Internal Network Penetration & Vulnerability Assessment

-Daniel Sheehan

Introduction

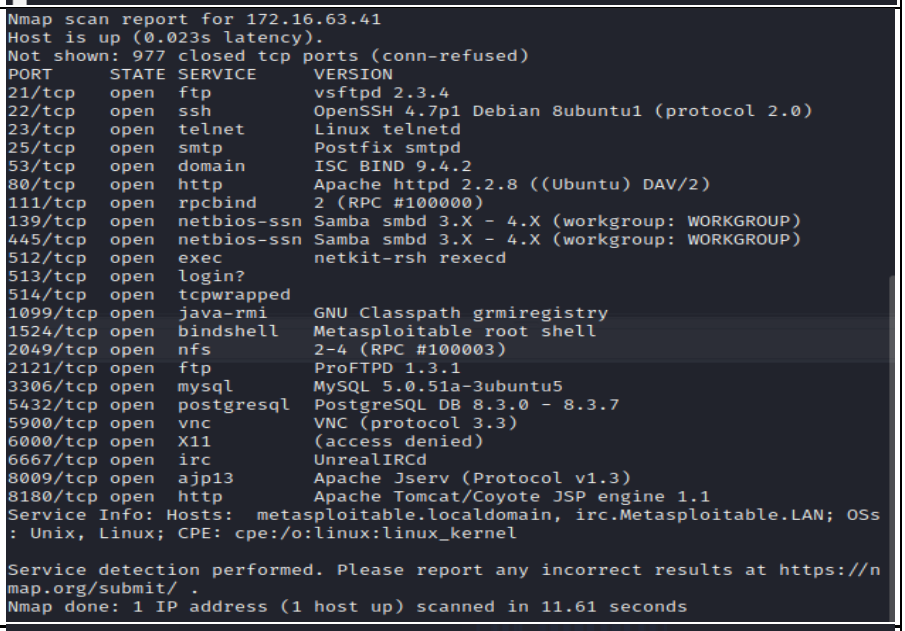
In this report I will be performing an audit on a remote machine and looking for vulnerabilities using NMAP to scan ports, find outdated software and searching if there are any exploits online on exploit databases. I will be identifying outdated software, analysing each service's purpose and how vulnerable it might leave the machine. I will then be using penetration software such as Armitage to exploit the outdated software and check its effectiveness. Finally using a topology provided I will set up static routing and an access control list to restrict and allow access on specified networks.

Scan of Remote Machine (NMAP)

I will begin by running an NMAP scan on the IP Address 192.16.63.41

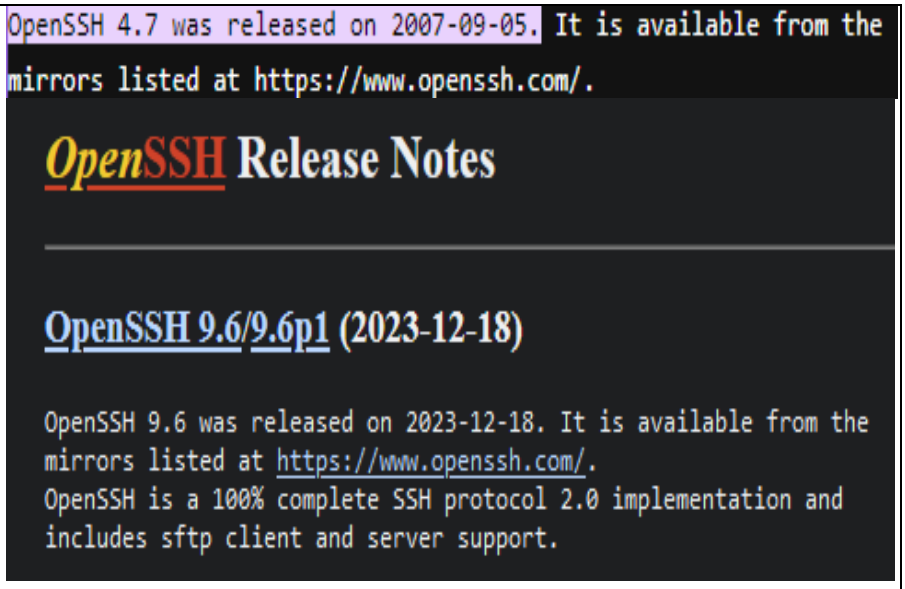
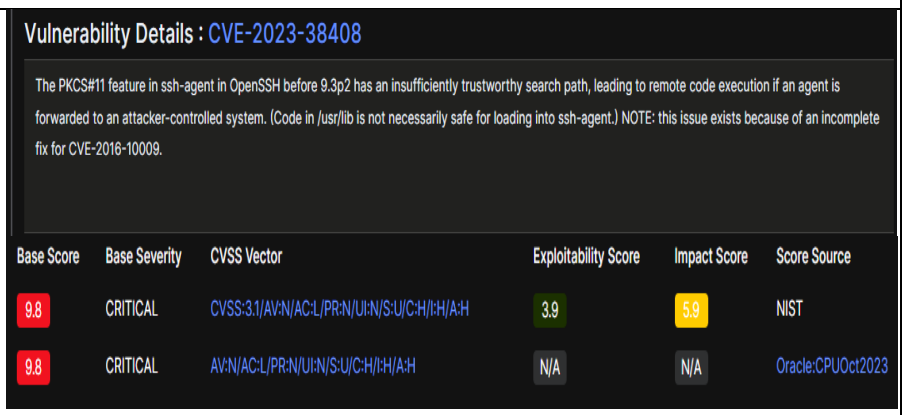
To do this run the following command:
Nmap -sV 172.16.63.41

