**Using ChatGPT to Write Defensive & Offensive Scripts for ICS/OT**

**Mike Holcomb**

**GRID, CISSP, ISA 62443**

**mikeholcomb.com**

Contents

[Using ChatGPT to Write Defensive & Offensive Scripts for ICS/OT 1](#_Toc164436640)

[Lab Requirements 3](#_Toc164436641)

[Exercise 1.1 Installing Python on Windows 3](#_Toc164436642)

[Exercise 1.2 Creating Your First ChatGPT Python Script 3](#_Toc164436643)

[Exercise 1.2 Updating an Existing Script 4](#_Toc164436644)

[Exercise 1.3 Converting Python Scripts to Other Languages 5](#_Toc164436645)

[Exercise 1.4 Creating a Basic ICS/OT Port Scanner 5](#_Toc164436646)

[Exercise 1.5 Troubleshooting Error Code 6](#_Toc164436647)

[Exercise 1.6 Adding Functionality to an Existing Script 7](#_Toc164436648)

## Lab Requirements

* Windows host
* Virtualization software (e.g., VMware, Oracle VirtualBox, Hyper-V)
* Kali Linux
* A paid for ChatGPT account is preferred
  + A free ChatGPT can work but will be limited in some cases
* Lab setup

## Exercise 1.1 Installing Python on Windows

In this exercise, you will install the current version of the Python language on your Windows system. Python will be the main language used to write scripts in this course as it can run on both Windows and Linux.

1. On your main Windows host, download the current version of Python at <https://www.python.org/downloads/>. You should see a yellow button near the top of the page under the header “Download the latest version for Windows.”
2. Once downloaded, install Python. For the purpose of this course, accept all defaults.
3. To verify that Python is installed correctly on your system, open a command prompt and type the following:

**python --v**

1. The screen should display output something similar to the following:

**C:\Users\micha>python --version**

**Python 3.12.2**

## Exercise 1.2 Installing PIP

While PIP comes installed with Python 3.4 or higher, it is important to ensure that it is installed on your system. PIP allows additional modules of Python functionality to be added easily to the system. Many of the scripts we will be creating in this course will require such modules to be installed and PIP is the easiest way to make that happen!

1. Run the following command to see if PIP is installed:

**python -m ensurepip --default-pip**

If you receive a message that starts with “Requirement already satisfied” then you do not need to take any further action.

1. If it appears that PIP is not installed on your system, download the current version of PIP using the following link: https://bootstrap.pypa.io/pip/pip.pyz.
2. Once downloaded, launch the .pyz file which is a specially crafted ZIP file for Python. At this point, PIP should install on your system.

## Exercise 1.2 Creating Your First ChatGPT Python Script

1. Login to your ChatGPT account at https://chat.openai.com.
2. Copy the following prompt into the ChatGPT window and click the up arrow icon (“Send Message”) to process.

Prompt:

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

1. Once the prompt is processed, ChatGPT should display a code window with the script. Click the ‘Copy code’ button in the upper right corner of the code window to copy the script to your Windows clipboard.
2. In a folder on your Windows system, such as C:\Scripts, create a text file names hello.py. Edit the contents, paste in the code from ChatGPT and save the file.
3. Once saved, run your Python script from the Windows command link.

**C:\Scripts\hello.py**

1. If run successfully, you should see the following on your screen:

A black screen with white text

Description automatically generated

1. Now we have our first running Python script written by ChatGPT.

## Exercise 1.2 Updating an Existing Script

Often, we might want to make updates to an existing script. In this case, we want to add a blank line before the script writes text to the screen to make it easier to read.

1. Copy the following prompt into the ChatGPT window and click the up arrow icon (“Send Message”) to process.

Prompt:

**Update the previous script to include a blank line before writing to the screen.**

1. Note the output of the new script. ChatGPT added the **print()** command to add a blank line.
2. Update your hello.py script with the new code and run it. You should see the “easier to read” output:

A black background with white text

Description automatically generated

## Exercise 1.3 Converting Python Scripts to Other Languages

While writing Python scripts can be useful, Python is not installed on all Windows systems by default. While Python can be manually installed, as we’ve done previously, this might not always be an option, especially in offensive operations. If possible, offensive capabilities prefer to use Living off the Land techniques to use the default capabilities of a system rather than install any additional software which might be detected by defenders.

In this exercise, we’ll convert the previous Python script to PowerShell which is installed by default on Windows to leverage Living off the Land techniques.

