TryHackMe — Rogue Poller

📊 TryHackMe Task 22 — Rogue Poller

Tags: #TryHackMe #Networking #Modbus #PCAPAnalysis #OTSecurity #CTF

🧠 Scenario

> *An intruder on the OT network is polling Modbus registers via TCP/502. Your job is to **analyze the network capture** and determine what data was accessed.*

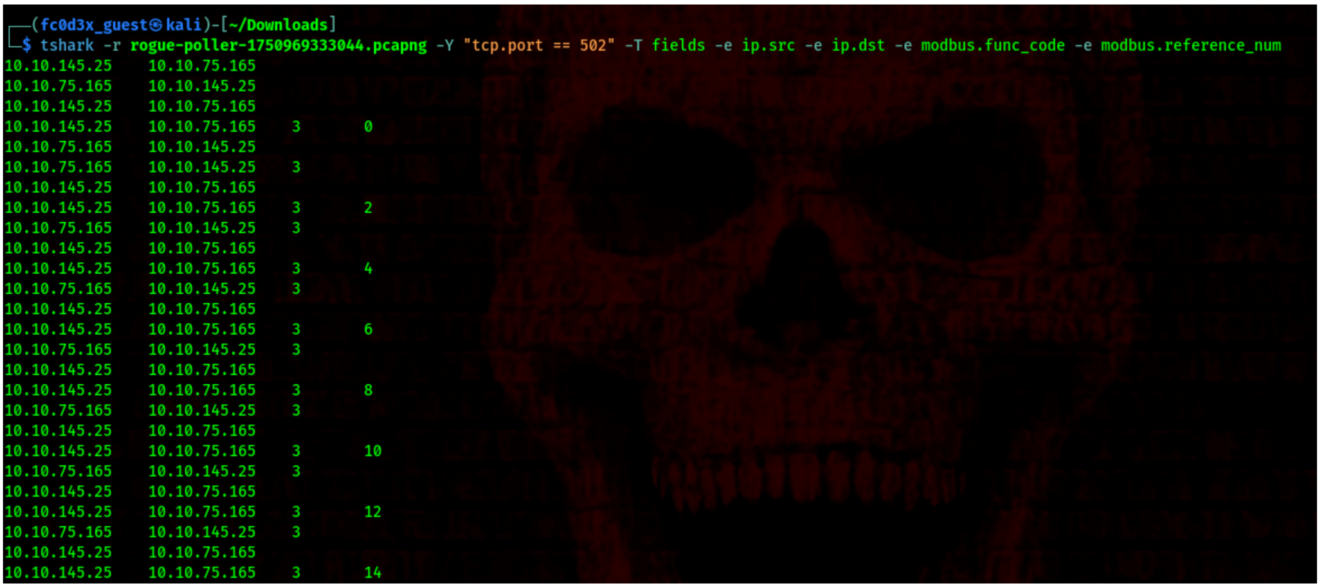
---

1. TShark Modbus Traffic Filtering

The first step in my attack chain was to isolate Modbus traffic. Using TShark (the CLI version of Wireshark), I filtered out all non-Modbus traffic in the provided PCAP file (`rogue-poller-1750969333044.pcapng`). The command I used was:

```
tshark -r rogue-poller-1750969333044.pcapng -Y "tcp.port == 502" -T fields
-e ip.src -e ip.dst -e modbus.func_code -e modbus.reference_num
```

This gave me the source and destination IPs, function codes, and reference numbers, allowing me to pinpoint the exact Modbus interactions between the attacker and the PLC.



