**System Enumeration: Executive Summary**

Date: 11/30/2022

Team: Darknet

Lab Description

For this lab we are performing scanning and enumeration on two hosts in our environment, one of the hosts is a vulnerable Linux machine called Metasplotable and the second is an outdated windows 7 device. We will be creating python scripts that leverage security tools to scan for vulnerabilities and hardening some of these vulnerabilities with python scripts. The tools that we are leveraging include OpenVAS and Nikto.

Lab Results

The results of the scanning and enumeration were quite alarming, on the Linux devices we detected an alarming number of ports open and services being out of date. On the Metasplitable host, we found major ports open such as port 22 (SSH) and port 25 (SMTP) they were also running outdated versions of these protocols making them extremely vulnerable points of entry for potential attackers. We also found PHP vulnerabilities and TCP header vulnerabilities as well as insecure web traffic going through port 80 (HTTP).

Problem Description

The issue that we are overcoming is that we have devices in our environment that have many vulnerabilities that can give an attacker access to our environment. In order to overcome this issue, we are creating python scripts to find and patch these vulnerabilities..

Accomplishments

In this blue teaming exercise, we were able to create scripts that did web scanning and port scanning on the hosts in our environment as well as scripts for path-specific vulnerabilities. This will help us to automate our process of scanning and enumerating our environment's vulnerabilities.

Results

*Detailed results.*

* *Windows 10 scanning (OpenVAS Script)* ***Fransis***
* *Linux Scanning (Port Scanning)* ***Joseph***
* *Linux Scanning (Vuln Scanning “Nikto Script”)* ***Joseph***
* *Windows 7 hardening (script)* ***Isaac***
* *Metasploitable Hardening (script) -* ***Ny***

Team Bio

**Windows 10 scanning (OpenVAS Script***)*

*Host: Windows 10 address 192.168.169.75*

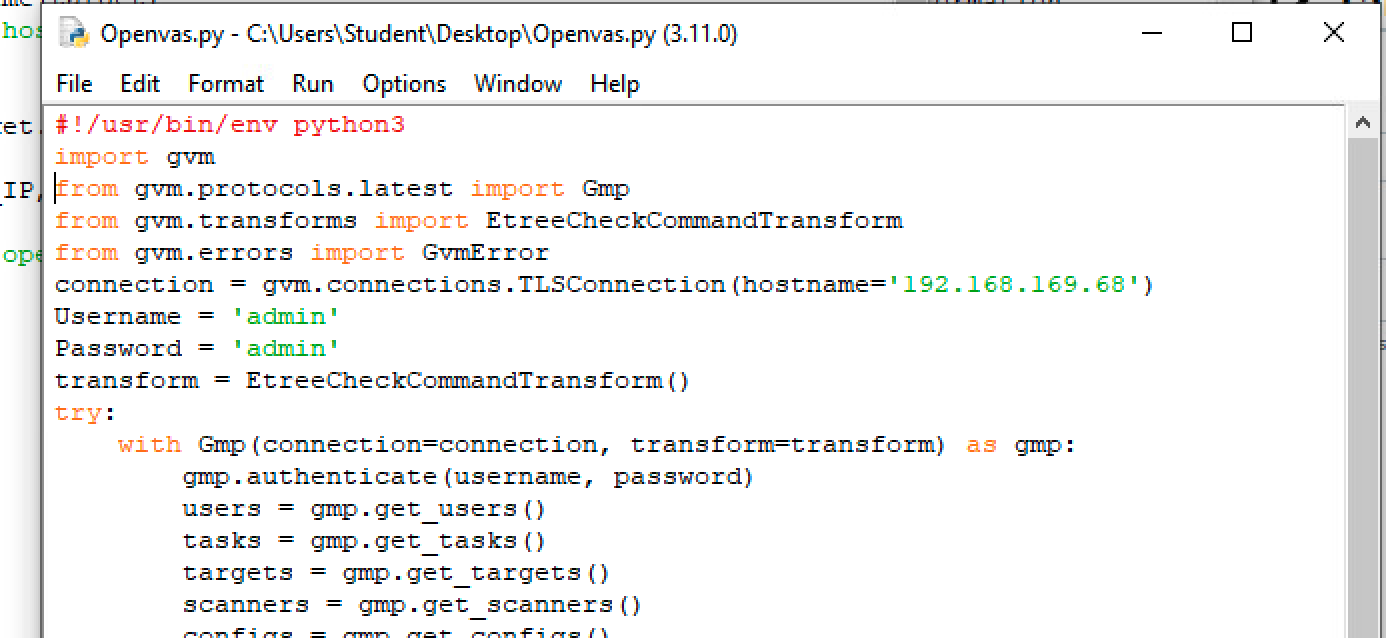
*Target: Windows 7 address 192.168.169.57*

