, and cooling-valve position.



Name: Cyberdyne Lab (#cyberdyne_lab) Control Program: -18-22fan_coil_unit Object Instance: 1

Controller: Dev 03:SE6104 (#dev_03_se6104) Address: 2444 : 02 on the BACnet/Arcnet work 2444 network   Equipment status: Red 🟥

Last Parameter Change: Mon Nov 13 12:05:17 PST 2023

Notes:

Fan Start/Stop (BBO) On ☐ Lock at value: On Expander: 00 Type: Relay / Triac Output Number: 01

Fan Control:

Enable fan 0 : 01 mm:ss after zone conditions become:

☐ ☐ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

Zone Setpoints:

OCCUPIED   Heating 62.00 Cooling 68.00

50 55 60 65 70 75 80 85 90 95

Hysteresis: 1.0

Effective Setpoints:

OCCUPIED   Heating 61.00 Cooling 67.00

54 56 58 60 62 64 66 68 70 72 74

Figure 4. WebCTRL properties Web page

Figure 5 shows one part of the ladder-logic program to control equipment in the hybrid honeypot. The user sees timing, alarms, power, equipment-run conditions, and smaller components such as power switches.

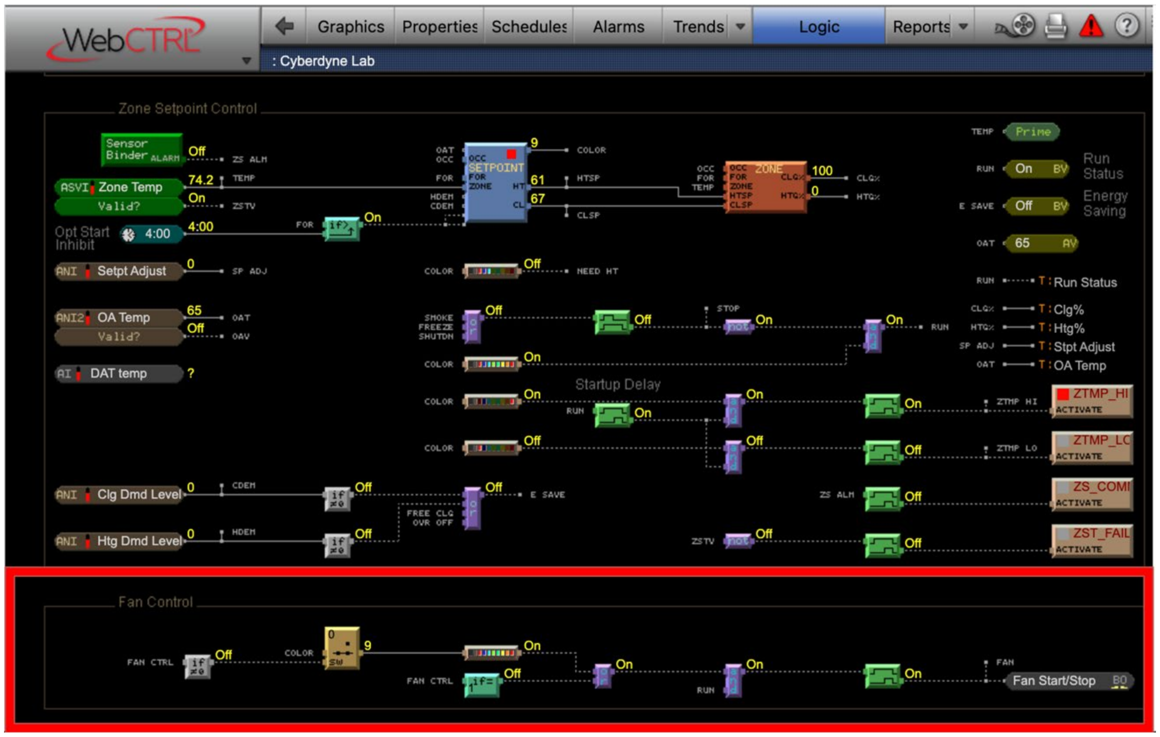
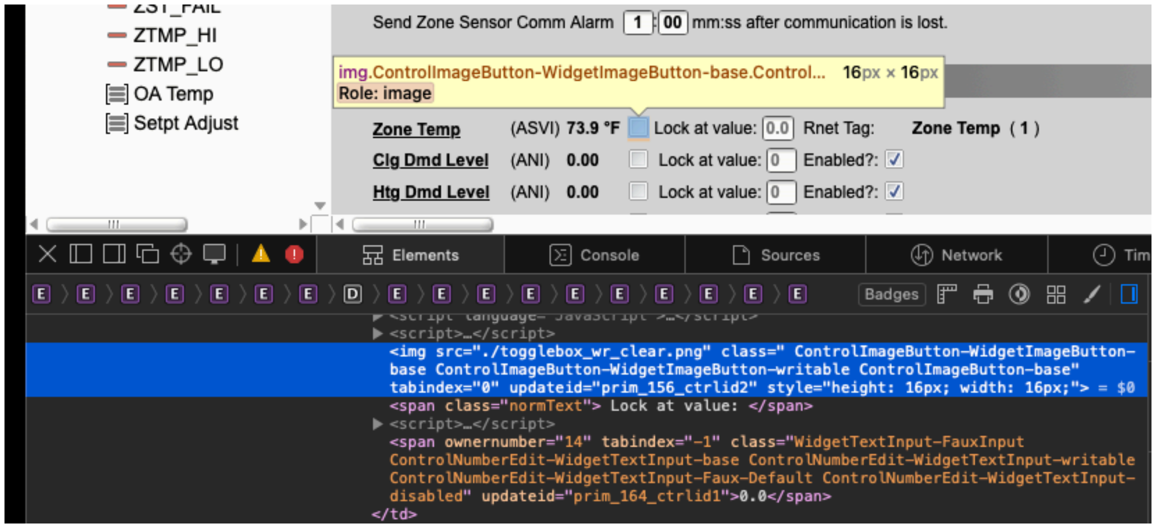


Figure 5. WebCTRL ladder logic Web page

3. Communications Protocols

A remote user can control the FCU system using a Web browser which communicates with the WebCTRL Web server using HTTP over port 8080. WebCTRL uses HTML and cascading style sheets for constructing webpages. After the user logs in and the main website loads, the Web browser begins sending three types of POST messages to the server every 4–6 seconds using a subscriber-and-publisher client-server messaging model. We discovered that a POST message containing the word "subscriber" in the message body is polling for equipment updates from WebCTRL; a POST message with "publisher" in the body is submitting an equipment change to WebCTRL. In both cases, the WebCTRL server's HTTP 200 acknowledgment response includes all equipment updates since the previous HTTP 200 response message. This thesis focuses on the equipment-change message.

We discovered that each changeable item on a Web page has a unique primitive identifier. The format of these identifiers is prim_xxx, where xxx is a numeric value; we call them prim_IDs. We used a Web browser's inspection tool to analyze the use of the prim_IDs of different items. Figure 6 shows an example prim_ID assigned to a temperature-override check box.



Send Zone Sensor Comm Alarm 1 : 00 mm:ss after communication is lost.

img.ControlImageButton-WidgetImageButton-base.Control... 16px × 16px
Role: image

Zone Temp	(ASVI)	73.9 °F	▢	Lock at value:	0.0	Rnet Tag:	Zone Temp (1)
Clg Dmd Level	(ANI)	0.00	☐	Lock at value:	0	Enabled?:	✓
Htg Dmd Level	(ANI)	0.00	☐	Lock at value:	0	Enabled?:	✓

```
▶ <script language="JavaScript">…</script>
▶ <script>…</script>
  <img src="./togglebox_wr_clear.png" class=" ControlImageButton-WidgetImageButton-
  base ControlImageButton-WidgetImageButton-writable ControlImageButton-base"
  tabindex="0" updateid="prim_156_ctrlid2" style="height: 16px; width: 16px;"> = $0
  <span class="normText"> Lock at value: </span>
▶ <script>…</script>
  <span ownernumber="14" tabindex="-1" class="WidgetTextInput-FauxInput
  ControlNumberEdit-WidgetTextInput-base ControlNumberEdit-WidgetTextInput-writable
  ControlNumberEdit-WidgetTextInput-Faux-Default ControlNumberEdit-WidgetTextInput-
  disabled" updateid="prim_164_ctrlid1">0.0</span>
</td>
```

Figure 6. Browser inspection tool shows prim_156 as the prim_ID for the blue-highlighted check box

Once the user changes a webpage item and clicks the accept button, the browser sends a user-requested change message with the prim_ID of the item and the new equipment status (on, off, true, false, 1, or 0) to WebCTRL. The WebCTRL server makes the requested changes, updates the equipment status, and includes the new status in the HTTP 200 acknowledgment response to a Web browser's POST message. After the Web browser receives the change acknowledgment message, the browser returns to sending a subscriber message every four seconds to poll for new updates.

