



Using ChatGPT to Create Defensive & Offensive ICS/OT Tools

Disclaimer

- The information provided in this course is for informational purposes only.
- All knowledge obtained must be used for legitimate, authorized purposes.
- The views written and discussed are those of the presenter and not any of their employers or other affiliated organizations.
- Some photos and other evidence included are fictitious and not related to any real incident.

About Mike Holcomb



- Fellow for Cyber Security at Fluor
- ICS/OT Cyber Security Global Lead
- Founder of Upstate SC ISSA and BSides Greenville
- Wrote and taught all six cyber security courses for Greenville Technical College's cyber degree program
- CISSP, GRID, GICSP, ISA 62443, GPEN, GCIH, etc.
- Master's degree in ICS/OT cyber security

About Mike Holcomb (cont.)



Agenda

- ❑ Getting Started with ChatGPT
- ❑ Living off the Land
- ❑ Troubleshooting Errors
- ❑ Building Asset Registers from Packet Captures
- ❑ Finding PLCs on the Internet
- ❑ Helping ChatGPT Understand Little by Little
- ❑ Going on the Offensive
- ❑ What Comes Next?



Our Story Begins

**“It’s a PLC.
It’s NOT vulnerable.”**



Monitoring PLC Operational Modes

PLC Monitoring Station

PLC Name	Vendor	Model	Firmware	Location	IP Address	MAC Address	Mode
PLC-1	Siemens	S7-1200	V4.2	Production Line 1	192.168.1.2	00:1B:21:12:34:56	Run
PLC-2	Allen-Bradley	MicroLogix 1400	21.003	Assembly Line 3	192.168.2.3	00:1D:9C:78:90:AB	Run
PLC-3	Schneider Electric	M340	V3.5	Packaging Area	192.168.3.4	00:80:F4:56:78:9A	Run
PLC-4	Siemens	S7-1500	V2.1	Warehouse Section A	192.168.4.5	00:1B:21:23:45:67	Run
PLC-5	Allen-Bradley	ControlLogix	32.011	Quality Control	192.168.5.6	00:1D:9C:89:01:BC	Run
PLC-6	Schneider Electric	M580	V3.0	Utility Room	192.168.6.7	00:80:F4:67:89:AB	Run
PLC-7	Siemens	S7-300	V3.3	Production Line 2	192.168.7.8	00:1B:21:34:56:78	Run
PLC-8	Allen-Bradley	CompactLogix	28.015	Loading Dock	192.168.8.9	00:1D:9C:90:12:CD	Run
PLC-9	Schneider Electric	M221	V1.8	Maintenance Bay	192.168.9.10	00:80:F4:78:9A:BC	Run
PLC-10	Siemens	S7-400	V6.0	Warehouse Section B	192.168.10.11	00:1B:21:45:67:89	Run

Monitoring PLC Operational Modes

PLC Monitoring Station

PLC Name	Vendor	Model	Firmware	Location	IP Address	MAC Address	Mode
PLC-1	Siemens	S7-1200	V4.2	Production Line 1	192.168.1.2	00:1B:21:12:34:56	Run
PLC-2	Allen-Bradley	MicroLogix 1400	21.003	Assembly Line 3	192.168.2.3	00:1D:9C:78:90:AB	Run
PLC-3	Schneider Electric	M340	V3.5	Packaging Area	192.168.3.4	00:80:F4:56:78:9A	INVESTIGATE Resolve
PLC-4	Siemens	S7-1500	V2.1	Warehouse Section A	192.168.4.5	00:1B:21:23:45:67	Run
PLC-5	Allen-Bradley	ControlLogix	32.011	Quality Control	192.168.5.6	00:1D:9C:89:01:BC	Run
PLC-6	Schneider Electric	M580	V3.0	Utility Room	192.168.6.7	00:80:F4:67:89:AB	Run
PLC-7	Siemens	S7-300	V3.3	Production Line 2	192.168.7.8	00:1B:21:34:56:78	Run
PLC-8	Allen-Bradley	CompactLogix	28.015	Loading Dock	192.168.8.9	00:1D:9C:90:12:CD	Run
PLC-9	Schneider Electric	M221	V1.8	Maintenance Bay	192.168.9.10	00:80:F4:78:9A:BC	Run
PLC-10	Siemens	S7-400	V6.0	Warehouse Section B	192.168.10.11	00:1B:21:45:67:89	Run



Monitoring PLC Operational Modes

PLC Monitoring Station

PLC Name	Vendor	Model	Firmware	Location	IP Address	MAC Address	Mode
PLC-1	Siemens	S7-1200	V4.2	Production Line 1	192.168.1.2	00:1B:21:12:34:56	Run
PLC-2	Allen-Bradley	MicroLogix 1400	21.003	Assembly Line 3	192.168.2.3	00:1D:9C:78:90:AB	Run
PLC-3	Schneider Electric	M340	V3.5	Packaging Area	192.168.3.4	00:80:F4:56:78:9A	Run
PLC-4	Siemens	S7-1500	V2.1	Warehouse Section A	192.168.4.5	00:1B:21:23:45:67	Run
PLC-5	Allen-Bradley	ControlLogix	32.011	Quality Control	192.168.5.6	00:1D:9C:89:01:BC	Run
PLC-6	Schneider Electric	M580	V3.0	Utility Room	192.168.6.7	00:80:F4:67:89:AB	Run
PLC-7	Siemens	S7-300	V3.3	Production Line 2	192.168.7.8	00:1B:21:34:56:78	Run
PLC-8	Allen-Bradley	CompactLogix	28.015	Loading Dock	192.168.8.9	00:1D:9C:90:12:CD	Run
PLC-9	Schneider Electric	M221	V1.8	Maintenance Bay	192.168.9.10	00:80:F4:78:9A:BC	Run
PLC-10	Siemens	S7-400	V6.0	Warehouse Section B	192.168.10.11	00:1B:21:45:67:89	Run

Getting Started with ChatGPT

Prompt:

Write a Python script that will print "Hello, friend..." on the screen.



Creating Your First Python Script

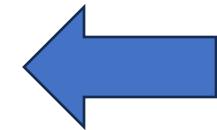


Here's a simple Python script that will print "Hello, friend..." on the screen:

```
python
```

 Copy code

```
print("Hello, friend...")
```



You can save this script in a file with a ` `.py` extension, for example, `hello_friend.py` , and run it using a Python interpreter. To run the script, open a terminal or command prompt, navigate to the directory where you saved the file, and execute:

```
bash
```

 Copy code

```
python hello_friend.py
```

This will print "Hello, friend..." to the screen.