*Vulnerbilites: Port 443 and Port 137 are open can be exploit using SMB vulnerabilities.*

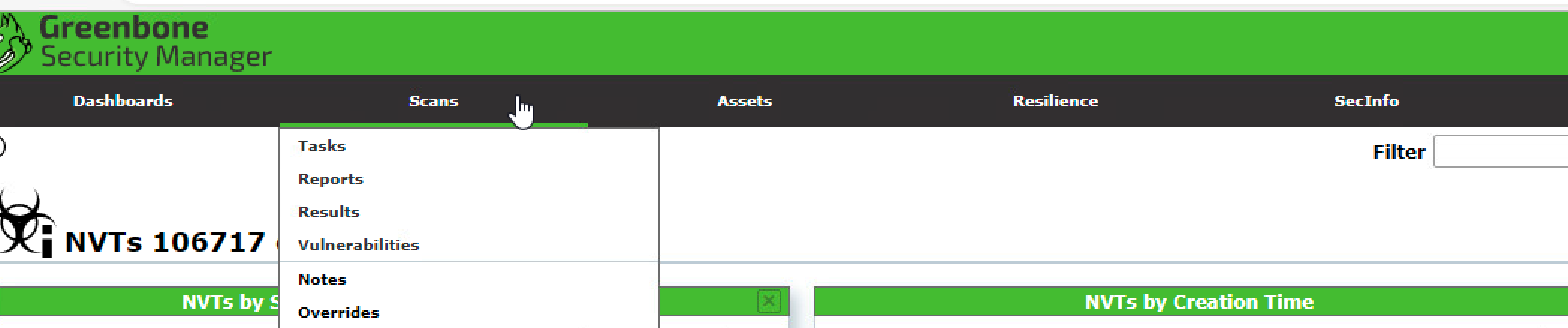
In this window 10 scanning, will be using OpenVAS, and Open port scanning to perform vulnerabilities scan. We built launch an OpenVAS server in a VM, and access it as interface. As we can see, OpenVAS can only be accessed by web interface. And it will perform access to webserver and use the scanning tools in the web. And we can see the results only in the webservers.

Here is the initial script to call out all the modules are needed to perform the scan. However, it requires downgrade modules and Visual studios C++ install in order to import gvm modules on Windows. Greenbone (aka Openvas ) uses python-gvm (gvm module) to collect APIs that help with remote controlling Greenbone Community Edition installations and Greenbone Enterprise Appliances including Openvas.

After that, we need to put the credential to access the vulnerability scanner, credential such Host address, Username, and Password.