B. DECEPTION OVERVIEW

The purpose of our deceptive FCU proxy was to intercept any change request made by a suspected malicious user and imitate a server response. This causes a malicious user to incorrectly believe the server implemented the change. Figure 7 shows the high-level design of the hybrid honeypot and placement of the FCU proxy.

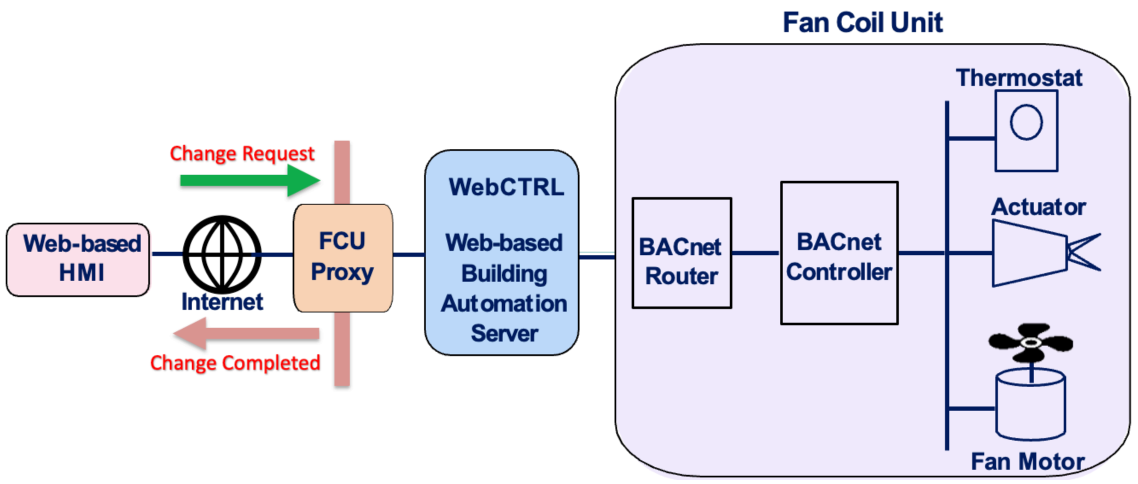


Figure 7. Design of FCU hybrid honeypot

1. Required Proxy Capabilities

We used an open-source server-side proxy to implement the FCU proxy. Our criteria for good proxy software are:

1. Transparency: Be able to operate as a passive node on the network.
2. Deep packet inspection: Be able to analyze network traffic at the application layer with a primary focus on HTTP and XML.
3. Multi-connection handling: Be able to process simultaneous network connections as the proxy must manage malicious and authorized connections.
4. Live packet alteration: Be able to manipulate or replay live HTTP packets in transit to a Web browser.
5. Logging: Be able to implement diverse logging points to ensure data redundancy if the hybrid honeypot is compromised.
6. Usability: Be able to verify its provenance, be able to actively maintain its software and documentation, and demonstrate ease of use (requiring minimal experience to operate).
7. Overhead: Use minimal computational resources.

8. Custom scripting: Provide a programming interface to run custom scripts.

2. Selection of Proxy Technologies

We examined man-in-the-middle proxy technologies BeEF (BeEF, n.d.), BetterCAP (BetterCAP, n.d.), Burp Suite (PortSwigger, n.d.), Ettercap (Ettercap, n.d.), NGINX (NGINX, n.d.), MITMproxy (MITMproxy, n.d.), and OWASP ZAP (ZAP, n.d.). Table 1 summarizes our assessment of them. BetterCAP, Ettercap, and Burp Suite are capable man-in-the-middle software. However, MITMproxy focuses on HTTP interception, has a lower overhead, is accompanied by detailed documentation, and is easy to use. Hence, we chose to use MITMproxy for our project.

Table 1. Applications supporting the required capabilities

	Transparency	Deep Packet Inspection	Live Packet Alterations	Multi-connection Handling	Logging	Usability	Overhead	Custom Scripting
BeEF	Yes	No	No	Yes	Yes	Moderate	Moderate	Yes
OWASP ZAP	Yes	Yes	No	Yes	Yes	Moderate	Low	Yes
Burp Suite	Yes	Yes	Yes	Yes	Yes	Moderate	Moderate	Yes
Ettercap	Yes	Yes	Yes	Yes	Yes	Moderate	Low	Limited
BetterCap	Yes	Yes	Yes	Yes	Yes	Easy	Moderate	Yes
NGINX	Yes	No	No	Yes	Yes	Easy	Low	Limited
MITMproxy	Yes	Yes	Yes	Yes	Yes	Easy	Low	Yes

C. FCU PROXY REQUIREMENTS

The FCU proxy must meet the following requirements:

1. It must determine if TCP connections and user-login attempts are malicious.

2. It must analyze HTTP packets to identify user-submitted changes and prevent them from reaching the server.

3. It must reply to user-submitted change requests with fake webpages and a success status within the normal response time.

IV. DECEPTION IMPLEMENTATION

This chapter describes the implementation of the FCU proxy and the test environment.

A. FCU PROXY

The FCU proxy is a transparent reverse proxy written in the Python language. It intercepts all traffic between a Web browser and the WebCTRL server at port 8080.

1. Detection of Malicious Connection

The proxy maintains a list of authorized external IP addresses; all others are considered malicious. When a browser establishes a TCP connection to WebCTRL, if the source IP address is not authorized, the proxy forwards the packet to WebCTRL but flags the connection as malicious. Also, the proxy continually monitors all connections for failed authorization messages from the WebCTRL server. The connection is considered malicious if a client device makes three failed login attempts to WebCTRL using the same username (Figure 8). If 10 consecutive failed login attempts are made for a specific username, WebCTRL will lock that user account for a configurable lock-out period.

The proxy can read the username and password submitted to WebCTRL as the username is sent in plain text, and the password is obfuscated using a simple obfuscation algorithm. WebCTRL enforces a password policy that prevents easily guessed usernames and passwords such as <root, password123>. Although WebCTRL can be configured to not require user authentication, doing so would make the system look like a honeypot.

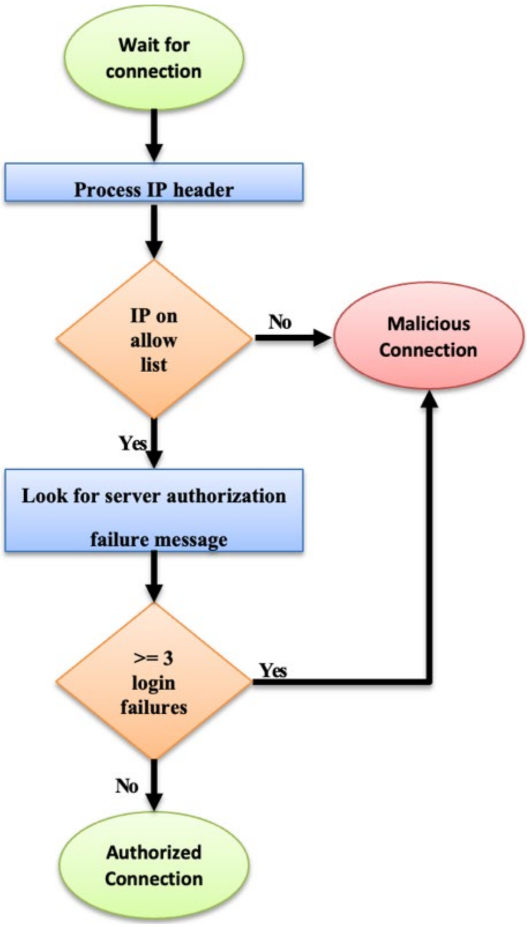


Figure 8. Connection verification logic

2. Change Request Detection and Interception

After a successful login, the proxy reads all server responses to the client's requests and caches the current conditions of webpage items being monitored for deception in Python global variables. For this work, those items are the fan (on or off) and the manual override option (enabled or disabled). The cached conditions determine how the FCU proxy reacts to change requests.