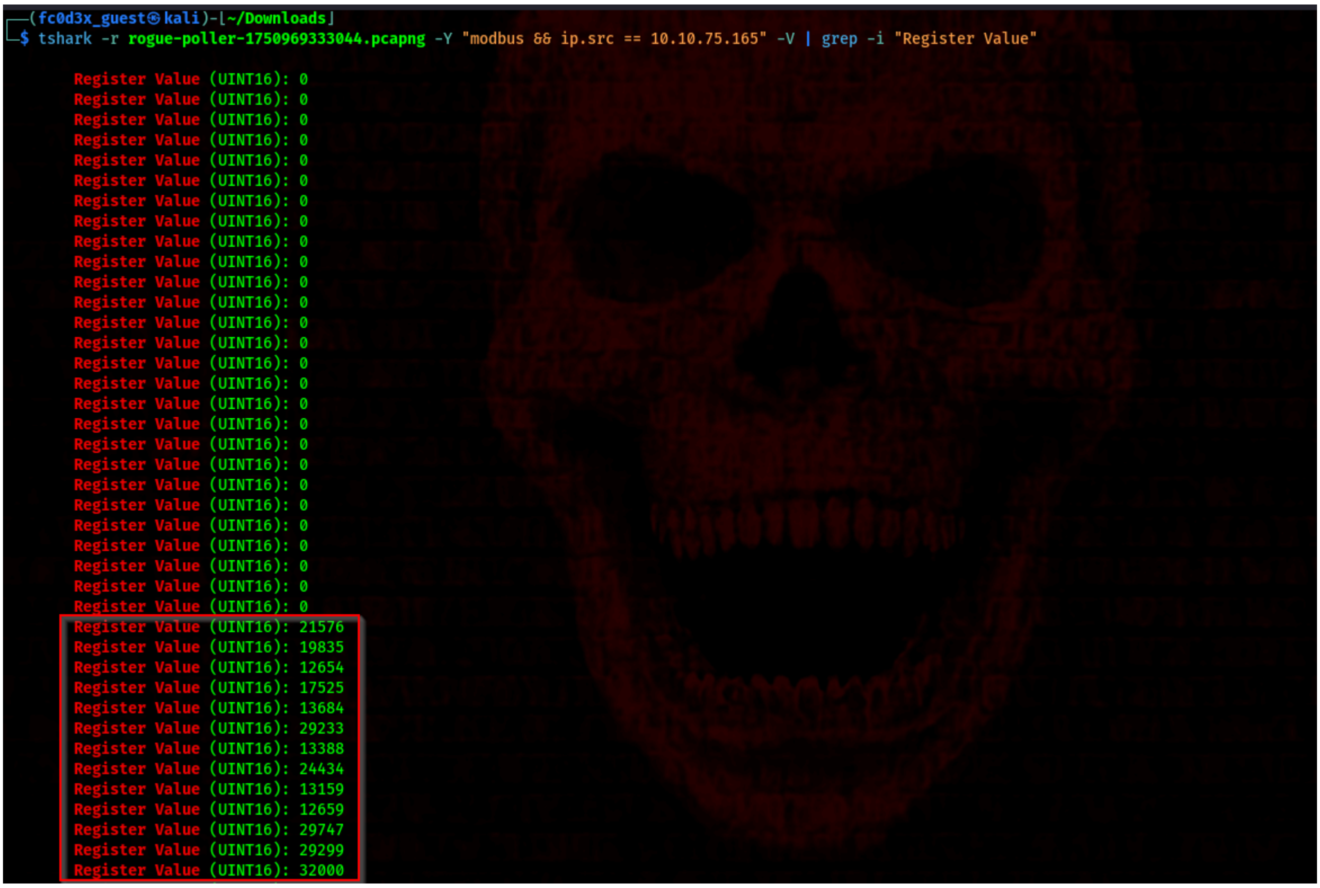
---

2. Extract Response Register Values

I then focused on the response packets, as they contained the actual data. Using the following command to filter by Modbus and source IP, I was able to extract register values:

```
tshark -r rogue-poller-1750969333044.pcapng -Y "modbus && ip.src ==
10.10.75.165" -V | grep -i "Register Value"
```

From here, I identified several non-zero values from the Modbus response that stood out among the mostly zeroed registers.



---

3. Identifying Interesting Non-Zero Register Values

The non-zero register values were my next target. These values likely contained important data or the flag. The response I obtained looked like this:

> **Register Value (UINT16):** 21576
> **Register Value (UINT16):** 19835
> **Register Value (UINT16):** 12654
> **Register Value (UINT16):** 17525
> **Register Value (UINT16):** 13684
> **Register Value (UINT16):** 29233
> **Register Value (UINT16):** 13388
> **Register Value (UINT16):** 24434

Register Value (UINT16): 13159

Register Value (UINT16): 12659

Register Value (UINT16): 29747

Register Value (UINT16): 29299

Register Value (UINT16): 32000

These values were clearly significant, so I wrote a quick bash script to decode them.