The reason for running this scan is to check for open ports and look for outdated software so that I can check an online exploit database and seeing how hackers can take advantage of this.



On port 22 I can see there is an open ssh server.

Here I can see that the version of SSH is OpenSSH 4.7 on a Ubuntu machine, I checked the OpenSSH website to check the

current version and found that OpenSSH is currently on version 9.6, this tells us that this server is running an extremely outdated version of the software leaving their machine vulnerable to attacks. According to their website OpenSSH 4.7 was released in September 2007 meaning the version running on this machine is 16 years old. This is extremely unsafe and should be updated urgently as this provides attackers with an easy point of access. I recommend updating to the current version	
An open ssh server on port 22 can leave a machine vulnerable because it gives attackers a potential entry point allowing them to access the system. SSH stands for secure shell and is used to secure remote access and control over a network, if it is left open and isn't updated to the current version it can expose the system to brute force attacks and other exploits, If the unauthorized user gains entry then they have a free run at all data stored on the network. Upon researching this version of OpenSSH online I found a variety of sites such as exploitdb and cve-details where exploits have been shared in relation to this version of OpenSSH (4.7) As an example here[i] I found a post on cvedetails stating that versions of OpenSSH before 9.3 including this one have an untrustworthy search path which could allow hackers to remotely execute their own code on the remote machine	

This is ranked as a critical exploit meaning it could have severe repercussions it may be best to install the current version to prevent this from happening.	
The next thing that I noticed about this machine is that it had an Apache server running on port 80. Apache is an open source HTTP software used to accept HTTP requests allowing users to visit websites. This Apache server is running on a Ubuntu Machine and the app version is Apache 2.2.8, Upon researching this Apache is currently on version 2.4.58 meaning that the software is outdated and could potentially be leaving the remote machine vulnerable to attack. An outdated Apache http server on port 80 could leave the machine vulnerable as it may contain bugs and security risks that have been patched in newer and more recent versions. Hackers could exploit these oversights to steal unencrypted data or even use the machine as a way into the server hosting it and attack on a larger scale, they could also potentially run denial of service attacks to block legitimate requests. The fact that the website is running through HTTP and not HTTPS is also a massive security risk as the site data such as usernames and passwords are being stored as plain text making it readable to anybody who has the ability to intercept the communication and follow the stream, If the site was HTTPS this would not be an issue as all stored data is encrypted. I would recommend	

changing to HTTPS when handling data as it is more secure and poses less risk.

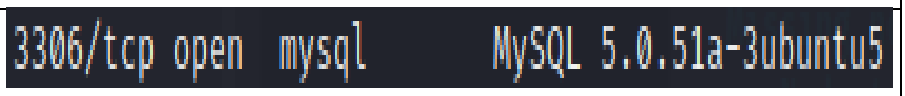
I would also recommend updating Apache to it's current version of 2.4.58 in order to patch any bugs and eliminate a variety of security risks.