The FCU proxy looks for change requests in POST messages by identifying the text string "publisher" in the XML message. The "publisher" string indicates that the Web browser has submitted a change request to the server. The program parses the rest of the XML message, looking for the prim_IDs associated with the monitored webpage items. The proxy lets the packet through if the monitored prim_IDs are not in the XML message.

Otherwise, the proxy stops the POST message from reaching the server, and replies with a fake message imitating the server's response (Figure 9). Message interception is a feature of MITMproxy, permitting the real-time manipulation of data packets in transit. The proxy uses MITMproxy's Python application-programming interface to implement the fake responses.

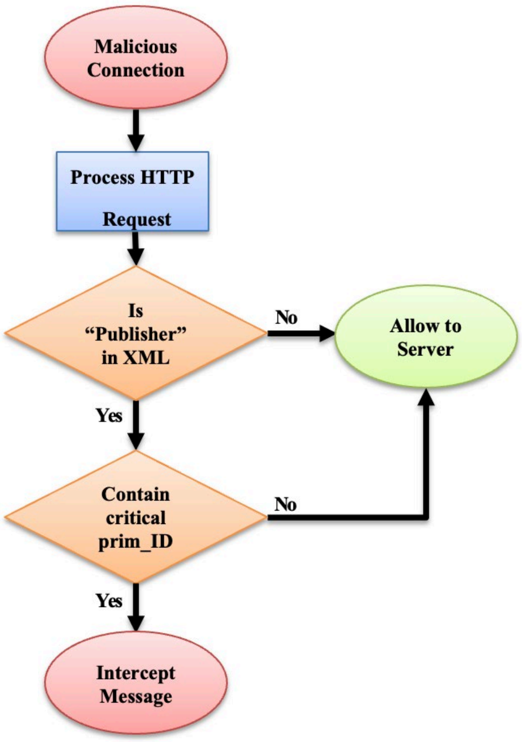


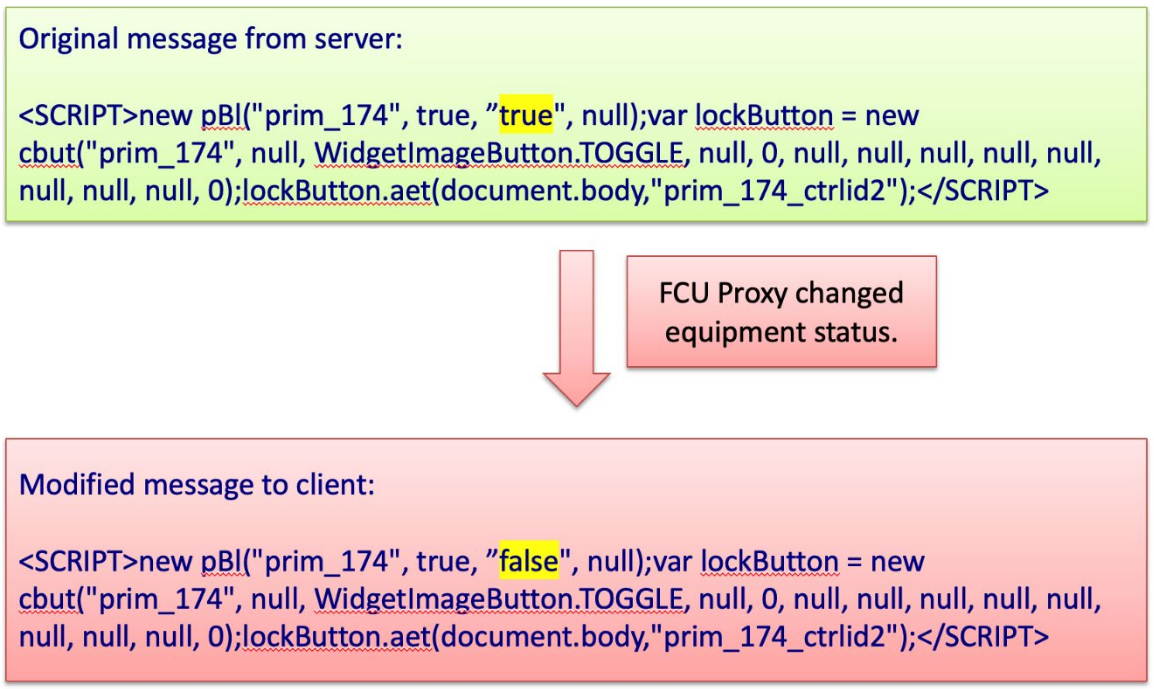
Figure 9. Change request tracking and message interception logic

3. Deceptive Message Reply

The proxy generates deceptive messages using the cached condition of the equipment that the server sends to the browser after a successful login. The proxy tracks the deceptive states of the monitored equipment in Python global variables. The proxy tracks malicious connections, so if a malicious user logs out and logs back in with the same IP address, they will return to the deceptive state they left.

If a POST message is intercepted for deception, the proxy looks for the prim_ID and the new equipment status in the XML message. The proxy then updates the Python global variables to record the new equipment status. Next, the proxy drops the packet, builds a fake server response using the WebCTRL server's HTTP header, the prim_ID, and fake equipment status, and sends the fake response back to the client. Once the client's browser receives this fake server response, the browser shows the spoofed change.

As the user navigates through the WebCTRL website or refreshes the browser, the proxy reads the HTML messages from the server. If the received HTML messages have an equipment status different from the global variables, the proxy will change the HTML enroute to reflect the equipment's deceptive state. For example, if the HTML message indicates prim_174 as on and the deceptive state of prim_174 is off, the proxy will modify the HTML message to indicate prim_174 is off. Figure 10 shows an example of what the server would send as part of a webpage and what the client receives.


Original message from server:

```
<SCRIPT>new pBl("prim_174", true, ”true", null);var lockButton = new
cbut("prim_174", null, WidgetImageButton.TOGGLE, null, 0, null, null, null, null, null,
null, null, null, 0);lockButton.aet(document.body,"prim_174_ctrlid2");</SCRIPT>
```

Modified message to client:

```
<SCRIPT>new pBl("prim_174", true, ”false", null);var lockButton = new
cbut("prim_174", null, WidgetImageButton.TOGGLE, null, 0, null, null, null, null, null,
null, null, null, 0);lockButton.aet(document.body,"prim_174_ctrlid2");</SCRIPT>
```

Figure 10. Server response modified by FCU proxy

B. TEST ENVIRONMENT

The test environment had two clients, an FCU proxy, a switch, a WebCTRL server, and HVAC equipment, as shown in Figure 11. Two clients tested simultaneous connections between a malicious client using the Chrome Web Browser and an authorized client using the Firefox Web browser. The goal was to demonstrate that the FCU proxy correctly identified the malicious connection, detected change requests, intercepted malicious messages, sent fake responses to the malicious browser, modified the websites as they downloaded while remaining undetected. The proxy logs all malicious activities.