Running hello_friend.py

```
c:\Scripts>hello_friend.py
```

```
Hello, friend...
```



Creating a Basic ICS/OT Port Scanner

Prompt:

Write a Python script to detect the presence of open TCP ports on a single target IP address. The script will ask the user to provide the IP address. The script should check to see if the IP address provided is valid or not. If not valid, ask the user again for the IP address. If valid, scan the IP address to determine if TCP 80, 443, 102, 502 and 44818 are open on the target system.

Provide an organized list of each port and their status.



Running simple_scanner.py

```
c:\Scripts>simple_scanner.py
Please enter the IP address: 10.2.1.1

Scanning IP address: 10.2.1.1

Open Ports:
Port 443: Open
```



Running simple_scanner.py

```
c:\Scripts>simple_scanner.py
Please enter the IP address: 192.168.100.200

Scanning IP address: 192.168.100.200

Open Ports:
Port 502: Open
```



Living off the Land

- ❑ When we don't want to make changes to a system by installing a new component
 - ❑ Leverages built-in capabilities of the system
 - ❑ No need to create ICS-specific malware
 - ❑ Helps adversaries and pentesters remain stealthy
 - ❑ Tools like PowerShell can be extremely flexible
 - ❑ Operators and admins that simply don't want to break things



Creating a Basic ICS/OT Port Scanner

Prompt:

Convert the previous Python script to PowerShell.



Running the PowerShell Port Scanner

```
PS C:\Scripts> .\simple_scanner.ps1
At C:\Scripts\simple_scanner.ps1:47 char:26
+             Write-Host "Port $port: $status"
+                                         ~~~~~
Variable reference is not valid. ':' was not followed by a valid variable
name character. Consider using ${} to
delimit the name.
+ CategoryInfo          : ParserError: () [], ParseException
+ FullyQualifiedErrorId : InvalidVariableReferenceWithDrive
```



Troubleshooting Errors

Prompt:

```
PS C:\Scripts> .\simple_scanner.ps1
At C:\Scripts\simple_scanner.ps1:47 char:26
+     Write-Host "Port $port: $status"
+                 ~~~~~~
```

Variable reference is not valid. ':' was not followed by a valid variable name character. Consider using \${} to delimit the name.

```
+ CategoryInfo      : ParserError: () [], ParseException
+ FullyQualifiedErrorId : InvalidVariableReferenceWithDrive
```



Troubleshooting Errors



The issue arises because PowerShell is interpreting ` `\$port` as a drive colon. To avoid this, you should enclose the variable in curly braces ` \${}`.

Here's the corrected script:

```
powershell
Copy code

function Test-Port {
    param (
        [string]$ip,
        [int]$port
    )
    try {
        $tcpClient = New-Object System.Net.Sockets.TcpClient
```



Running the Fixed PowerShell Port Scanner

```
PS C:\Scripts> .\simple_scanner.ps1
Please enter the IP address: 192.168.100.200

Scanning IP address: 192.168.100.200

Port 80: Closed
Port 443: Closed
Port 102: Closed
Port 502: Open
Port 44818: Closed
```



What Do You Find Useful?

You're only limited by
your imagination.



Building an Asset Register from PCAPs

Prompt:

Create a Python script to read a PCAP file, extract all internal IP addresses and the associated MAC address for each internal IP address. Map each MAC address to the associated vendor. Save the parsed data to an asset register in Excel format. The Excel spreadsheet should be structured so that the Asset Register will include fields for PLC Name, Vendor, Model, Firmware, Location, IP Address, and MAC Address.



Parsing Packet Captures



Here's a Python script that reads a PCAP file, extracts all internal IP addresses and the associated MAC addresses, maps each MAC address to the associated vendor, and saves the parsed data to an asset register in Excel format. This script uses `scapy` to parse the PCAP file, `mac_vendor_lookup` to map MAC addresses to vendors, and `pandas` to save the data to an Excel file.

First, ensure you have the necessary libraries installed:

bash

Copy code

```
pip install scapy mac-vendor-lookup pandas openpyxl
```

Now, here's the script:

python

Copy code

```
import pandas as pd
from scapy.all import rdpcap, ARP
from mac_vendor_lookup import MacLookup
```



Parsing Packet Captures

```
c:\Scripts>parser.py
```

```
Enter the path to the PCAP file:
```