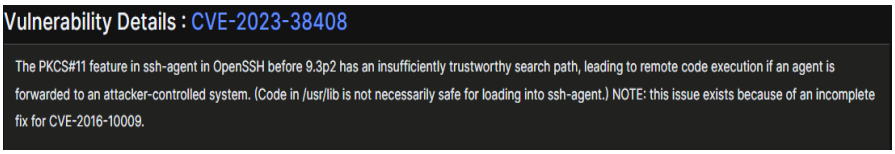
Upon researching any possible vulnerabilities relating to Apache 2.2.8 I found a post[ii] stating that any version of Apache before 2.4.53 including this one was at risk of denial of service as there was no default limit on the possible input size when a particular lua script was used essentially meaning the server could be pushed beyond a limit it could handle.

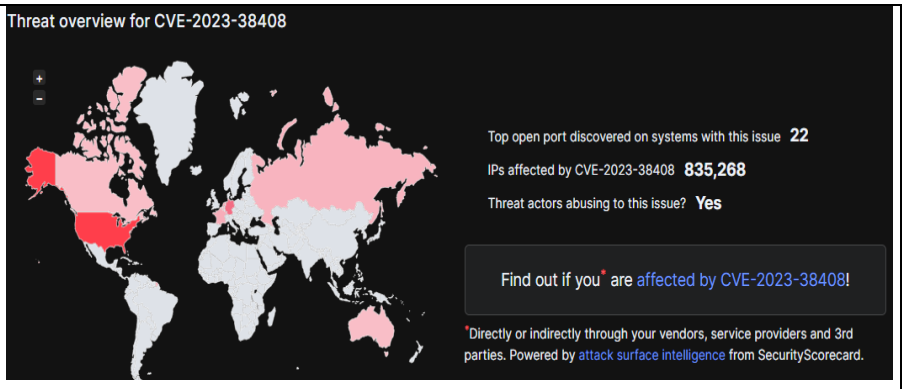
This means that an attacker with knowledge of this could send a large amount of data to the server in order to overwhelm it, push it past its limit and cause it to stop responding. All of this can be prevented by simply updating the software.

Apache 2.2.8 released in 2005 meaning this software is 18 years out of date and should be top of our list of priorities in terms of updating outdated software.