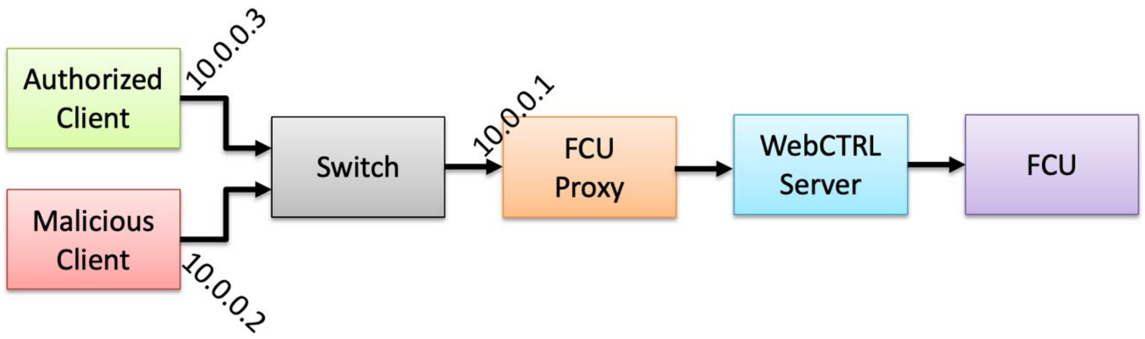


Figure 11. Test environment

1. Test Case 1: Malicious Connection Detection

Test Case 1 verifies that the proxy correctly identifies a malicious connection when 10.0.0.3 is the only authorized IP address. It has three steps:

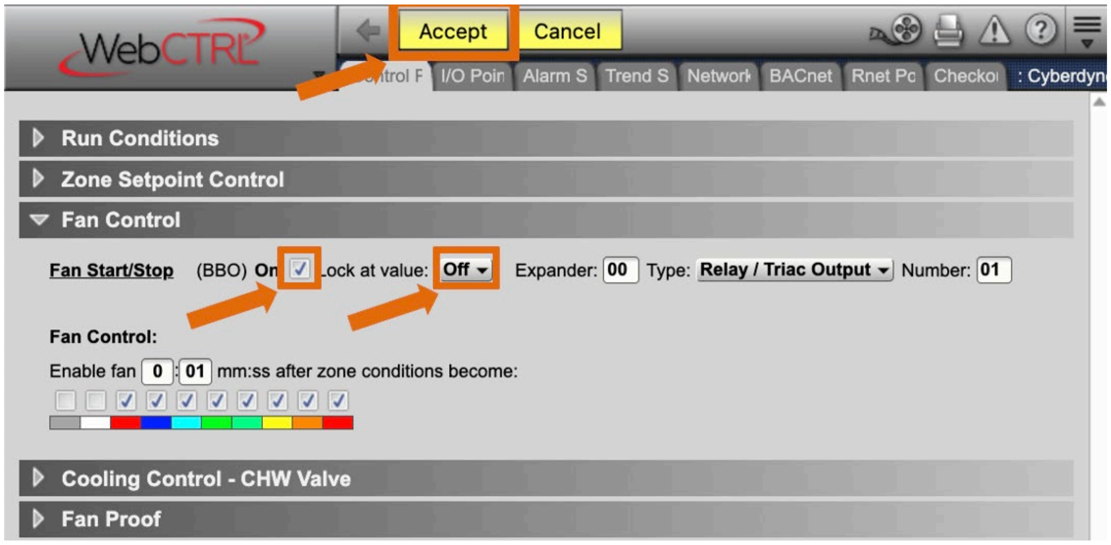
1. Reset all connections.
2. The browser on the malicious client connects to WebCTRL at 10.0.0.1::8080. The proxy should immediately flag 10.0.0.2 as malicious.
3. The authorized client fails to log in three times. The proxy should identify a brute-force login attempt using an approved IP address.

To pass the test, the FCU proxy must display the alerts in the proxy's logging terminal.

2. Test Case 2: Change Detection, Interception, and Deceptive Response

This test verifies that the proxy correctly detects and intercepts malicious change requests, and generates the corresponding fake responses. The steps are:

1. Reset all connections and ensure that the physical fan is on.
2. Both authorized and malicious clients log in to WebCTRL with the correct usernames and passwords. The 10.0.0.2 connection should be flagged as malicious since it is not in the list of authorized IP addresses.
3. Both clients navigate to the Control page (Figure 12) and the proxy should save the equipment statuses returned in the webpages.
4. The authorized client makes an equipment change. The server performs the requested change and publishes this change to all connected users. The malicious user should receive this change. However, no logging will occur because the proxy only flags malicious actions.
5. The malicious user requests to turn off the fan from the Control page (Figure 12). The requested change should not make it to the server, and the FCU proxy should generate an alert of the change request. The authorized user should observe no change to the equipment, and the physical fan should remain on.



Fan Start/Stop (BBO) On ✓ Lock at value: Off Expander: 00 Type: Relay / Triac Output Number: 01

Fan Control:
Enable fan 0 : 01 mm:ss after zone conditions become:

Figure 12. Control webpage, turn off the fan

1. The malicious user turns the fan back on and navigates to the Logic page (Figure 5). The FCU proxy should generate an alert of the change request.

2. The malicious user requests to turn off the fan from the Logic page (Figure 13). The requested change should not make it to the server, and the FCU proxy should trigger another alert. The authorized user should observe no change to the equipment, and the physical fan should remain on.

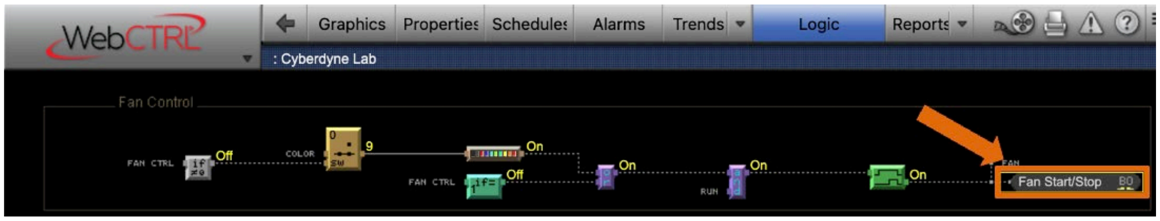


Figure 13. Fan start/stop button on Logic page

To pass the test, the FCU proxy must display the alerts in the proxy's logging terminal.

3. Test Case 3: Deceptive Web Interface

This test verifies that the proxy correctly intercepts change requests, tracks the deceptive state of the fan, and modifies the webpages received from WebCTRL. The steps are:

1. Repeat steps 1–2 of Test Case 2.
2. The malicious user requests to turn off the fan from the Control page (Figure 12). The malicious browser should receive a deceptive message from the FCU proxy saying that the fan is off. If the FCU proxy correctly processed the deceptive message and response, the Web browser should not display errors on the webpages. Possible errors include XML messages indicating that the fan is not off or browser alert messages such as unable to complete requests or reach the server.
3. The malicious and authorized users should navigate to the Graphics page (Figure 3) to visually verify the fan's operating condition. The malicious user should see that the fan is off, while the authorized user should see that the fan is running. The physical HVAC fan should stay on during these tests.
4. The malicious user navigates back to the Control page where the fan was turned off and verifies that the page loaded with the fake fan status indicating that the fan is off. The fake status displayed on the webpage demonstrates that the proxy correctly maintains state and modifies intercepted webpages accordingly.