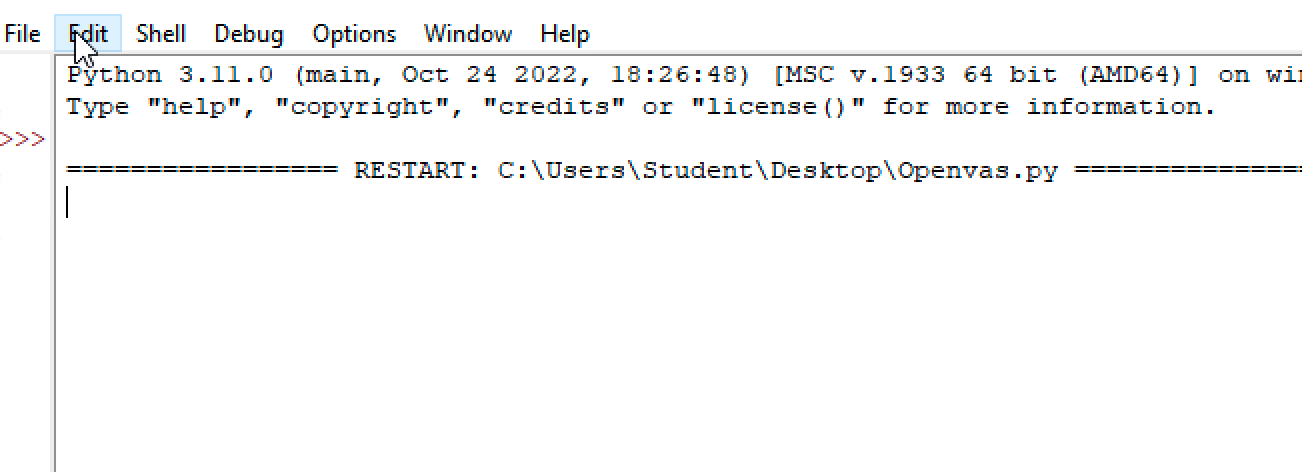


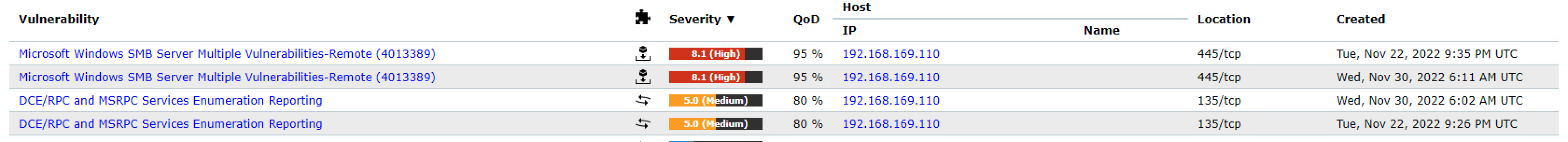
Below are the tasks that perform in the the Opeven that matches the Options bar in the web interface.