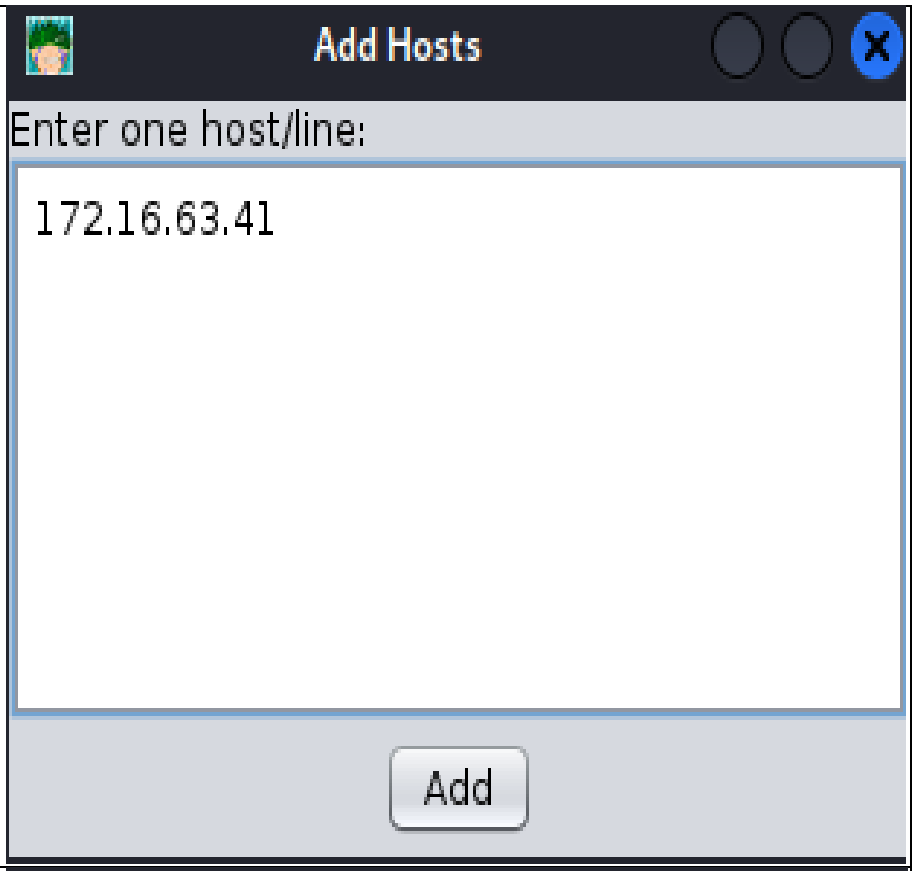
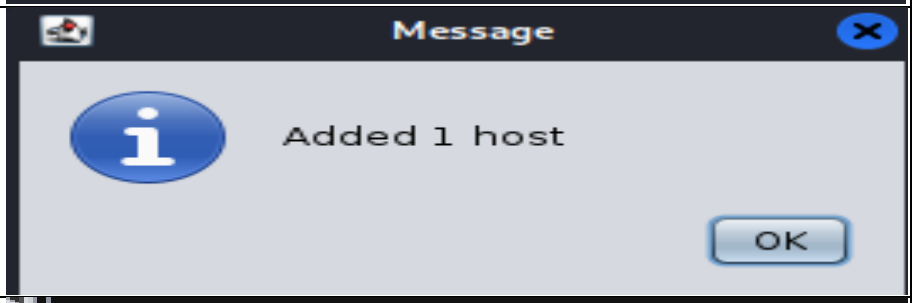
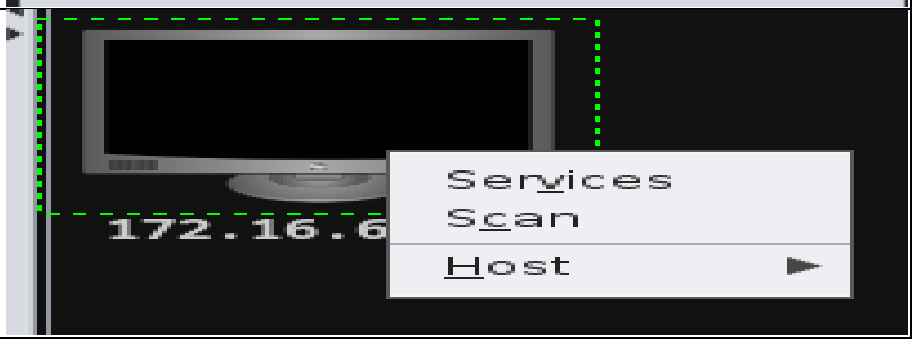
On port 3306 I noticed that there was an instance of MySQL running on our remote ubuntu machine connected to the server. SQL is a programming language used for managing databases. The version of MySQL running on this machine is 5.0.51a-3, the current version of MySQL is 8.3.0 meaning that