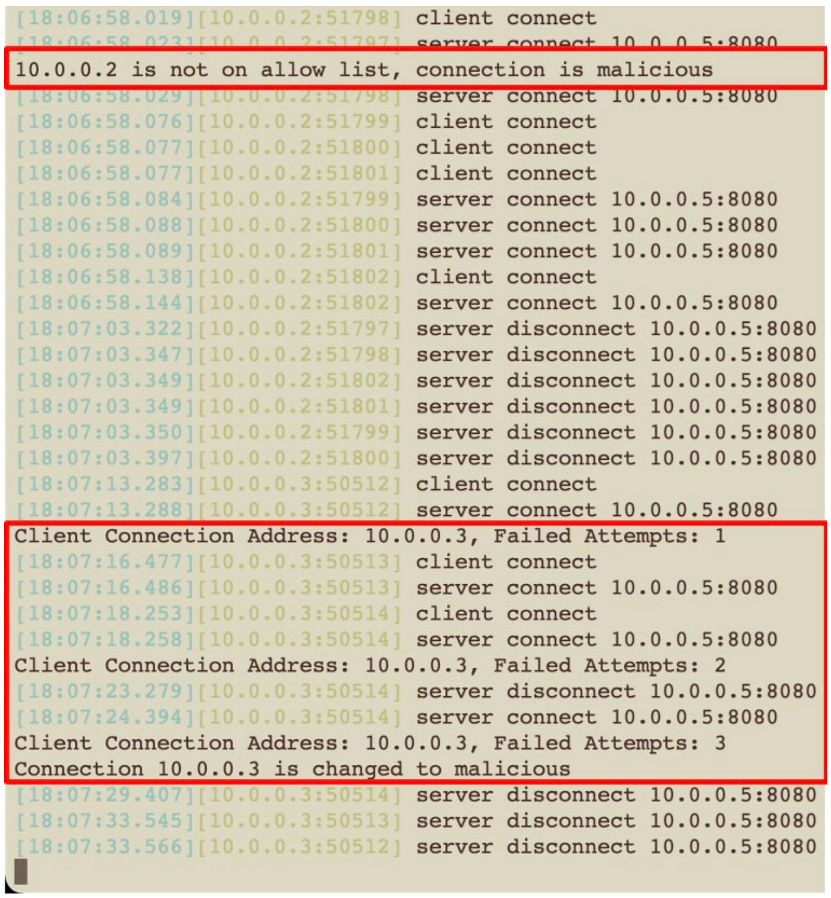
To pass the test, the proxy must return webpages with correct deceptive information to the malicious user and normal webpages to the authorized user. The returned webpages displayed by the malicious browser must also show no errors.

V. RESULTS

This chapter describes the results of the FCU proxy test.

A. TEST CASE 1 – MALICIOUS CONNECTION DETECTION

Test Case 1 was successful. The FCU proxy correctly detected malicious connections and tracked a brute-force login attempt. When the malicious client (with an IP address ending in 0.2) connected to the WebCTRL server, the connection was flagged by the FCU proxy as malicious as it was not on the allowed IP list (Figure 14). Although the authorized client (ending in 0.3) was on the allowed list, the connection was flagged as malicious because the client failed to log in three times (Figure 14).



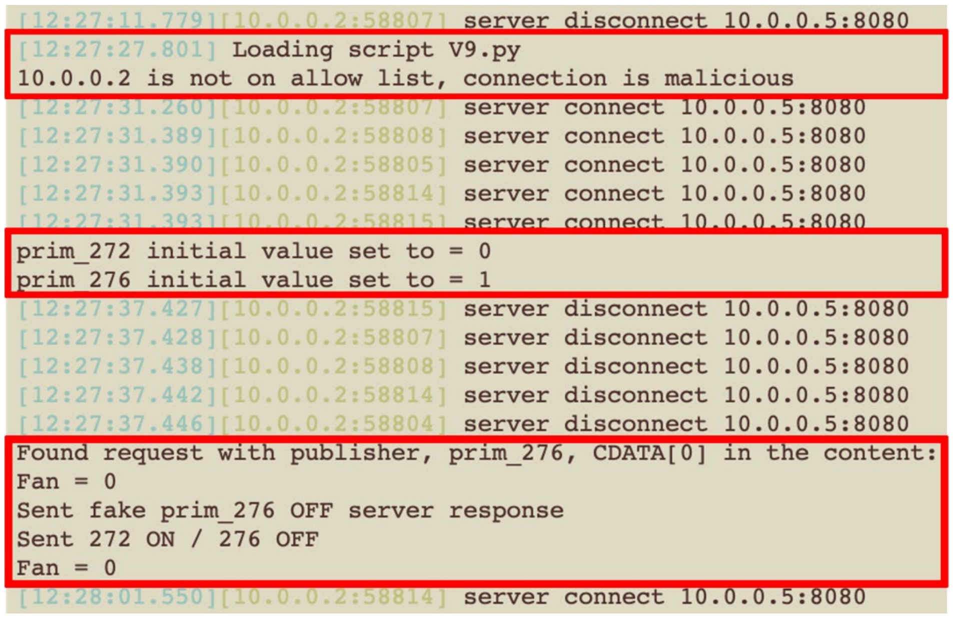
```
[18:06:58.019][10.0.0.2:51798] client connect
[18:06:58.023][10.0.0.2:51797] server connect 10.0.0.5:8080
10.0.0.2 is not on allow list, connection is malicious
[18:06:58.029][10.0.0.2:51798] server connect 10.0.0.5:8080
[18:06:58.076][10.0.0.2:51799] client connect
[18:06:58.077][10.0.0.2:51800] client connect
[18:06:58.077][10.0.0.2:51801] client connect
[18:06:58.084][10.0.0.2:51799] server connect 10.0.0.5:8080
[18:06:58.088][10.0.0.2:51800] server connect 10.0.0.5:8080
[18:06:58.089][10.0.0.2:51801] server connect 10.0.0.5:8080
[18:06:58.138][10.0.0.2:51802] client connect
[18:06:58.144][10.0.0.2:51802] server connect 10.0.0.5:8080
[18:07:03.322][10.0.0.2:51797] server disconnect 10.0.0.5:8080
[18:07:03.347][10.0.0.2:51798] server disconnect 10.0.0.5:8080
[18:07:03.349][10.0.0.2:51802] server disconnect 10.0.0.5:8080
[18:07:03.349][10.0.0.2:51801] server disconnect 10.0.0.5:8080
[18:07:03.350][10.0.0.2:51799] server disconnect 10.0.0.5:8080
[18:07:03.397][10.0.0.2:51800] server disconnect 10.0.0.5:8080
[18:07:13.283][10.0.0.3:50512] client connect
[18:07:13.288][10.0.0.3:50512] server connect 10.0.0.5:8080
Client Connection Address: 10.0.0.3, Failed Attempts: 1
[18:07:16.477][10.0.0.3:50513] client connect
[18:07:16.486][10.0.0.3:50513] server connect 10.0.0.5:8080
[18:07:18.253][10.0.0.3:50514] client connect
[18:07:18.258][10.0.0.3:50514] server connect 10.0.0.5:8080
Client Connection Address: 10.0.0.3, Failed Attempts: 2
[18:07:23.279][10.0.0.3:50514] server disconnect 10.0.0.5:8080
[18:07:24.394][10.0.0.3:50514] server connect 10.0.0.5:8080
Client Connection Address: 10.0.0.3, Failed Attempts: 3
Connection 10.0.0.3 is changed to malicious
[18:07:29.407][10.0.0.3:50514] server disconnect 10.0.0.5:8080
[18:07:33.545][10.0.0.3:50513] server disconnect 10.0.0.5:8080
[18:07:33.566][10.0.0.3:50512] server disconnect 10.0.0.5:8080
```

Figure 14. Test Case 1 alerts displayed on FCU proxy's terminal

B. TEST CASE 2 – CHANGE DETECTION, INTERCEPTION, AND DECEPTIVE RESPONSE

In Test Case 2, the FCU proxy successfully detected a user change request, intercepted the message, and sent a fake WebCTRL server response. Figure 15 shows the results from Steps 1–4 (Section IV.2), starting with resetting all connections by loading the FCU proxy program (V9.py). Immediately after the connection reset, the FCU proxy detected that 0.2 was not on the allowed list and flagged it as malicious (Step 2). However, the connection was allowed to continue.