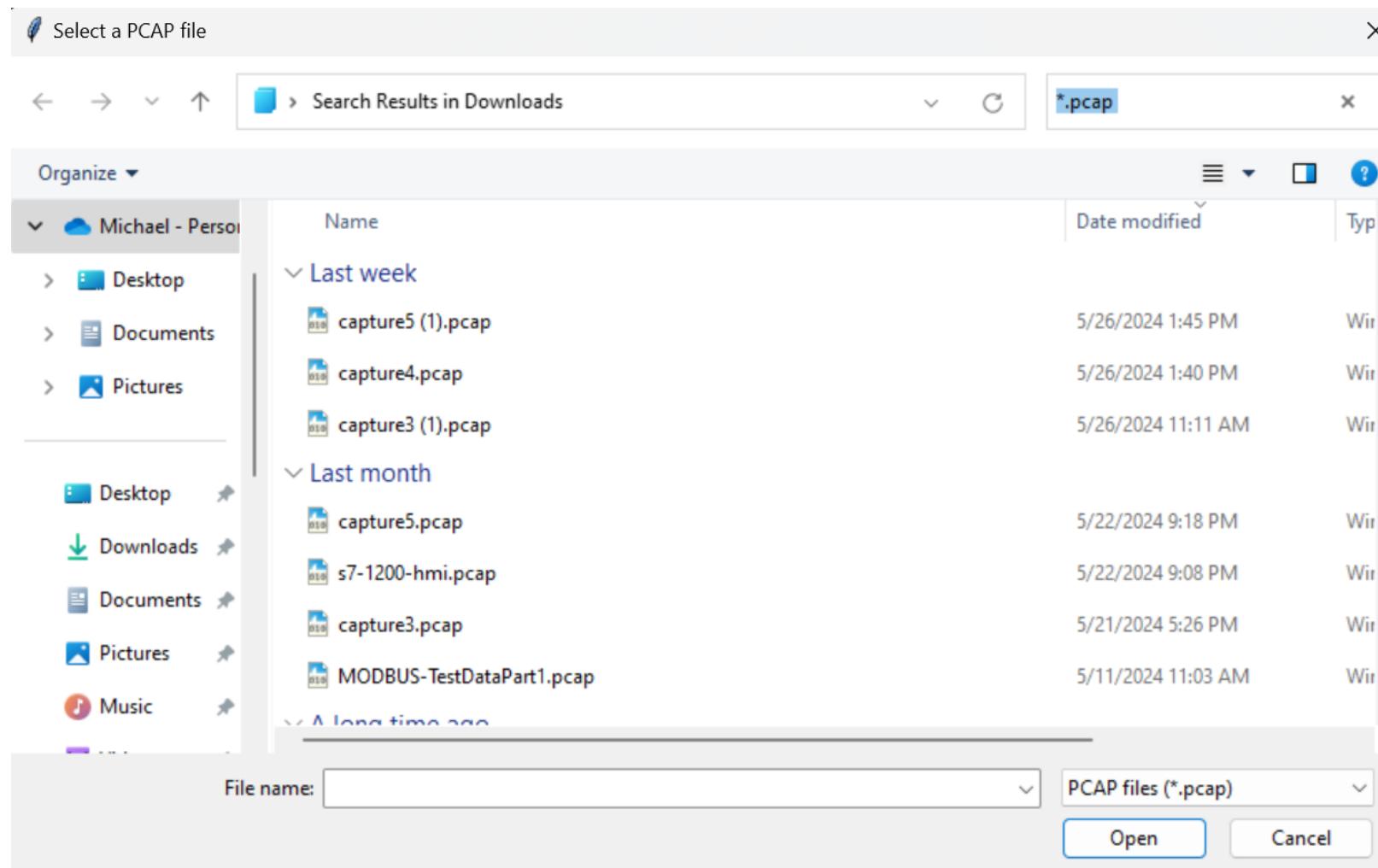
Parsing Packet Captures

Prompt:

Update the previous script to provide a graphical method for selecting the PCAP file.



Parsing Packet Captures



Parsing Packet Captures

	A	B	C	D	E	F	G
1	PLC Name	Vendor	Model	Firmware	Location	IP Address	MAC Address
2		MoxaTech			192.168.88.61	00:90:e8:27:8c:37	
3		Westeromo			192.168.88.1	00:07:7c:1a:61:83	
4		Pegatron			192.168.89.2	70:71:bc:3a:0d:e8	
5		RuggedCo			192.168.89.1	00:0a:dc:64:85:c2	
6		SiemensN			10.10.10.20	00:1c:06:27:64:11	
7		Siemens			10.10.10.10	28:63:36:89:59:82	
8		Wistronl			10.10.10.30	54:ee:75:3f:4a:db	
9		Apple			10.100.159.38	28:cf:e9:18:b5:ed	
10		Apple			192.168.88.52	28:cf:e9:18:b5:ed	



Finding PLCs on the Internet

**Remember the NSA's
ELITEWOLF project?**

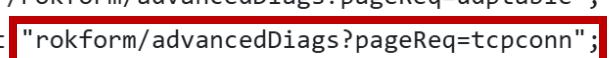


Finding PLCs on the Internet

ELITEWOLF / ELITEWOLF_SNORT_AllenBradley_RockwellAutomation.txt 

 iadgovuser61 Add files via upload b77dbc2 · 9 months ago 

Code Blame 14 lines (14 loc) · 2.35 KB Raw    

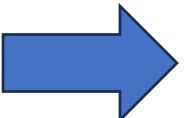
```
Rockwell Automation URL Path Activity-TCP REQUEST"; content:"/rokform/advancedDiags?pageReq=tcp"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-SYSTEM DATA DETAIL"; content:"/rokform/SysDataDetail?name="; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-UDP TABLE"; content:"/rokform/advancedDiags?pageReq=udptable"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-TCP CONNECT"; content "rokform/advancedDiags?pageReq=tcpconn"; sid:1; rev:1;)   
Rockwell Automation URL Path Activity-IP ROUTE"; content:"/rokform/advancedDiags?pageReq=iproute"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-GENERAL MEMORY"; content:"/rokform/advancedDiags?pageReq=genmem"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-HEAP REQUEST"; content:"/rokform/advancedDiags?pageReq=heap"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-ICMP REQUEST"; content:"/rokform/advancedDiags?pageReq=icmp"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-ARP REQUEST"; content:"/rokform/advancedDiags?pageReq=arp"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-UDP REQUEST"; content:"/rokform/advancedDiags?pageReq=udp"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-IF REQUEST"; content:"/rokform/advancedDiags?pageReq;if"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-IP REQUEST"; content:"/rokform/advancedDiags?pageReq=ip"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-CSS Path"; content:"/css/radvice.css"; sid:1; rev:1;)  
Rockwell Automation URL Path Activity-SYSTEM LIST DATA"; content:"/rokform/SysListDetail?name=";sid:1;rev:1;)
```



© 2024 Mike Holcomb

Finding PLCs on the Internet

PLC #1



rokform/advancedDiags?pageReq=tcpconn

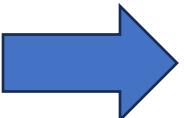


142.165.177
<http://142.165.177.228> › rokform › advancedDiags › p...

TCP Connection - Rockwell Automation

TCP Connection Table. State, Local Address, Local Port, Remote Address, Remote Port.
ESTABLISHED, 192.168.1.10, 80, 66.249.75.167, 37349.

PLC #2

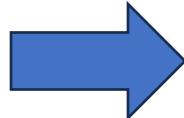


166.148.34
<http://166.148.34.152> › rokform › advancedDiags › pa...

TCP Connection - Rockwell Automation

TCP Connection Table. State, Local Address, Local Port, Remote Address, Remote Port.
ESTABLISHED, 10.25.10.40, 80, 66.249.77.77, 50585.

PLC #3



203.69.63
<http://203.69.63.85> › rokform › advancedDiags › page...

TCP Connection - Rockwell Automation

TCP Connection Table. State, Local Address, Local Port, Remote Address, Remote Port.
ESTABLISHED, 203.69.63.85, 80, 66.249.66.5, 42039.



Finding PLCs on the Internet

TCP Connection Table

State	Local Address	Local Port	Remote Address	Remote Port
ESTABLISHED	10.25.10.40	80	71.85.4.152	39840
TIME_WAIT	10.25.10.40	80	185.191.126.213	54748
TIME_WAIT	10.25.10.40	80	185.191.126.213	54756
TIME_WAIT	10.25.10.40	80	185.191.126.213	54762
TIME_WAIT	10.25.10.40	80	185.191.126.213	54774
ESTABLISHED	10.25.10.40	80	210.94.114.151	48298
ESTABLISHED	10.25.10.40	80	210.94.114.151	60888
ESTABLISHED	10.25.10.40	44818	10.25.10.41	49161
ESTABLISHED	10.25.10.40	44818	74.92.127.13	3725
ESTABLISHED	10.25.10.40	44818	74.92.127.13	19800
ESTABLISHED	10.25.10.40	44818	199.119.247.98	35454
ESTABLISHED	10.25.10.40	44818	199.119.247.98	41855
ESTABLISHED	10.25.10.40	50625	166.148.34.149	44818
ESTABLISHED	10.25.10.40	54509	166.148.34.198	44818
ESTABLISHED	10.25.10.40	62753	166.148.34.152	44818
ESTABLISHED	10.25.10.40	64485	166.148.34.150	44818



Finding PLCs on the Internet

Prompt:

Create a Python script that asks the user for a URL. The script will then connect to the URL every 60 seconds and enumerate all IP addresses from the page and then print those IP addresses on the screen.