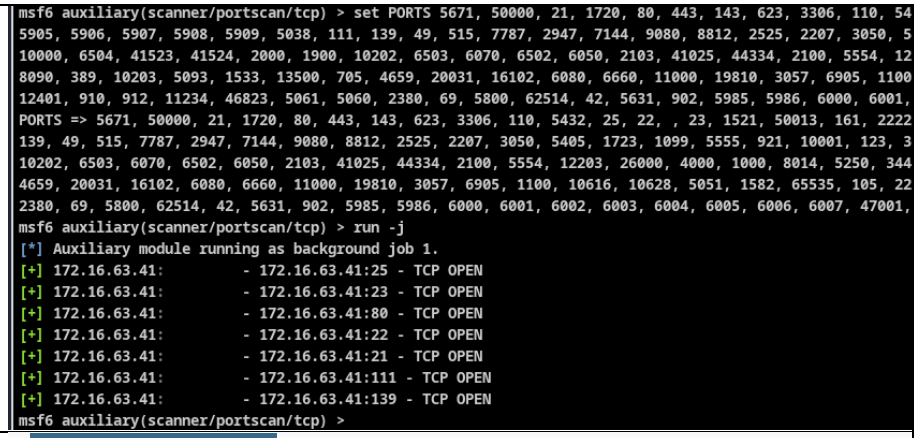
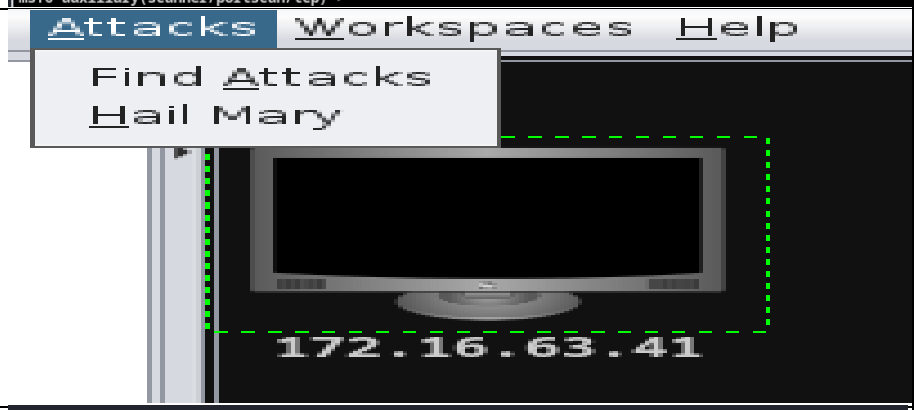
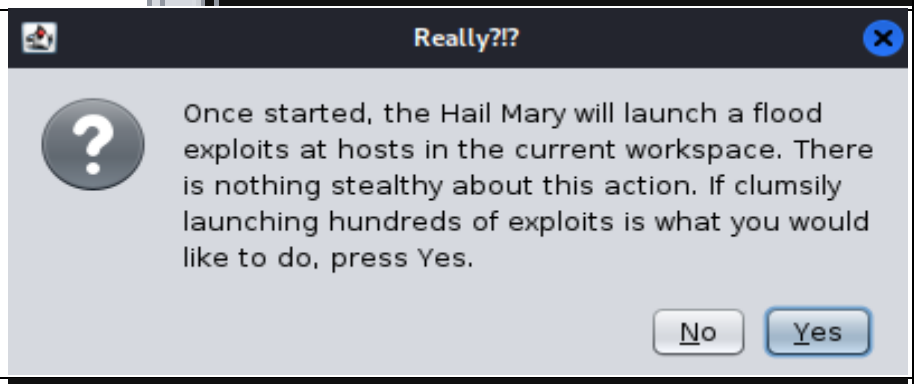
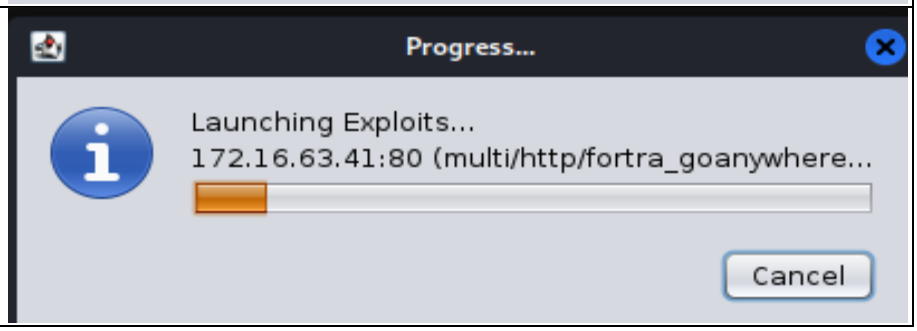




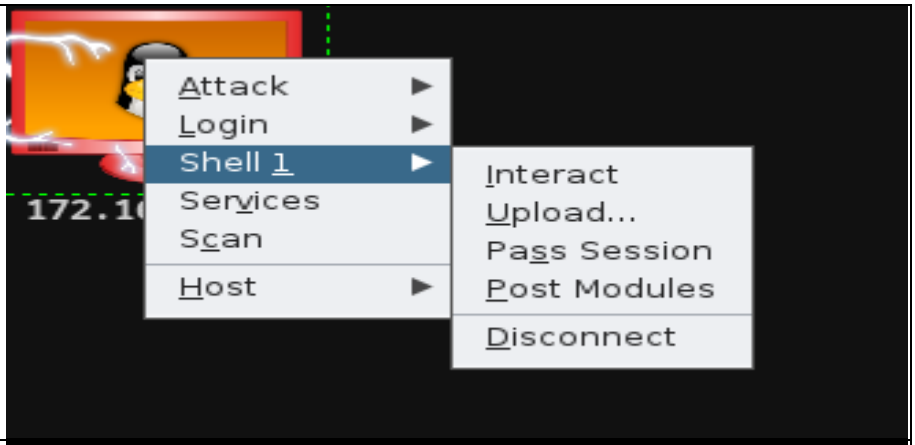
the version on our remote machine is extremely out of date. One major risk of an outdated version of MySQL is that it could allow attackers to execute malicious code that could allow unauthorized access and loss of data. After researching exploits online for this version of MySQL I found this post[iii] that stated any versions of MySQL before version 9.3p2 had untrustworthy search paths that could allow an attacker to remotely inject code and harm the system. I would suggest updating OpenSSH to the current version immediately in order to prevent this from happening. OpenSSH version 5.0 51a came out in January 2008 meaning it is 16 years out of date which is a massive concern in regard to our audit.	