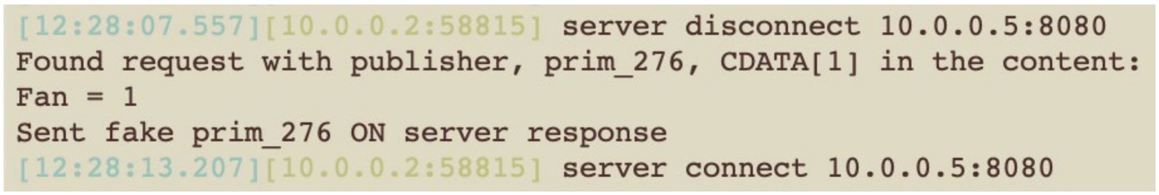
In Step 3, when the malicious client navigated to the Control webpage (Figure 12), the FCU proxy gathered the current equipment configurations of prim_272 (override checkbox) as OFF and prim_276 (fan status) as ON (Figure 15). For Step 4, no logging occurs as the change was from the authorized client. For Step 5, when the malicious client turned off the fan, the browser sent a change request message to WebCTRL. The FCU proxy correctly observed the keywords (publisher, prim_276, CDATA[0]) in the body of the message and intercepted the message. The FCU proxy updated the global variable for the fan's deceptive state to OFF (the message "Fan = 0" in Figure 15), generated the fake responses, and sent them to the client (the two "sent" terminal messages in Figure 15).



```
[12:27:11.779][10.0.0.2:58807] server disconnect 10.0.0.5:8080
[12:27:27.801] Loading script V9.py
10.0.0.2 is not on allow list, connection is malicious
[12:27:31.260][10.0.0.2:58807] server connect 10.0.0.5:8080
[12:27:31.389][10.0.0.2:58808] server connect 10.0.0.5:8080
[12:27:31.390][10.0.0.2:58805] server connect 10.0.0.5:8080
[12:27:31.393][10.0.0.2:58814] server connect 10.0.0.5:8080
[12:27:31.393][10.0.0.2:58815] server connect 10.0.0.5:8080
prim_272 initial value set to = 0
prim_276 initial value set to = 1
[12:27:37.427][10.0.0.2:58815] server disconnect 10.0.0.5:8080
[12:27:37.428][10.0.0.2:58807] server disconnect 10.0.0.5:8080
[12:27:37.438][10.0.0.2:58808] server disconnect 10.0.0.5:8080
[12:27:37.442][10.0.0.2:58814] server disconnect 10.0.0.5:8080
[12:27:37.446][10.0.0.2:58804] server disconnect 10.0.0.5:8080
Found request with publisher, prim_276, CDATA[0] in the content:
Fan = 0
Sent fake prim_276 OFF server response
Sent 272 ON / 276 OFF
Fan = 0
[12:28:01.550][10.0.0.2:58814] server connect 10.0.0.5:8080
```

Figure 15. Test Case 2 results – steps 1–4

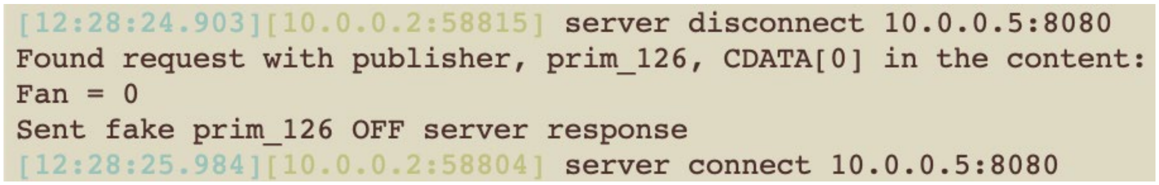
Figure 16 shows the result from Step 6 of Test Case 2 (Section IV.2), i.e., when the malicious client turned on the fan. The proxy detected the change request by identifying the keywords (publisher, prim_276, CDATA[1]) in the change request message, and intercepted the message. The FCU proxy updated the global variable for the fan's deceptive state to ON (the message "Fan = 1 in Figure 16). The "sent" message in Figure 16 shows the proxy successfully created and sent a fake response when a request to turn the fan on was detected.



```
[12:28:07.557][10.0.0.2:58815] server disconnect 10.0.0.5:8080
Found request with publisher, prim_276, CDATA[1] in the content:
Fan = 1
Sent fake prim_276 ON server response
[12:28:13.207][10.0.0.2:58815] server connect 10.0.0.5:8080
```

Figure 16. Test Case 2 result – step 6

Figure 17 shows the result from Step 7 of Test Case 2 (Section IV.2) when the malicious user turned off the fan from the Logic webpage (Figure 13). The proxy detected the change request by identifying the keywords (publisher, prim_126, CDATA[0]) in the change request message. The proxy set the global variable for the fan's deceptive state to OFF (the message Fan = 0 in Figure 17). The "sent" message in Figure 17 shows the proxy correctly detected a user change message, intercepted a malicious attempt to turn the fan off, and sent a deceptive response.

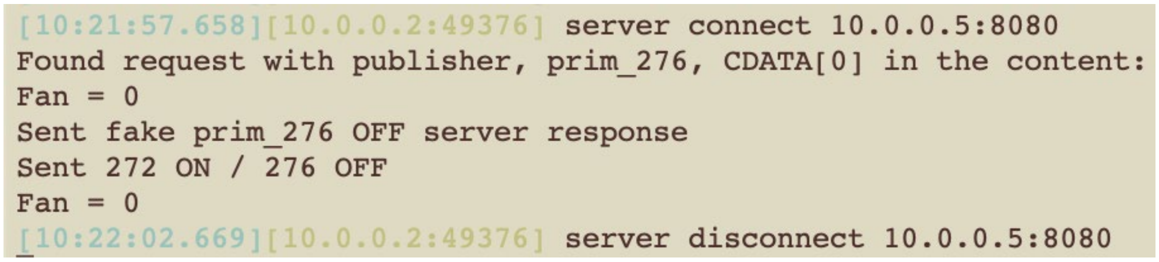


```
[12:28:24.903][10.0.0.2:58815] server disconnect 10.0.0.5:8080
Found request with publisher, prim_126, CDATA[0] in the content:
Fan = 0
Sent fake prim_126 OFF server response
[12:28:25.984][10.0.0.2:58804] server connect 10.0.0.5:8080
```

Figure 17. Test Case 2 results – step 7

C. TEST CASE 3 – DECEPTIVE WEB INTERFACE

In Test Case 3, the proxy successfully detected and intercepted a change message, updated the global variables, sent the deceptive WebCTRL server response, and maintained the deceptive state as the user navigated to different webpages. After completing Step 3 (Section IV.3), the proxy's terminal displayed the change request, interception, and deceptive message response (Figure 18). Both clients then navigated to the Graphics webpage (Figure 3). The webpage on the malicious client showed the fan was off, the GIF image of the fan stopped spinning, and the valve was shown at 0 (Figure 19). The webpage on the authorized client, on the other hand, showed the correct equipment status with the fan on, the GIF image spinning, and the valve at 100 (Figure 20).



```
[10:21:57.658][10.0.0.2:49376] server connect 10.0.0.5:8080
Found request with publisher, prim_276, CDATA[0] in the content:
Fan = 0
Sent fake prim_276 OFF server response
Sent 272 ON / 276 OFF
Fan = 0
[10:22:02.669][10.0.0.2:49376] server disconnect 10.0.0.5:8080
```