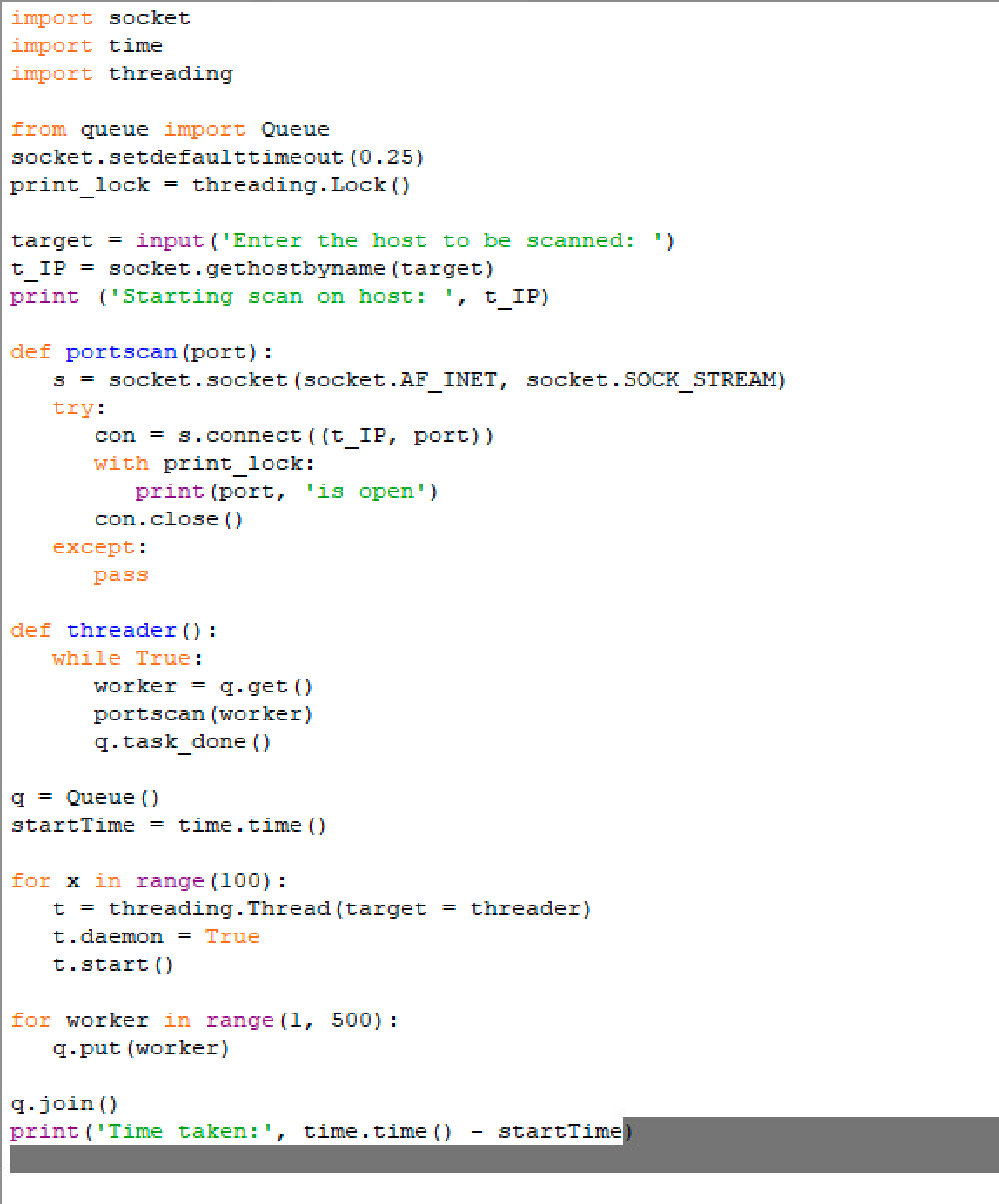


After it launched, the methods will not show because the gvm is not compatible with windows, missing module on Windows. However, the result will still run and shown in the OpenVAS system. (it needs manuelly to put the target in the OpenVas Web in advanace)

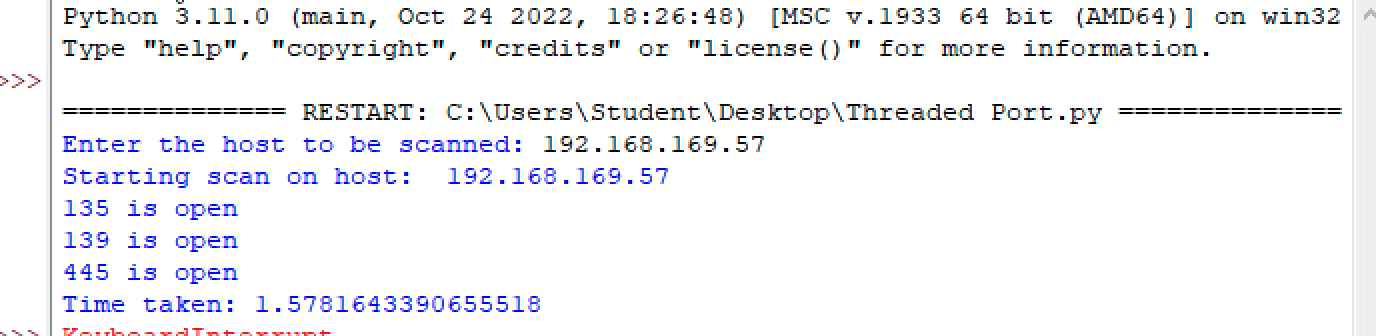


For another Port scanning gives the same results:

ThreadsPort.py is a script to scann the targets vulnerabilities if there are any available vulnerabilities and open ports.



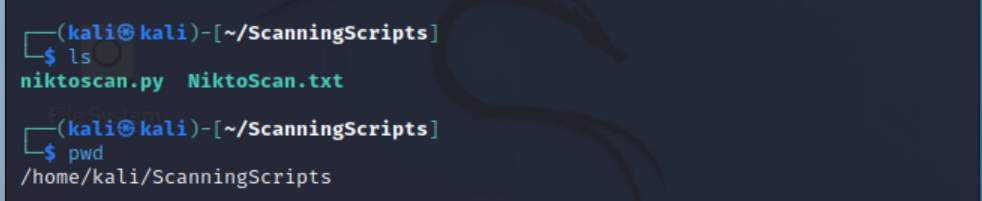
As a result, it has similar results as the openvas. Attackers can use these techniques to explore targets.



**Web Scanning Metasploitable**

In order to accurately assess the vulnerabilities that are found within the Metsaplotable host we decided to write a python script that would leverage the Nikto scanning tool that is already installed natively on the Kali Linux distribution on Linux. The script that we wrote using python three interacts with the user and identifies the target, then automatically scans that host and dumps the results into a text file to be read and referenced at convenience. I will provide the screenshots of this scan is completed.