For each IP address, perform a GeolIP location lookup. Display the GeolIP location along with each IP address.



Finding PLCs on the Internet

```
c:\Scripts>scraper.py
Please enter the full URL including https: http://166.239.24.83/rokform/advancedDiags?pageReq=tcpconn
Monitoring http://166.239.24.83/rokform/advancedDiags?pageReq=tcpconn for IP addresses. Press 'S' to stop.

New unique IP address found: 210.94.114.151 - Location: Yongsan-dong, Seoul, KR
New unique IP address found: 74.92.127.13 - Location: Philadelphia, Pennsylvania, US
New unique IP address found: 199.119.247.98 - Location: Jersey City, New Jersey, US
New unique IP address found: 166.148.34.149 - Location: Bristol, Wisconsin, US
New unique IP address found: 10.25.10.40 - Location: Private IP Address - No Location
New unique IP address found: 10.25.10.41 - Location: Private IP Address - No Location
New unique IP address found: 71.85.4.152 - Location: Simpsonville, South Carolina, US
New unique IP address found: 166.148.34.198 - Location: Bristol, Wisconsin, US
New unique IP address found: 166.148.34.150 - Location: Bristol, Wisconsin, US
New unique IP address found: 166.148.34.152 - Location: Bristol, Wisconsin, US
```



Finding PLCs on the Internet

Report: Unique IP addresses and locations found so far:

10.25.10.40 – Location: Private IP Address – No Location
10.25.10.41 – Location: Private IP Address – No Location
71.85.4.152 – Location: Simpsonville, South Carolina, US
74.92.127.13 – Location: Philadelphia, Pennsylvania, US
147.185.132.45 – Location: Charlotte, North Carolina, US
164.92.129.74 – Location: Frankfurt am Main, Hesse, DE
166.148.34.149 – Location: Bristol, Wisconsin, US
166.148.34.150 – Location: Bristol, Wisconsin, US
166.148.34.152 – Location: Bristol, Wisconsin, US
166.148.34.198 – Location: Bristol, Wisconsin, US
172.104.4.17 – Location: Morris Plains, New Jersey, US
185.191.127.212 – Location: Amsterdam, North Holland, NL
185.244.36.236 – Location: Almere Stad, Flevoland, NL
199.119.247.98 – Location: Jersey City, New Jersey, US
210.94.114.151 – Location: Yongsan-dong, Seoul, KR

Total unique IP addresses found: 15

New unique IP address found: 80.66.76.134 – Location: Amsterdam, North Holland, NL

New unique IP address found: 122.97.138.187 – Location: Nanjing, Jiangsu, CN

New unique IP address found: 167.94.146.53 – Location: Frankfurt am Main, Hesse, DE



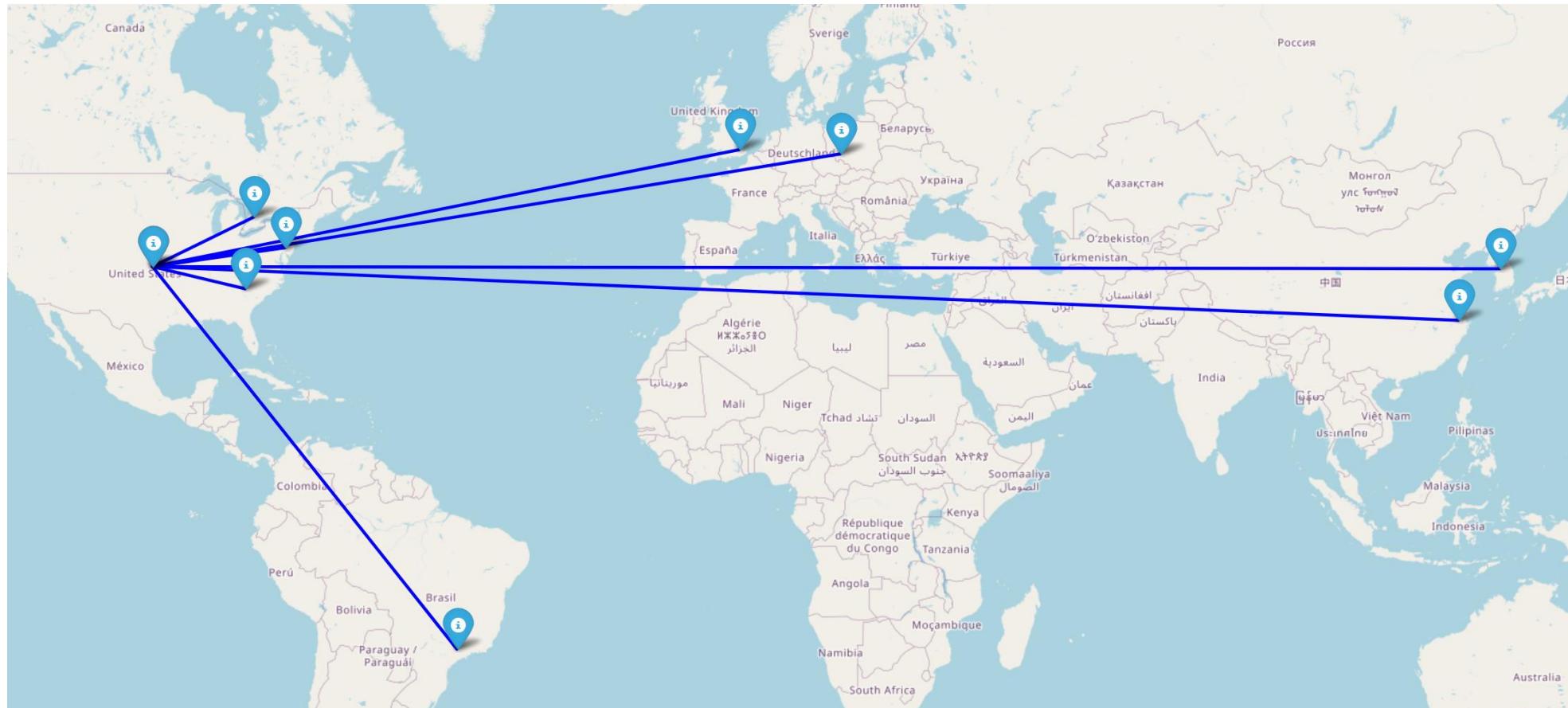
Thou Shalt Not...

Prompt:

Update the following script to write all valid public IP addresses to a text file after 1 minute. Once the file is written, use the IP addresses listed to create a map of the world in an HTML page. The IP address in the URL should be represented by a big red dot. Represent all other IP addresses by a blue dot at their geo location. Draw a line from each blue dot to the big red dot.