Figure 18. Test Case 3 result – step 3

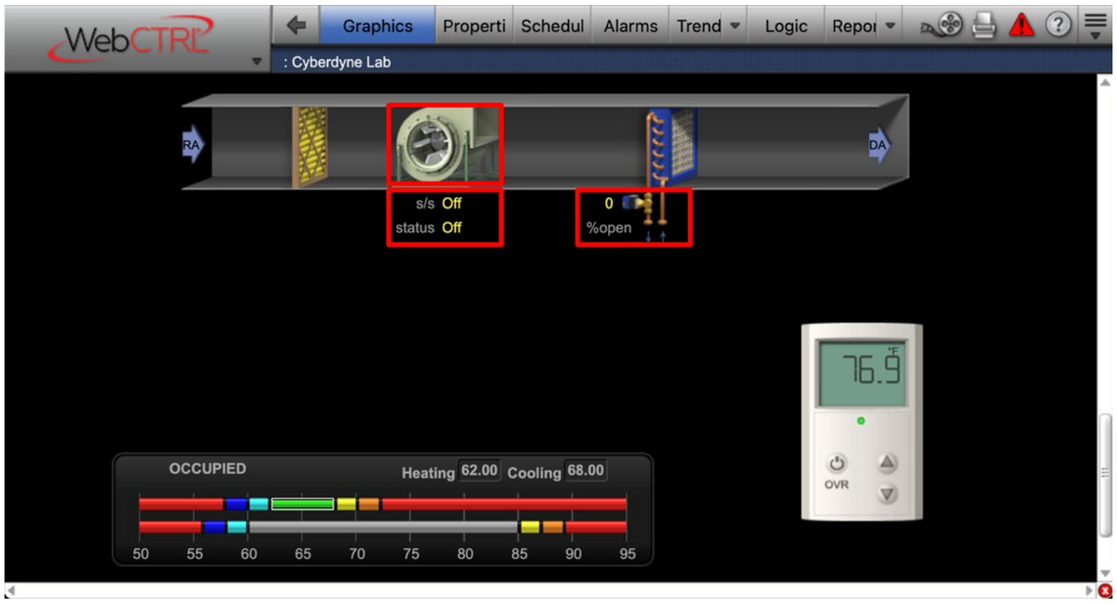


Figure 19. Test Case 3 result on malicious client – step 3

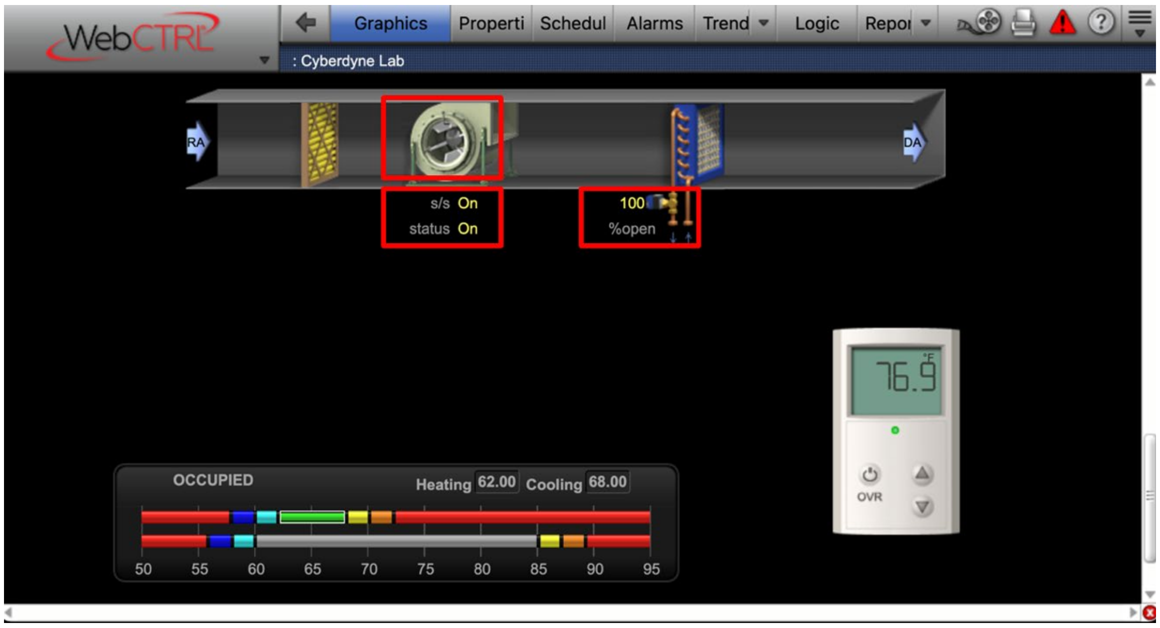
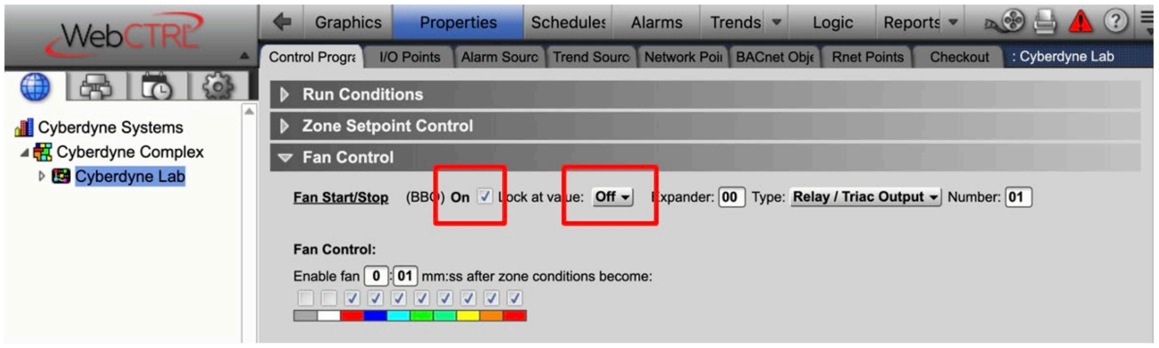


Figure 20. Test Case 3 result on authorized client – step 3

In Step 4 (Section IV.3), both clients navigated back to the Control webpage to check the fan's status. The FCU proxy modified the malicious client's webpage to show the deceptive state of the manual override with the checkbox (enabled) and the fan position (off) (Figure 21). In comparison, the authorized client's webpage showed the correct equipment configuration (Figure 22).

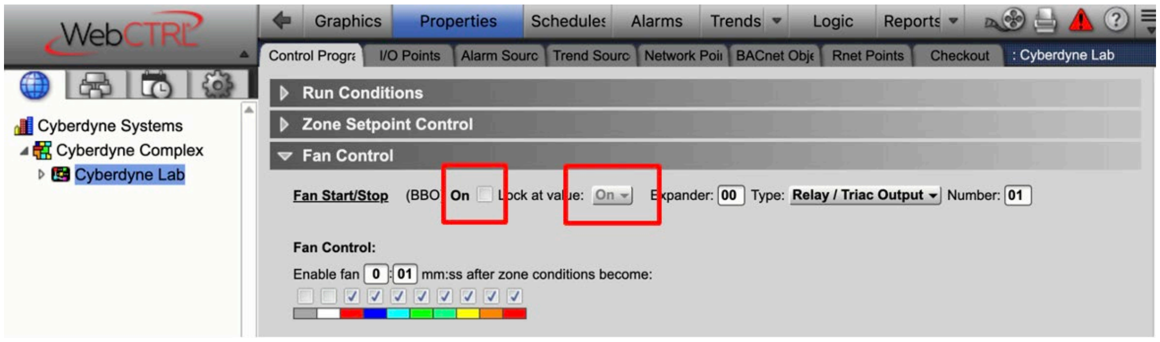


Fan Start/Stop (BBO) On ✓ Lock at value: Off Expander: 00 Type: Relay / Triac Output Number: 01

Fan Control:

Enable fan 0 : 01 mm:ss after zone conditions become:

Figure 21. Test Case 3 result on malicious client – step 4



Fan Start/Stop (BBO On Lock at value: On Expander: 00 Type: Relay / Triac Output Number: 01

Fan Control:

Enable fan 0 : 01 mm:ss after zone conditions become:

Figure 22. Test Case 3 result on authorized client – step 4

VI. CONCLUSION AND FUTURE WORK

This thesis explored using cyberdeception to protect building-automation systems. For this investigation, we developed a high-interaction hybrid honeypot that combines a deceptive reverse-proxy service and commercial building-automation equipment (controllers, sensors, and actuators) to deceive attackers with real data. Our fan-control honeypot can fool attackers into thinking that they can control part of an HVAC system. An attack scenario would be turning off the fans in a server room to cause the servers to overheat.

A. SUMMARY OF FINDINGS

To create the deception, we analyzed the communications protocols used by the WebCTRL server to allow a remote user to control a fan-coil system. We developed methods to intercept and inject fake data in response to suspicious requests while allowing authenticated users access to real data. The FCU proxy also created false graphics to lure and convince attackers to interact with the honeypot. The results of Test Case 1 (Section V.A) demonstrated the FCU proxy successfully detected malicious connections and brute-force login attempts. The results of Test Case 2 (Section V.B) showed the FCU proxy correctly detected and intercepted malicious equipment-change requests and generated fake responses. The results of Test Case 3 (Section V.C) demonstrated the FCU proxy properly maintained deceptive states across different webpages.