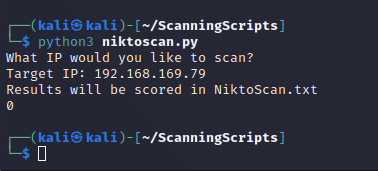
The python script is titled niktoscan.py and is in the folder /home/kali/ScanningScripts on the Kali 02 host.



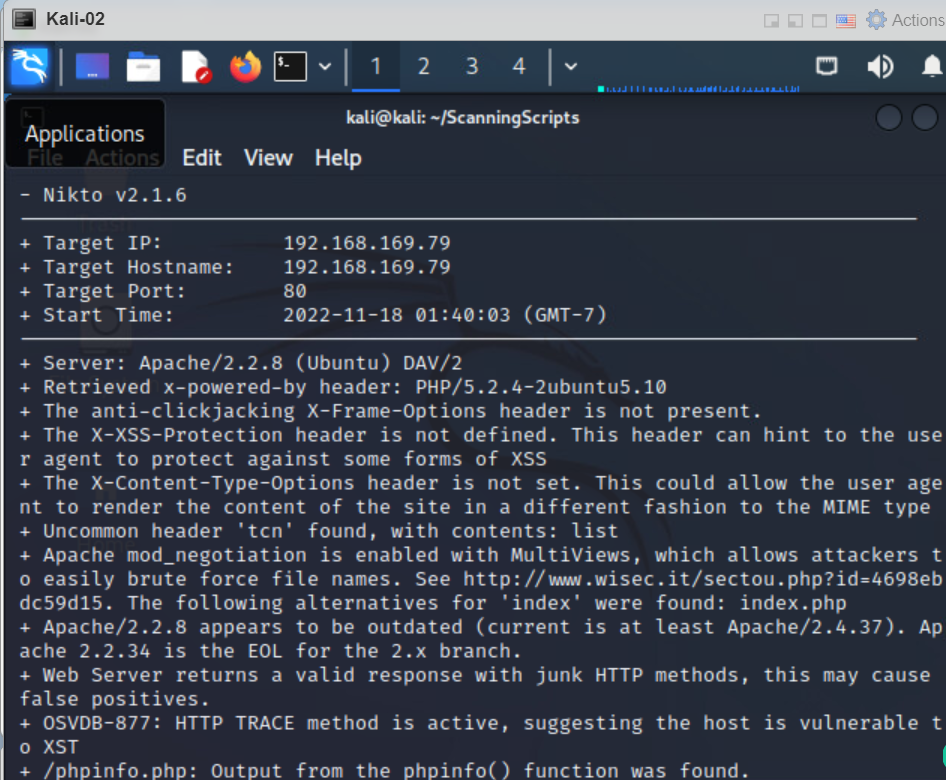
In the image below is a screenshot of the script with comments written into the code.



As you can see the script writes the results onto a text file titled NiktoScan.txt. This is displayed in the image below.



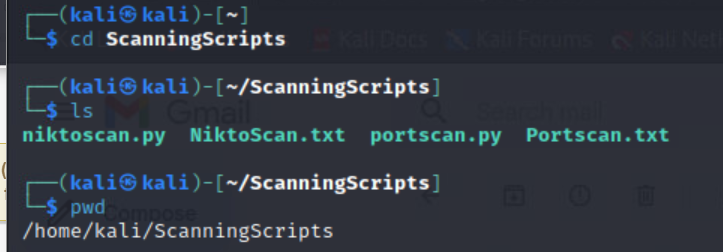
Now we will look at the results of the scan itself. First, we notice that this host is running an Apache web server. The version on this host is 2.2.8. The current version of Apache is 2.4.54. Out-of-date software is a major security risk and leaves this box exposed to attackers who could root the system with a Metasploit payload. Other vulnerabilities include TCP headers with content lists. Output from the function phpinfo() and even a phpMyAdmin directory. These vulnerabilities can leave a host extremely vulnerable, and it is in the best interest to harden and patch these weaknesses as soon as possible.