1. In ChatGPT, submit the following prompt to convert the previous Python script to its PowerShell equivalent.

NOTE: Some times ChatGPT will provide details on what it is doing for you. If you want more, or less, details, you can simply let us know by using “Provide more (or less) details on what you are doing.”

1. Save the new script to hello.p1 and run it. You should see very similar output compared to the previous Python script, including the extra line we added for readability.

A black background with white text

Description automatically generated

## Exercise 1.4 Creating a Basic ICS/OT Port Scanner

Text

1. Use the following script to create a basic network scanner in Python to detect the presence of common ICS/OT ports on a specific host. We will be continuing to add functionality to the scanner.
2. In ChatGPT, submit the following prompt to create a basic ICS/OT port scanner. The scanner will check for the presence of open ports on a single IP address. The open ports checked for will be TCP 80, TCP 443, TCP 102, TCP 502.

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 or 502 are open on the target system.**

**Provide an organized list of each port and their status.**

1. Save the output as scanner.py and run the script. Conduct a port scan against a known IP address.
2. Convert to PowerShell and save as scanner.ps1.

## Exercise 1.5 Troubleshooting Error Code

1. Run scanner.ps1.

When you do, you will receive an error message similar to the one seen here:

A screen shot of a computer

Description automatically generated

1. Troubleshooting error code can be a challenging process. Thankfully, ChatGPT does an exceptional job troubleshooting code errors. Especially ones that it’s created!
2. To troubleshooting the PowerShell scanner script, copy the red error message into ChatGPT and submit it as a prompt. Once submitted, ChatGPT should find the issue, explain the problem and how to fix it.
3. It should also provide a fixed script or a fixed script portion. In this example, ChatGPT only returns the single line that needs to be fixed which it has corrected:

A black and white rectangle with white text

Description automatically generated

1. If it only returns the portion that needs to be fixed and not the entire script, you can simply use the following prompt:

Prompt:

**Provide the full fixed script.**

For the sake of space, the fixed script is not included here.

1. Update your scanner.ps1 file and run it to make sure it runs just as the Python version does now that it is fixed.
2. As you can see, while the functionality is the same, the output is different with the PowerShell version.

A screenshot of a computer program

Description automatically generated

## Exercise 1.6 Adding Functionality to an Existing Script

While the port scanner has basic functionality, we could add additional features. For example, we might want to add the ability to scan an entire subnet.

1. Use the following prompt to update the port scanner Python script to include the ability to scan a subnet in addition to single IP addresses.

Prompt:

**Update the previous script to allow the user to provide a single IP address or subnet with CIDR notation.**

1. Update your scanner.py script with the updated script provided by ChatGPT.
2. Run the script and run a scan against your local subnet.

As you can see, your port scanner isn’t necessarily the fastest, but it does work!

1. Consider what other updates you might make to the script to make it more useful. Some items to consider include:

* Adding ‘help’ output
* Error handling

## Exercise 1.7 Creating a Scrapper for Harvesting IP Addresses from Web Sites

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

1. Use the following prompt

Prompt:

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

1. In your browser, visit the IP test page at <https://www.mikeholcomb.com/iptest> to see a simple page created to supply several IP addresses for testing.
2. Run the script. Use it to connect to the test page at <https://www.mikeholcomb.com/iptest>.

Your script should only parse and display valid IP addresses.

A screen shot of a computer

Description automatically generated

Depending on how the script runs and parses IP addresses, you might see an order that is different than what is on the web page.

1. Update the script to display IP addresses in numerical order. You should now see the following output:

A screen shot of a computer

Description automatically generated

If you need a little help, use the following to have the scraper.py display the IP address list in numerical order.

Prompt:

**Update the previous script so that the gathered IP addresses are listed in numerical order.**

Of course, there are many different ways to try. If you’re able to tell ChatGPT what you want, it “should” be able to understand. We’ll see some examples as we go throughout the exercises of when ChatGPT can and cannot understand necessarily.

## Exercise 1.8 Adding GeoIP Location Information with API Integration

1. Let’s update our scraper.py script to add the functionality for performing GeoIP location lookups for each IP address and displaying it on the screen. For the ability to perform GeoIP lookups, we will use the free subscription plan for the **ipinfo.io** service.
2. Access the website at ipinfo.io.
3. On the main page, you should see the GeoIP information displayed for your own public IP address.