---

4. Decoding the Flag with a Bash Script

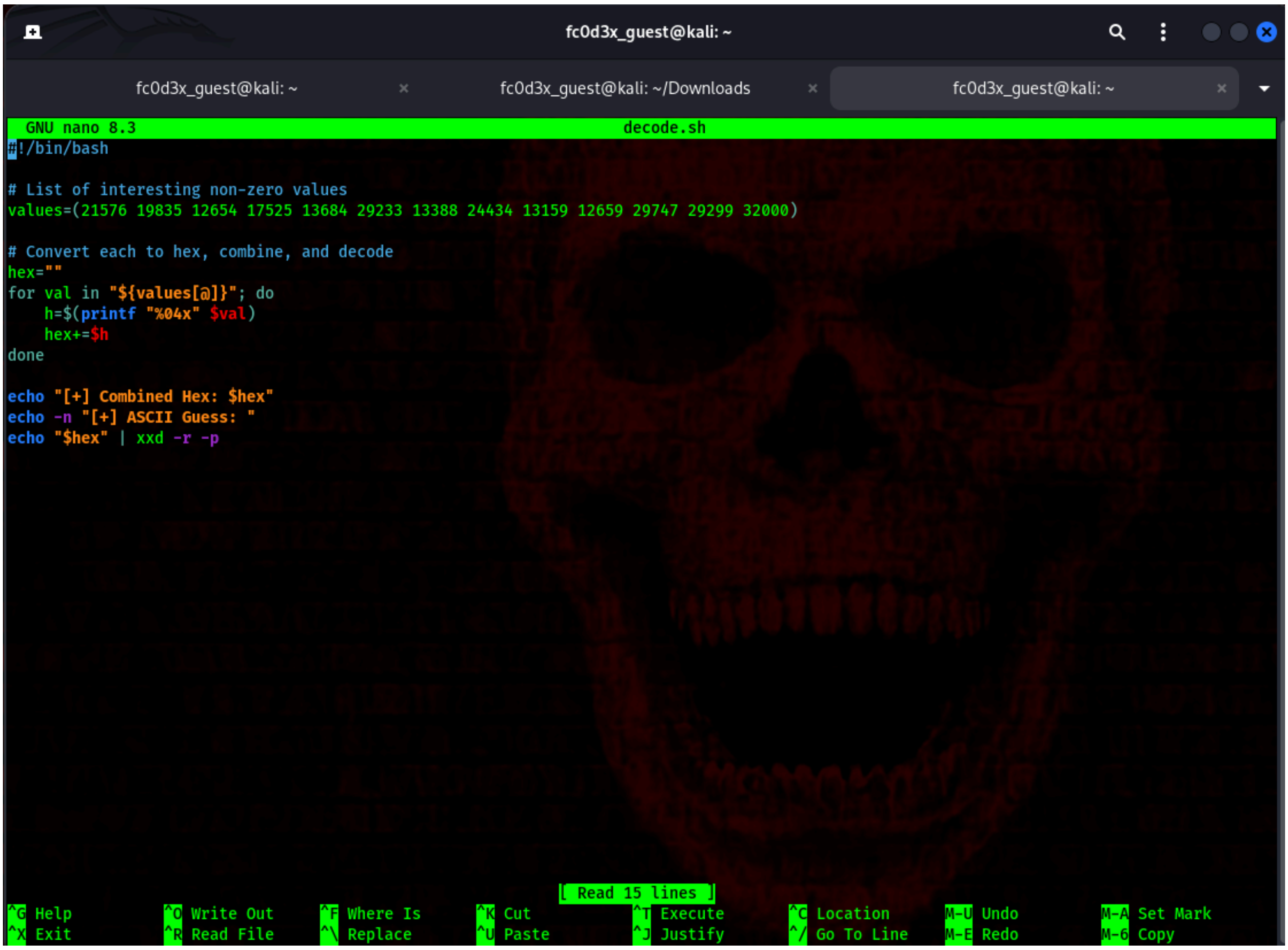
I created a simple bash script to convert these UINT16 values into hexadecimal, concatenate them, and then decode them into ASCII. Here is the script I used:

```
#!/bin/bash

values= 21576 19835 12654 17525 13684 29233 13388 24434 13159 12659 29747
29299 32000

hex=""
for val in "$ values[@] "  do
h=$ printf "%04x" $val
hex+=$h
done

echo "[+] Combined Hex: $hex"
echo -n "[+] ASCII Guess: "
echo "$hex" | xxd -r -p
```



After running the script, the flag revealed itself as:

```
THM Industrial_registers
```

---

5. Conclusion

What I learned:

- **Modbus Vulnerabilities:** Modbus lacks encryption or authentication by default, which is a huge security risk when exposed on the network, especially in unsegmented environments.

- **TShark:** It proved to be an extremely powerful tool for packet forensics, enabling me to automate the extraction of meaningful data from the traffic.
- **Memory Access:** Even seemingly innocuous Modbus read requests can expose sensitive data such as flags, configuration secrets, and more when decoded correctly.

In this red team exercise, I successfully used my knowledge of industrial protocols to extract and decode sensitive information hidden within Modbus traffic. This highlights the importance of securing industrial control systems (ICS) against such attacks.