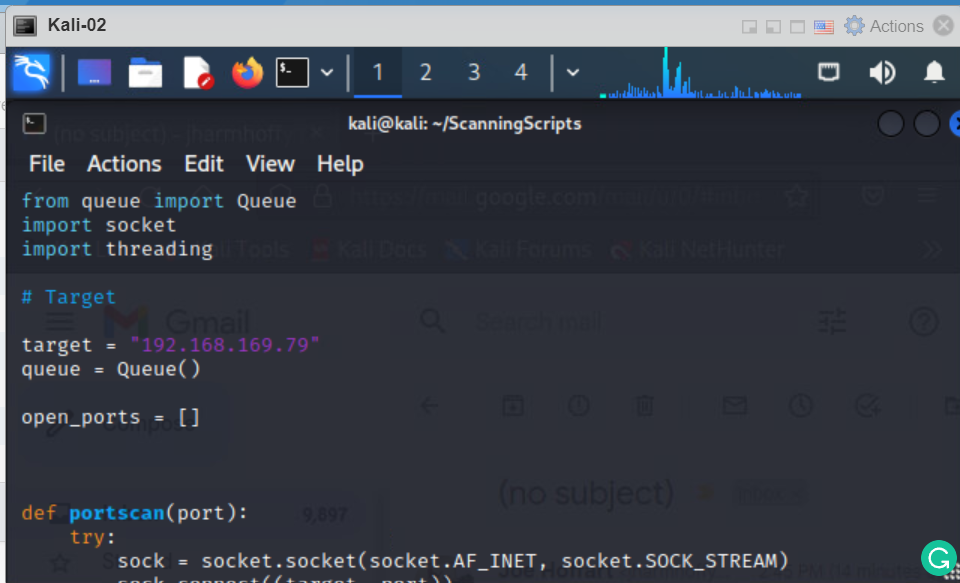
**Port Scanning Metasploitable**

For this next scan of the Metasploitable box we created a python script that scans a range of ports on the target machine. The script determines if the targets host ports are open or closed with the specified range and prints that data to the console. The python script then dumbs the open ports to a text file called Portscan.txt.

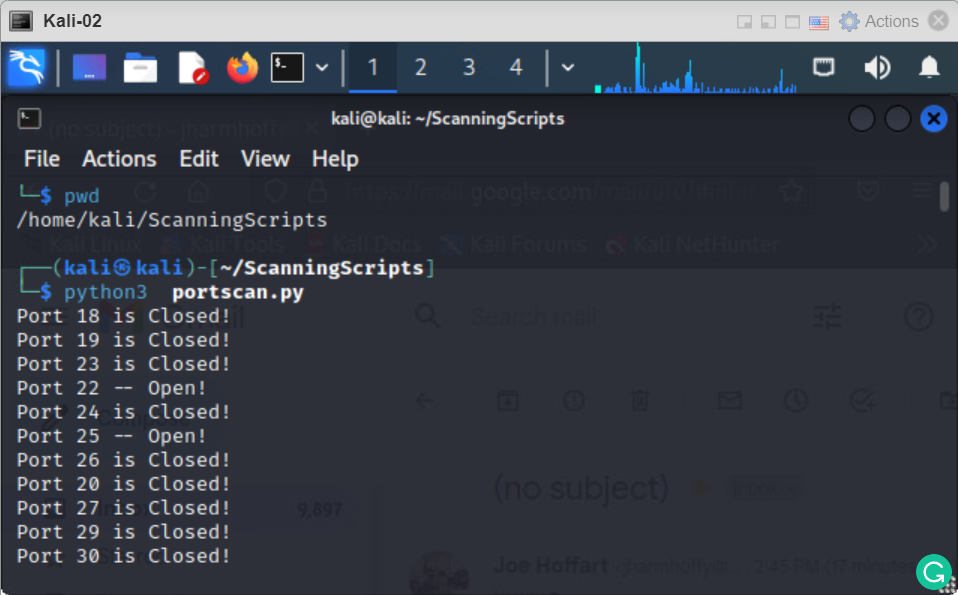
In the screenshot below we can see that our port scan script is in the /home/kali/ScanningScripts on the Kali 02 host.



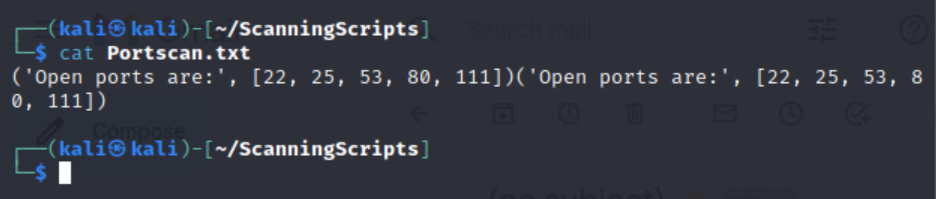
The screenshot below shows the part of the script opened in a text editor our kali machine, the target IP that is set is the IP of the target host, Metasploitable 02.