A screenshot of a computer program

Description automatically generated

Hello from Simpsonville, SC!

1. Sign up for a free account at ipinfo.io.
2. Once signed in, click on the ‘Token’ option on the left-hand menu. This will take you to the page that lists your API token for the ipinfo.io service. Leave the page open as we’ll be referring to it shortly.
3. Back in ChatGPT, let’s tell it to update our previous scraper.py script with the ipinfo.io API functionality to perform GeoIP lookups and display the information on the screen.

Prompt:

**Update the previous script to perform a GeoIP location lookup for each IP address using API functionality with ipinfo.io. Display the GeoIP location along with each IP address.**

1. ChatGPT should provide you with an updated script along with instructions on how to update the script with your ipinfo.io API token. Make the specified adjustments and save your script.

For example, in my version, I had to find the API\_TOKEN value and replace it with my actual API token listed on the ipinfo.io Token page we looked at earlier.

Before updating the API\_TOKEN value:

**def get\_geoip\_location(ip\_address):**

**try:**

**# Send a GET request to ipinfo.io API with the access token**

**response = requests.get(f"https://ipinfo.io/{ip\_address}/json?token={API\_TOKEN}")**

After updating the API\_TOKEN value:

**def get\_geoip\_location(ip\_address):**

**try:**

**# Send a GET request to ipinfo.io API with the access token**

**response = requests.get(f"https://ipinfo.io/{ip\_address}/json?token={ b40068db3857}")**

1. Run your updated version of the scraper.py script with GeoIP lookup functionality. You should see something similar to the following.

A screen shot of a computer

Description automatically generated

## Exercise 1.9 Finding PLCs Exposed to the Internet

While there are several ways to find Internet exposed PLCs, our friends at the NSA gave us a few methods for doing so in their ELITEWOLF project. While the project provided several Snort intrusion detection signatures for ICS/OT defenders to identify potentially malicious activity on their networks, the signatures also provided us with some Google searches to find PLCs.

1. Access the ELITEWOLF project site at <https://github.com/nsacyber/ELITEWOLF>.
2. Select the ‘ELITEWOLF\_SNORT\_AllenBradley\_RockwellAutomation.txt’ file.
3. Looking at the last part of each Snort rule, the URL portion specified in the content: section can be used in Google to find Allen Bradley/Rockwell Automation PLCs exposed to the Internet.

A screen shot of a computer code

Description automatically generated

1. Go to Google and run a search for the first URL portion listed in the ELITEWOLF file as seen below.

A close up of a logo

Description automatically generated

1. While some results will refer to PLC documentation and the ELITEWOLF Github page itself, most results will refer to actual PLCs exposed to the Internet. For example, my search results start with the two exposed PLCs at 173.181.149.217 and 185.9.25.82 as seen below.

A screenshot of a computer

Description automatically generated

As expected, based on the ELITEWOLF file we pulled the search string from, these appear to be AllenBradley/Rockwell Automation PLCs.

1. Click on one of the links for the exposed PLCs. You should see a web page that displays various statistics related to the PLC’s TCP functionality such as seen below.

A screenshot of a data

Description automatically generated

1. While each of the URLs can provide interesting pieces of information, I’m always fascinated when I can see the netstat output of a remote system. Run a Google search using the partial URL of ‘/rokform/advancedDiags?pageReq=tcpconn’.
2. Access the URL. You should see the exposed PLCs netstat information, showing all IP addresses (external AND internal) that are communicating with the PLC.

A screenshot of a table

Description automatically generated

1. In the previous example, there are several items of note:

* The internal IP address of the PLC is 192.168.1.100.
* The public IP address is the same as the IP in the URL.
* The PLC is communicating with multiple internal AND external IP addresses over HTTP (TCP port 80) and EthernetIP (TCP port 44818).

1. Run the scraper.py against the **tcpconn** page you had visited.

A screen shot of a computer

Description automatically generated

1. In the sample above, we can note that:

* The PLC is being remotely accessed from Shenyang in China, Sao Paulo in Brazil, Brussels in Germany and (of course) Simpsonville, South Carolina, United States.
* Internal hosts communicating with the PLC include 192.1168.1.100,192.168.1.122,192.168.1.125, 192.168.1.127,192.168.1.130 and 192.168.1.131.

So it appears we are not the only external visitors to the PLC’s web interface. Not only that, but we have also enumerated more than a few internal hosts on the PLCs ICS/OT network.