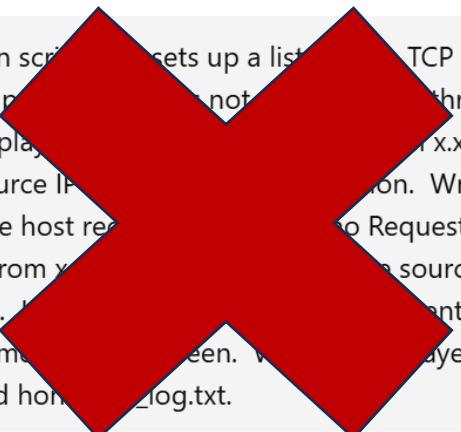
Thou Shalt Not... Draw Pew Pew Maps!



Help ChatGPT Understand Little By Little

Prompt:

Create a Python script that sets up a listening socket on TCP port 502. If it receives a SYN packet but does not receive a response to complete the three-way handshake, display a message to the screen that reads ** POSSIBLE PORT SCAN from x.x.x.x ** where x.x.x.x is the source IP address of the connection. Write a message to the screen if the host receives an ICMP Echo Request packet. ** ICMP Echo Request from x.x.x.x ** where x.x.x.x is the source IP address of the connection. If the host receives an ICMP Echo Request, write the event happens when displaying the message to the screen. Write all displayed output to a local file named honeypot_log.txt.



NO

Prompt:

Create a Python script that listens on TCP 502.

Update the script to display a message to the screen when the listening host receives an ICMP Echo Request.

Update the script to detect when a SYN packet is received but the three-way handshake is not completed. Display a message that reads ** POSSIBLE PORT SCAN from x.x.x.x **.

Update the script to include the date & time of when the event occurred.

Write all displayed output to a local file named honeypot_log.txt.

YES ✓



Creating a Simple Modbus Honeypot

Prompt:

Create a Python script that listens on TCP 502.

Update the script to display a message to the screen when the listening host receives an ICMP Echo Request.

Update the script to detect when a SYN packet is received but the three-way handshake is not completed. Display a message that reads ** POSSIBLE PORT SCAN from x.x.x.x **.



Speaking of Honeypots

```
The Modbus honeypot service is listening on TCP 502...  
2024-08-04 17:10:45 - ** ICMP Echo Request from 10.2.1.226 **  
2024-08-04 17:10:46 - ** ICMP Echo Request from 10.2.1.226 **  
2024-08-04 17:10:47 - ** ICMP Echo Request from 10.2.1.226 **  
2024-08-04 17:10:48 - ** ICMP Echo Request from 10.2.1.226 **  
2024-08-04 17:11:12 - ** POSSIBLE PORT SCAN from 10.2.1.226 **
```



Speaking of Honeypots

```
The Modbus honeypot service is listening on TCP 502...  
2024-08-04 17:10:45 - ** ICMP Echo Request from 10.2.1.226 **  
2024-08-04 17:10:46 - ** ICMP Echo Request from 10.2.1.226 **  
2024-08-04 17:10:47 - ** ICMP Echo Request from 10.2.1.226 **  
2024-08-04 17:10:48 - ** ICMP Echo Request from 10.2.1.226 **  
2024-08-04 17:11:12 - ** POSSIBLE PORT SCAN from 10.2.1.226 **
```



Making Interactive Honeypots

Prompt:

Include a Modbus listener which provides coil and register values when requested. Returned values for some coils and registers should always be 0. Other values for specific coils and registers should provide random values. Display any received Modbus command to the screen.



Making Interactive Honeypots

```
c:\Scripts\Honeypot>vegas_honeypot.py
2024-08-04 20:37:40,707 Starting packet sniffer for TCP/502 and ICMP...
2024-08-04 20:37:40,708 Starting Modbus TCP Honeypot on port 502...
2024-08-04 20:37:40,708 Server listening.
2024-08-04 20:37:49,517 Received Read: Source IP=None, Function Code=1, Address=0,
Count=10, Values=[0, 1, 1, 1, 0, 1, 0, 1, 1, 0]
2024-08-04 20:38:00,665 Received Read: Source IP=None, Function Code=3, Address=0,
Count=10, Values=[22158, 10744, 42221, 59100, 30206, 16219, 17560, 43809, 18452, 60
438]
2024-08-04 20:38:26,172 Received Write: Source IP=None, Function Code=5, Address=0,
Values=[True]
2024-08-04 20:38:26,173 Received Read: Source IP=None, Function Code=5, Address=0,
Count=1, Values=[True]
```



Making Interactive Honeypots

```
c:\Scripts\Honeypot>vegas_honeypot.py
2024-08-04 20:37:40,707 Starting packet sniffer for TCP/502 and ICMP...
2024-08-04 20:37:40,708 Starting Modbus TCP Honeypot on port 502...
2024-08-04 20:37:40,708 Server listening.
2024-08-04 20:37:49,517 Received Read: Source IP=None, Function Code=1, Address=0,
Count=10, Values=[0, 1, 1, 1, 0, 1, 0, 1, 1, 0]
2024-08-04 20:38:00,665 Received Read: Source IP=None, Function Code=3, Address=0,
Count=10, Values=[22158, 10744, 42221, 59100, 30206, 16219, 17560, 43809, 18452, 60
438]
2024-08-04 20:38:26,172 Received Write: Source IP=None, Function Code=5, Address=0,
Values=[True]
2024-08-04 20:38:26,173 Received Read: Source IP=None, Function Code=5, Address=0,
Count=1, Values=[True]
```



Making Interactive Honeypots

```
c:\Scripts\Honeypot>vegas_honeypot.py
2024-08-04 20:37:40,707 Starting packet sniffer for TCP/502 and ICMP...
2024-08-04 20:37:40,708 Starting Modbus TCP Honeypot on port 502...
2024-08-04 20:37:40,708 Server listening.
2024-08-04 20:37:49,517 Received Read: Source IP=None, Function Code=1, Address=0,
Count=10, Values=[0, 1, 1, 1, 0, 1, 0, 1, 1, 0]
2024-08-04 20:38:00,665 Received Read: Source IP=None, Function Code=3, Address=0,
Count=10, Values=[22158, 10744, 42221, 59100, 30206, 16219, 17560, 43809, 18452, 60
438]
2024-08-04 20:38:26,172 Received Write: Source IP=None, Function Code=5, Address=0,
Values=[True]
2024-08-04 20:38:26,173 Received Read: Source IP=None, Function Code=5, Address=0,
Count=1, Values=[True]
```