When the script is launched it instantly starts checking all of the ports within the specified range and prints to the console whether they are opened or closed.



Once the script is finished running it prints all of the open ports to a text file titled Portscan.txt. These results can be seen below.



After examining the results of this port scan, it has left us with a lot of valuable information. We see from the results that some key ports are left open for attackers to take advantage of an attempt to gain access to the Metasploitable 02 host. Port 22 is found opened and is responsible for remote connections to the device. This can be very dangerous because if the version of SSH is out of date attackers can gain remote access to the host. Port 25 is open as well with is responsible for email, the protocol is known as SMTP and finally, port 80 is open which runs HTTP and signifies that we have unsecured web traffic running from our target. Open ports can give an attacker great guidance for what doors to start knocking on and looking for a way into your environment.

**Hardening windows 7**

As we see in the previous report. We had three ports that are open which are 135, 139 and 445.

In PowerShell we can find the PIDs of those ports with the command "netstat –ano". And these PIDs do not change so we can run a python script to close them.

A screenshot of a computer

Description automatically generated with medium confidence

In this situation I made a code that will close those ports and I will run it every time my windows 7 machine restarts. I added the "print('hi')" at the end just to make sure my code ran.

So, to run the code I use PowerShell with admin privileges.

I only have to type down Python and the python script location and voila! My code closes those 2 ports.

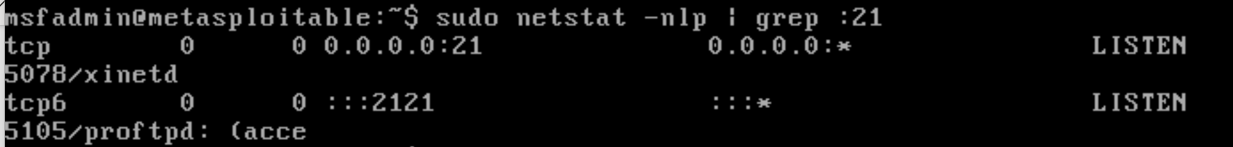
Graphical user interface, text

Description automatically generated

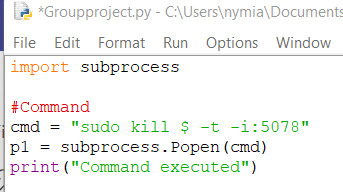
The PID 668 I closed it before hand, and it does not re open when I restart the c

**Hardening Metasploitable**

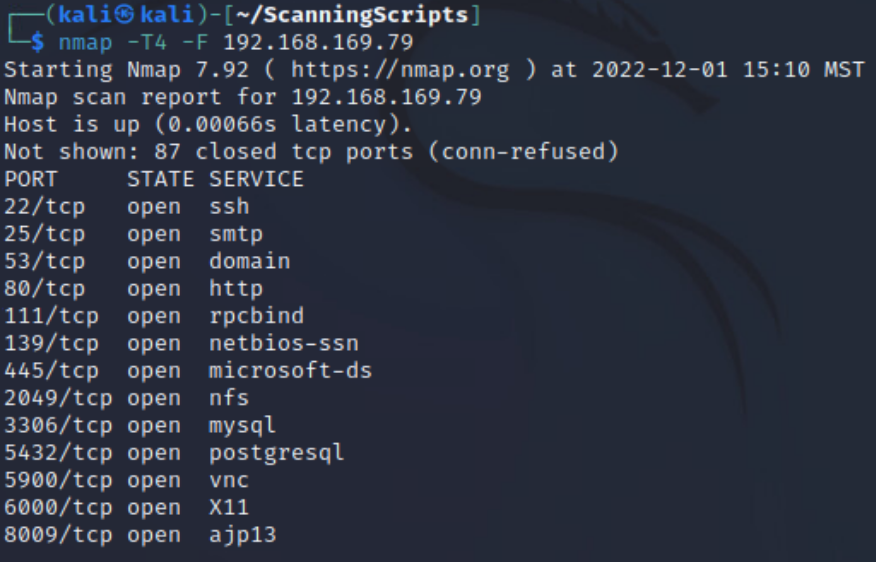
We have seen in the previous scan that we have several ports open. When accessing Metasploitable, we can see the PID of the open ports by typing on the Terminal: “sudo netstat –nlp | grep :[port number]”. We can see here that port 21’s PID number is 5078 and we will use this number in the Python script.