B. FUTURE WORK

Deception was only implemented in selected webpages relating to the fan in two website navigation menus (Control and Logic). A next step could be to add deception to all other fan-related webpages, including providing more feedback, such as changing temperature readings on the thermostat after the fan is turned off. Besides fan control, the WebCTRL server also can control the thermostat and other related equipment. The fan-control proxy could implement additional deceptions for these devices.

Opportunities exist for protecting the WebCTRL server from cyberattacks other than brute-force ones. More capability could be placed into screening incoming connections to provide more enticing data to Internet scans or to block bot (autonomous) attacks. Also, the fan-control proxy could be moved to the Internet to collect data on real attackers. Other areas to explore are examining network traffic to discover potential exploitable weaknesses in the WebCTRL server and developing black-box security testing that considers only externally visible behavior of the WebCTRL server.

LIST OF REFERENCES

American National Standards Institute [ANSI]. (2020). *BACnet a data communication protocol for building automation and control networks* (ANSI/ASHRAE Standard 135–2020*).* https://bacnet.org/about-bacnet-standard/

Assante, M., Conway, T., & Lee, R., (2016, March 18). *Analysis of the cyber-attack on the Ukrainian power grid.* Electricity Information Sharing and Analysis Center (E-ISAC). https://media.kasperskycontenthub.com/wp-content/uploads/sites/58/2016/12/21181126/E-ISAC_SANS_Ukraine_DUC_5.pdf

Automated Logic (n.d. a). *LGE.* Retrieved November 13, 2023, from https://www.manualslib.com/manual/1827699/Automated-Logic-Lge.html

Automated Logic (n.d. b). *SE6104A and SE6104SP*. Retrieved November 13, 2023, from https://www.automatedlogic.com/en/media/SE6104_SE6104sp_CS-1222_tcm702-100136.pdf

Automated Logic (2019, May 9). *WebCTRL V7.0.* https://www.mindmeister.com/generic_files/get_file/10093951?filetype=attachment_file

BeEF [The Browser Exploitation Framework] (n.d.). *What is BeEF?* Retrieved October 7, 2023, from https://beefproject.com

Beerman, J., Berent, D., Falter, Z., & Bhunia, S. (2023). A Review of Colonial Pipeline Ransomware Attack. 2023 *IEEE/ACM 23rd International Symposium on Cluster, Cloud, and Internet Computing Workshops* (CCGridW), 8–15. https://doi.org/10.1109/CCGridW59191.2023.00017

Berners-Lee, T., Connolly, D. (1995, November). *Hypertext markup language – 2.0*. https://www.rfc-editor.org/rfc/pdfrfc/rfc1866.txt.pdf

BetterCAP (n.d.). *BetterCAP stable documentation*. Retrieved October 28, 2023, from https://www.bettercap.org/legacy/

Bythwood, W., Kien, A., & Vakilinia, I. (2023). Fingerprinting bots in a hybrid honeypot. *SoutheastCon 2023. 76–80.* https://doi.org/10.1109/southeastcon51012.2023.10115143

Cortesi, A., Hils, M., Kriechbaumer, T., & contributors (2023) *MITMproxy: A free and open-source interactive HTTPS proxy.* https://mitmproxy.org

Colbert, E., & Kott, A. (2016). Operational Technology and Information Technology in Industrial Control Systems. In *Cyber-Security of SCADA and Other Industrial Control Systems* (Vol. 66, pp. 51–68). Springer International Publishing AG. https://doi.org/10.1007/978-3-319-32125-7_4

DeJonghe, D. (2022). *NGINX cookbook*. O'Reilly Media, Inc. https://www.studocu.com/in/document/indian-institute-of-technology-kanpur/principle-of-communication/nginx-cookbook-2nd-ed-2022/61162691

Docker (n.d.). *Overview of the get started guide.* Retrieved November 23, 2023, from https://docs.docker.com/get-started/

Ettercap (n.d.). *Ettercap Home Page*. Retrieved October 28, 2023, from https://www.ettercap-project.org/index.html

Fielding, R., Nottingham, M., & Reschke, J. (2022, June). *HTTP Caching*. Internet Engineering Task Force (IETF) 9112. https://www.rfc-editor.org/rfc/rfc9112.pdf

Franco, J., Aris, A., Canberk, B., & Uluagac, A. S. (2021). A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems. *IEEE Communications Surveys and Tutorials, 23*(4), 2351–2383. https://doi.org/10.1109/COMST.2021.3106669

Haney, M. (2019). Leveraging Cyber-Physical System Honeypots to Enhance Threat Intelligence. *Critical Infrastructure Protection XIII*, *570*, 209–233. https://doi.org/10.1007/978-3-030-34647-8_11

Hollenbeck, S., Rose, M., & Masinter, L. (2003, January). *Guidelines for the use of extensible markup language (XML) within IETF Protocols*. https://www.rfc-editor.org/rfc/pdfrfc/rfc3470.txt.pdf

Ivanova, S., & Moradpoor, N. (2023). Fake PLC in the Cloud, we thought the Attackers Believed that: how ICS Honeypot Deception gets Impacted by Cloud Deployments? *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*, Pavia, Italy, 2023, pp. 1–4, https://doi.org/10.1109/wfcs57264.2023.10144119

Koay, A., Ko, R., Hettema, H., & Radke, K. (2023). Machine learning in industrial control system (ICS) security: current landscape, opportunities, and challenges. *Journal of Intelligent Information Systems 60*, 377–405 (2023). https://doi.org/10.1007/s10844-022-00753-1

Litchfield, S. (2017, May). HoneyPhy: a physics-aware CPS honeypot. https://repository.gatech.edu/entities/publication/1632d65a-12fb-4442-aecc-ef96e48ee033

MITMproxy (n.d.). *MITMproxy Docs.* Retrieved September 29, 2023, from https://docs.mitmproxy.org/stable/

NGINX (n.d.). *NGINX*. Retrieved October 28, 2023, from https://www.nginx.com

National Institute of Standards and Technology [NIST] (2023, September). *Guide to operational technology (OT) security*. (NIST SP 800-82 Rev. 3). https://doi.org/10.6028/NIST.SP.800-82r3

PortSwigger (n.d.). *Burp Suite Professional.* Retrieved October 7, 2023, from https://portswigger.net/burp/pro/features

Razali, M. N., Mansor, F. Z., Muruti, G., & Jamil, N. (2018). IoT Honeypot: A Review from a Researcher's Perspective. *2018 IEEE Conference on Application, Information and Network Security (AINS)*, 93–98, https://doi.org/10.1109/AINS.2018.8631494

Rowe, N., & Rrushi, J. (2016). *Introduction to Cyberdeception* (1st ed. 2016.). Springer International Publishing. https://doi.org/10.1007/978-3-319-41187-3

Salesforce (n.d.) *"HASSH" – A profiling method for SSH clients and servers.* Retrieved from November 23, 2023, from https://github.com/salesforce/hassh

Senol, M. (2022). Cyber Security and Defense: Proactive Defense and Deterrence. *2022 3rd International Informatics and Software Engineering Conference (IISEC)*, 1–6, https://doi.org/10.1109/IISEC56263.2022.9998314

Saxena, N., Grijalva, S., & Choi, B. J. (2018). Securing restricted publisher-subscriber communications in smart grid substations. *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, 364–371. https://doi.org/10.1109/COMSNETS.2018.8328220

Wilhoit, K., & Hilt, S. (2015). *The GasPot Experiment: Unexamined perils in using gas-tank-monitoring systems*. Trend Micro. https://www.blackhat.com/docs/us-15/materials/us-15-Wilhoit-The-Little-Pump-Gauge-That-Could-Attacks-Against-Gas-Pump-Monitoring-Systems-wp.pdf

You, J., Lv, S., Sun, L. Y., & Wen, H. (2021). HoneyVP: A cost-effective hybrid honeypot architecture for industrial control systems. *ICC 2021 – IEEE International Conference on Communications,* 1–6. https://doi.org/10.1109/ICC42927.2021.9500567

ZAP (n.d.). *Getting Started.* Retrieved October 7, 2023, from https://www.zaproxy.org/getting-started/

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California