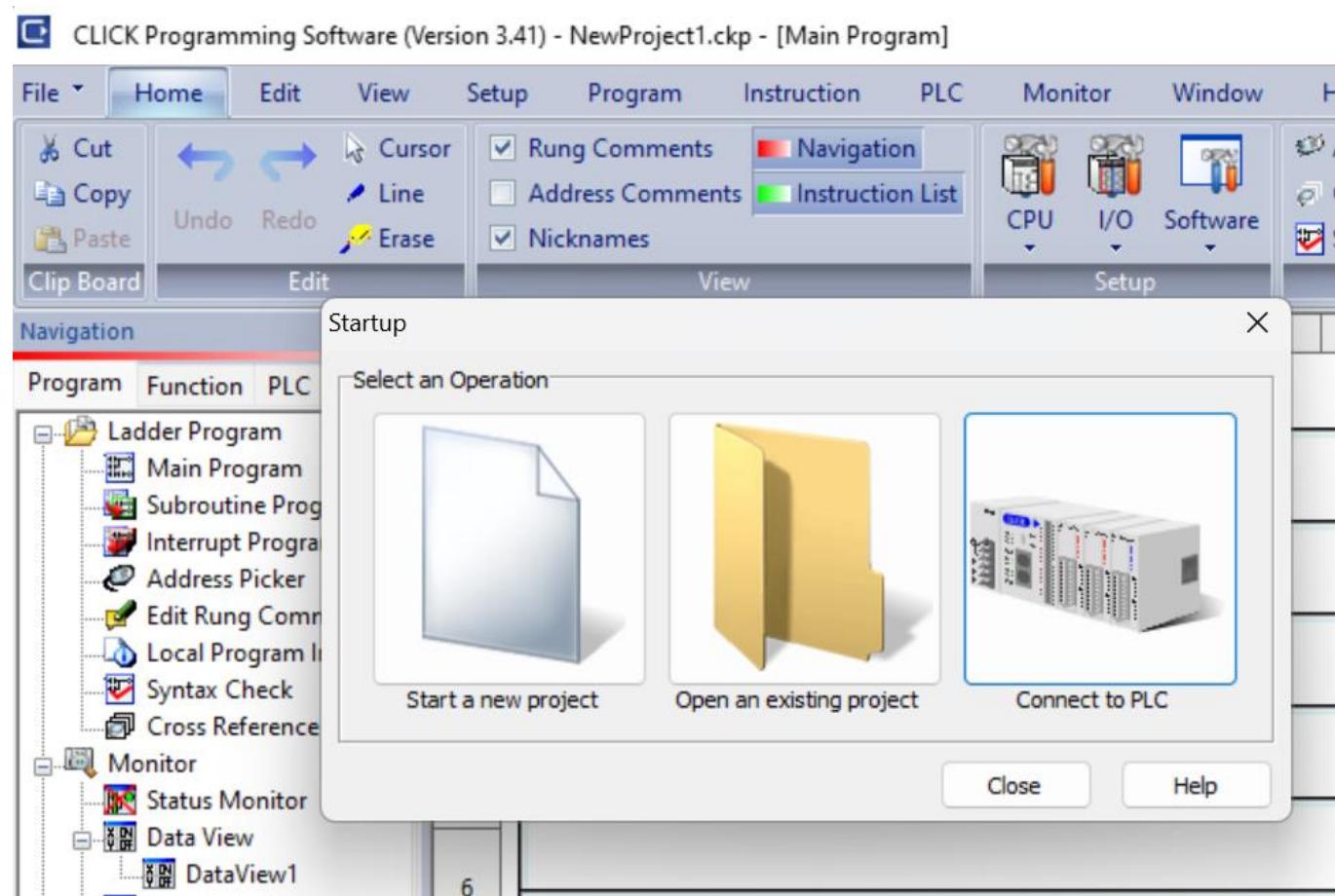


Going on the Offensive

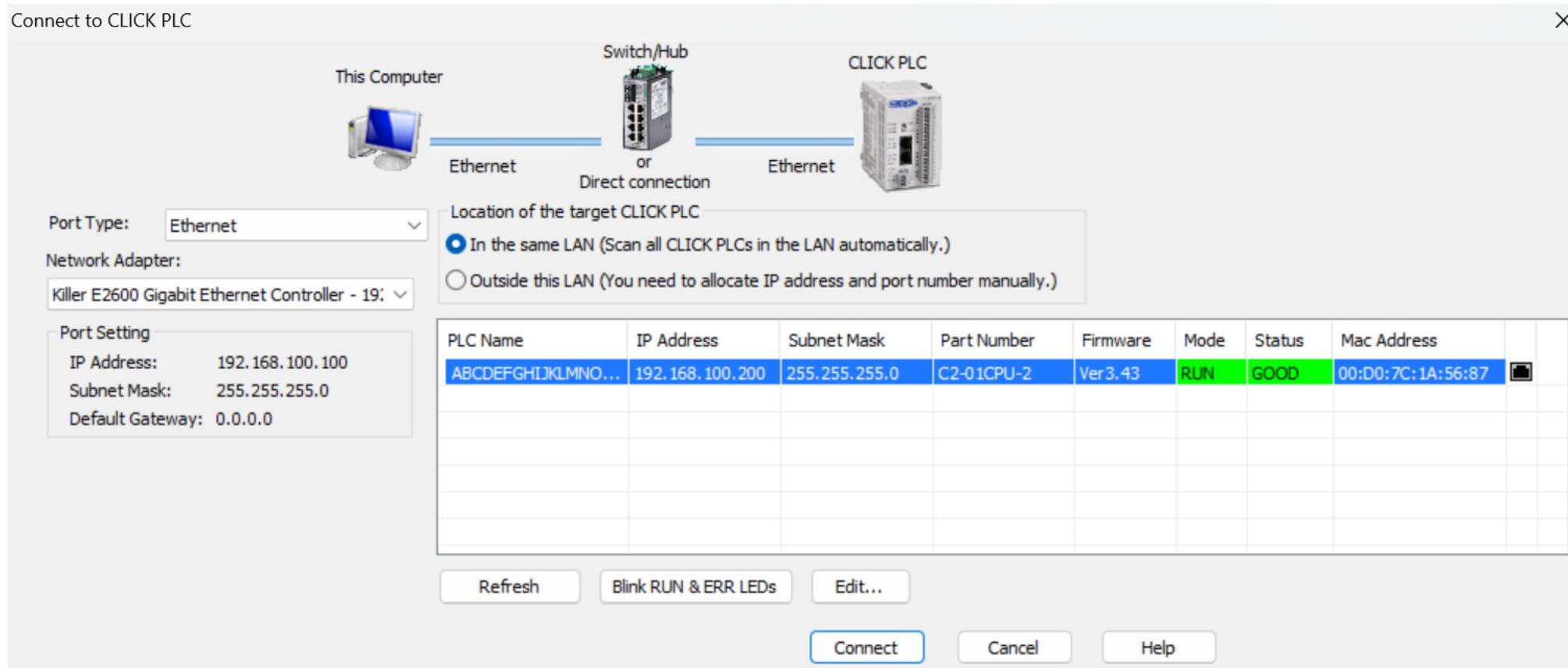
Initial Access	Execution	Persistence	Privilege Escalation	Evasion	Discovery	Lateral Movement	Collection	Command and Control	Inhibit Response Function	Impair Process Control	Impact
12 techniques	9 techniques	6 techniques	2 techniques	6 techniques	5 techniques	7 techniques	11 techniques	3 techniques	14 techniques	5 techniques	12 techniques
Drive-by Compromise	Change Operating Mode	Hardcoded Credentials	Exploitation for Privilege Escalation	Change Operating Mode	Network Connection Enumeration	Default Credentials	Adversary-in-the-Middle	Commonly Used Port	Activate Firmware Update Mode	Brute Force I/O	Damage to Property
Exploit Public-Facing Application	Modify Program	Modifying Command-Line Interface	Hooking	Exploitation for Evasion	Network Sniffing	Exploitation of Remote Services	Automated Collection	Connection Proxy	Alarm Suppression	Modify Parameter	Denial of Control
Exploitation of Remote Services	Module Firmware	Execution through API	Project File Infection	Indicator Removal on Host	Remote System Discovery	Hardcoded Credentials	Data from Information Repositories	Standard Application Layer Protocol	Block Command Message	Module Firmware	Denial of View
External Remote Services	Graphical User Interface	System Firmware	Valid Accounts	Masquerading	Remote System Information Discovery	Lateral Tool Transfer	Data from Local System		Block Reporting Message	Spoof Reporting Message	Loss of Availability
Internet Accessible Device	Hooking	Modify Controller Tasking		Rootkit	Program Download	Detected Operating Mode			Block Serial COM	Change Credential	Loss of Control
Remote Services	Native API	Scripting		Spoof Reporting Message	Wireless Sniffing	I/O Image			Data Destruction	Data Destruction	Loss of Productivity and Revenue
Replication Through Removable Media	User Execution				Valid Accounts	Monitor Process State			Denial of Service	Device Restart/Shutdown	Loss of Protection
Rogue Master						Point & Tag Identification			Manipulate I/O Image		Loss of Safety
Spearphishing Attachment						Program Upload			Modify Alarm Settings		Loss of View
Supply Chain Compromise						Screen Capture			Rootkit		Manipulation of Control
Transient Cyber Asset						Wireless Sniffing			Service Stop		Manipulation of View
Wireless Compromise									System Firmware		Theft of Operational Information