Since we now know the PID number, we can go ahead and close the ports by running a python script:



We can run the code for each open port in Metasploitable. So, to see if port 21 is still open, we can re-run the scan and we can see that port 21 is not listed anymore on the open ports’ list.



**All Scripts Written**

**Windows 10 scanning (OpenVAS Script***)*

#!/usr/bin/env python3

import gvm

from gvm.protocols.latest import Gmp

from gvm.transforms import EtreeCheckCommandTransform

from gvm.errors import GvmError

connection = gvm.connections.TLSConnection(hostname='192.168.169.68')

Username = 'admin'

Password = 'admin'

transform = EtreeCheckCommandTransform()

try:

with Gmp(connection=connection, transform=transform) as gmp:

gmp.authenticate(username, password)

users = gmp.get\_users()

tasks = gmp.get\_tasks()

targets = gmp.get\_targets()

scanners = gmp.get\_scanners()

configs = gmp.get\_configs()

feeds = gmp.get\_feeds()

nvts = gmp.get\_nvts()

print("Users\n------------")

for user in users.xpath('user'):

print(user.find('name').text)

print("\nTasks\n------------")

for task in tasks.xpath('task'):

print(task.find('name').text)

print("\nTargets\n-------------")

for target in targets.xpath('target'):

print(target.find('name').text)

print(target.find('hosts').text)

print("\nScanners\n-------------")

for scanner in scanners.xpath('scanner'):

print(scanner.find('name').text)

print("\nConfigs\n-------------")

for config in configs.xpath('config'):

print(config.find('name').text)

print("\nFeeds\n-------------")

for feed in feeds.xpath('feed'):

print(feed.find('name').text)

print("\nNVTs\n-------------")

for nvt in nvts.xpath('nvt'):

print(nvt.attrib.get('oid'),"-->",nvt.find('name').text)

except GvmError as error:

print('Error connection with server:', error)

**Web Scanning Script**

import os  
import subprocess  
  
# Function that defines the Nikto scan  
   
def nikto(host):  
  
 # Stores the open text file into the 'out' var  
  
 out = open('NiktoScan.txt', 'a')  
   
 # Stores the terminal command into thee 'cmd' var  
  
 cmd = ['nikto', '-h', host]  
  
 # Returns varibles  
  
 return subprocess.call(cmd, stdout = out)  
   
 return subprocess.call(cmd)  
   
 # close txt file  
  
 out.close()  
   
host = input('What IP would you like to scan?\nTarget IP: ')  
  
print ('Results will be scored in NiktoScan.txt')  
  
print(nikto(host))

**Port Scanning Script**

def portscan(port):  
 try:  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 sock.connect((target, port))  
 return True  
 except:  
 return False  
  
def get\_ports(mode):  
   
 for port in range(1, 1090):  
 queue.put(port)  
  
  
def worker():  
   
 while not queue.empty():  
 port = queue.get()  
 if portscan(port):  
 output = ("Port {} -- Open!".format(port))  
 print(output)  
 open\_ports.append(port)  
 else:  
 output2 = ("Port {} is Closed!".format(port))  
 print(output2)  
  
  
def run\_scanner(threads, mode):  
  
  
 x = open('C:\\Bin\\namp11.txt', 'w')  
  
  
 get\_ports(mode)  
  
 thread\_list = []  
  
 for t in range(threads):  
 thread = threading.Thread(target=worker)  
 thread\_list.append(thread)  
  
 for thread in thread\_list:  
 thread.start()  
  
 for thread in thread\_list:  
 thread.join()  
   
 results = ("Open ports are:", open\_ports)  
  
 print(results)  
  
 x.write(str(results))  
  
 x.close()  
   
  
  
run\_scanner(100, 1)

**Port Closing Script for Windows 7**

Import subprocess

cmd = 'taskill /pid 668 /F'

p1 = subprocess.Popen(cmd)

cmd = 'taskill /pid 1220 /F'

p1 = subprocess.Popen(cmd)

cmd = 'taskill /pid 4 /F'

p1 = subprocess.Popen(cmd)