I Love My CLICK PLC

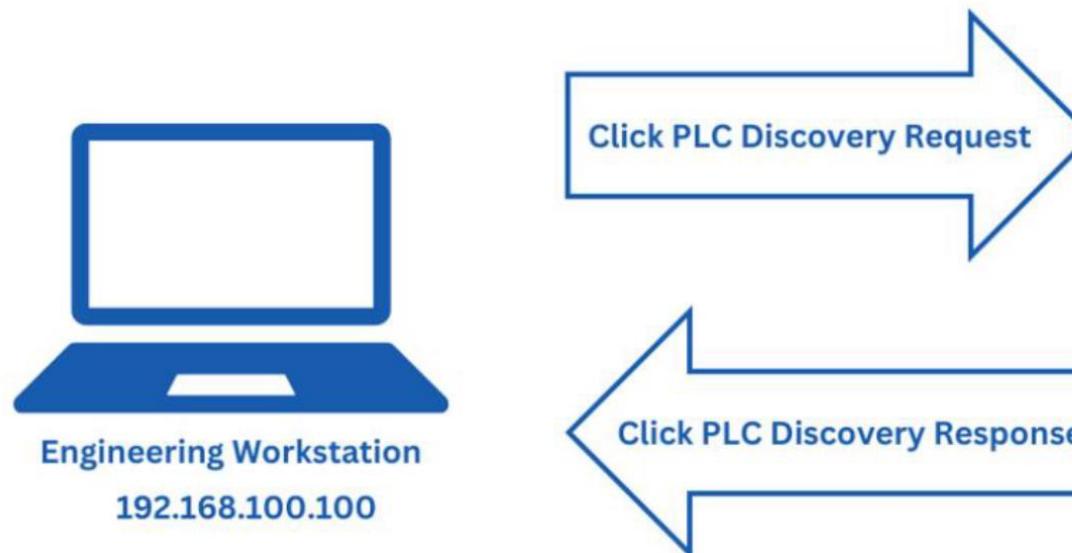


I Love My CLICK PLC



CLICK PLC Discovery Process

Source	Destination	Protocol	Leng	Info
192.168.100.100	255.255.255.255	UDP	56	2770 → 25425 Len=14
192.168.100.200	255.255.255.255	UDP	115	25425 → 2770 Len=73
192.168.100.100	255.255.255.255	UDP	56	2770 → 25425 Len=14
192.168.100.200	255.255.255.255	UDP	115	25425 → 2770 Len=73
192.168.100.100	255.255.255.255	UDP	56	2770 → 25425 Len=14
192.168.100.200	255.255.255.255	UDP	115	25425 → 2770 Len=73



Click Plus PLC
192.168.100.200



Discovery Response Packet

0000	ff ff ff ff ff ff 00 d0	7c 1a 56 87 08 00 45 00 -V-E-
0010	00 65 20 21 00 00 40 11	34 f7 c0 a8 64 c8 ff ff	-e ! @ 4 . d ..
0020	ff ff 63 51 0a d2 00 51	19 ca 4b 4f 50 eb 01 00	.cQ Q . KOP ..
0030	4d 09 3f 00 45 01 66 c1	10 00 01 00 c0 a8 64 c8	M ? E f d .
0040	ff ff 00 00 00 00 00 00	00 d0 7c 1a 56 87 00 00 -V- .
0050	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0060	00 00 00 00 00 00 00 00	00 00 00 00 c7 03 2b 00 + .
0070	80 00 01		...



Parsing the Response Packet

Prompt:

Create a Python script that sends a broadcast UDP packet containing a payload of 4b4f50000100edd5040045016680 to UDP port 25425 from UDP port 2770. Ask the user which interface to send the broadcast packet on. Send the packet from the specific IP address on the network interface card. Listen for responses and display any responding IP addresses on the screen.



Parsing the Response Packet

```
c:\Scripts>broadcast.py
WARNING: Wireshark is installed, but cannot read manuf !

Available network interfaces:
0: Local Area Connection (169.254.102.124)
1: Killer Ethernet Adapter (192.168.100.100)
2: Ethernet (192.168.56.1)
3: Local Area Connection* 1 (169.254.167.164)
4: Local Area Connection* 2 (169.254.50.84)
5: VMware Network Adapter VMnet1 (192.168.159.1)
6: Wi-Fi (10.2.1.252)
7: Bluetooth Network Connection (169.254.65.109)
8: Loopback Pseudo-Interface 1 (127.0.0.1)

Select the network interface to use (number): 1

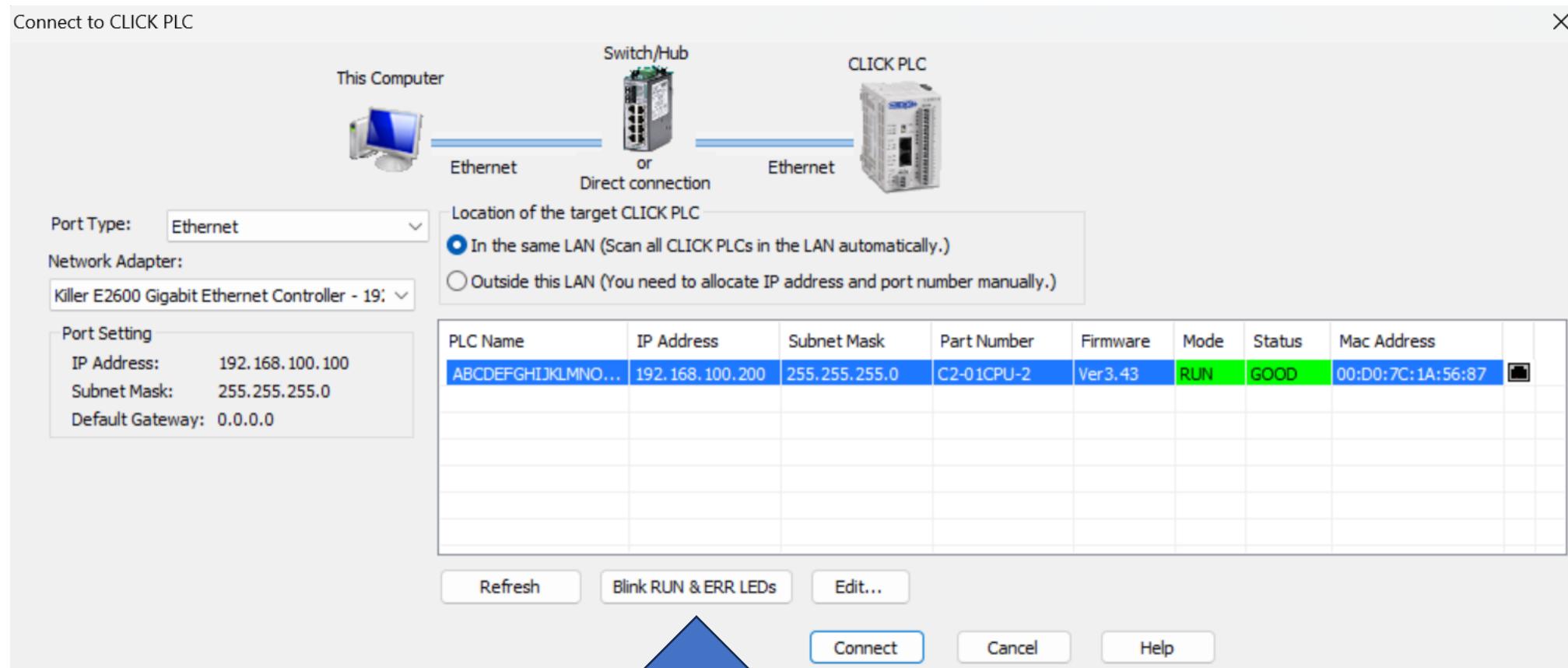
Sending UDP broadcast packet to 255.255.255.255 using interface Killer Ethernet Adapter from 192.168.100.100...
WARNING: Mac address to reach destination not found. Using broadcast.

Listening for responses...

Received responses from:
=====
192.168.100.200
```



What Other Options?



Blinky! Blinky!

Prompt:

Create a Python script that asks for an IP address and send the following payload

'4b4f500001006a570e0045016643c0a864c800d07c1a5687' to that IP address. The data must be sent over UDP to destination port 25425.

```
c:\Scripts\ChatGPT_Intro>click_blink.py
Enter the target IP address: 192.168.100.200
Payload sent to 192.168.100.200:25425
```



Change IP Address?

Connect to CLICK PLC

This Computer

Port Type: Ethernet

Network Adapter: Killer E2600 Gigabit Ethernet Controller - 19:

Port Setting

IP Address:	192.168.100.100
Subnet Mask:	255.255.255.0
Default Gateway:	0.0.0.0

PLC Network Information

PLC Name: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Obtain Address from DHCP

Use default fixed address

Use the following IP Address

IP Address: 192 . 168 . 100 . 205

Subnet Mask: 255 . 255 . 255 . 0

Default Gateway: 0 . 0 . 0 . 0

Part Number: C2-01CPU-2

MAC Address: 00:D0:7C:1A:56:87

Hardware Mode Status Mac Address

3.43	RUN	GOOD	00:D0:7C:1A:56:87
------	-----	------	-------------------

OK Cancel Help



Read and Write Coils & Registers?

Prompt:

Create a script that asks the user for the IP address of a PLC to connect to, the register number to update and the value to. Connect to the PLC and update the target register with the value provided. Afterwards, read the value for the specified value and the following 9 values. Display on the screen.

Read and Write Coils & Registers?

```
c:\Scripts>update_register.py
Please enter the PLC IP address: 192.168.100.200
Please enter the register number to update: 1
Please enter the value to write to the register: 1010
Successfully wrote 1010 to register 1
Register 1: 1010
Register 2: 0
Register 3: 0
Register 4: 0
Register 5: 1616
Register 6: 0
Register 7: 0
Register 8: 0
Register 9: 0
Register 10: 0
```



Pulling It All Together

```
ClickSploit
```

```
=====
```

- 1. Scan for Click PLCs
- 2. List PLC information
- 3. Flash LED
- 4. Update IP address
- 5. Read/Write coils
- 6. Read/Write registers
- 7. Change CPU state
- 8. Change PLC name

- X. Exit

Specify the IP address of a Click PLC or '1' to launch a scan:



What Comes Next?

For Defenders?

For Attackers?

FrostyGoop Anyone?



Just Remember...

**GenAI today is the
WORST it is ever going to
be.**



Thank You!

□ Please don't hesitate to reach out!

Mike Holcomb

[linkedin.com/in/mikeholcomb](https://www.linkedin.com/in/mikeholcomb)

mikeholcomb.com

mike@mikeholcomb.com

[youtube.com/@utilsec](https://www.youtube.com/@utilsec)

github.com/utilsec

Documented
Walkthrough
On GitHub!