Penetration Test (Armitage)

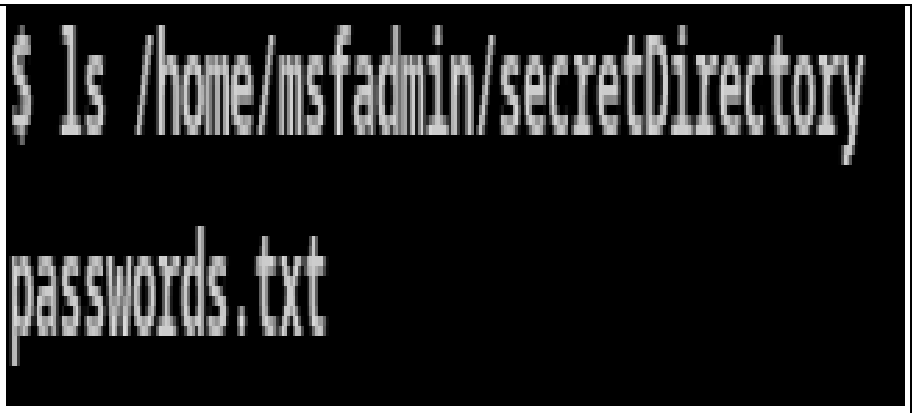
Now that I have clearly assessed our machine and identified outdated software I will be attempting to penetrate the machine using Armitage to test exploits and see any flaw in security. I will begin by booting up a Kali Linux machine with Armitage installed and running the software, once I have opened the software I will enter the host IP of our machine in the add hosts dialogue box	
Once I have been met with this message I can see that our attempt at adding the host has been successful.	
Next click Scan to run another quick scan on our host machine and check for open ports.	

Now that I have run our scan I will be ready to launch a variety of exploits at our host machine simultaneously, This method is known as a Hail Mary.	
Once ready, navigate to the attacks tab above the GUI of the machine and click “Hail Mary” from the drop down menu	
When met with this dialogue box click yes , this is giving us a brief explanation of what I am about to do.	
Now wait until all exploits have been launched	

Once this has finished, if successful The GUI image of the host PC should turn red indicating that our Hail Mary has been successful	[*] 172.16.63.41:445 (multi/samba/nttrans) [*] 172.16.63.41:139 (multi/samba/usermap_script) [*] 172.16.63.41:445 (multi/samba/usermap_script) [*] 172.16.63.41:23 (unix/misc/polycom_hdx_auth_bypass) [*] 172.16.63.41:23 (unix/misc/polycom_hdx_traceroute_exec) [*] 172.16.63.41:1099 (multi/misc/java_rmi_server) [*] 172.16.63.41:3306 (multi/mysql/mysql_udf_payload) [*] 172.16.63.41:5432 (multi/postgres/postgres_copy_from_program_cmd_exec) [*] 172.16.63.41:5432 (multi/postgres/postgres_createlang) [*] 172.16.63.41:6667 (multi/misc/legend_bot_exec) [*] 172.16.63.41:6667 (multi/misc/pbot_exec) [*] 172.16.63.41:6667 (multi/misc/ra1nx_pubcall_exec) [*] 172.16.63.41:6667 (multi/misc/w3tw0rk_exec) [*] 172.16.63.41:6667 (multi/misc/xdh_x_exec) [*] 172.16.63.41:6667 (unix/irc/unreal_ircd_3281_backdoor) [*] 172.16.63.41:5900 (multi/vnc/vnc_keyboard_exec) [*] 172.16.63.41:3632 (unix/misc/distcc_exec) [*] 172.16.63.41:6000 (unix/x11/x11_keyboard_exec) [*] Listing sessions... Listing sessions in 4 seconds
In example:	
Now that I have finished our Hail Mary successfully I will open a new shell to test how effective our exploits have been.	[*] Listing sessions... msf6 > sessions -v Active sessions ========== Session ID: 1 Name: Type: shell Info: Tunnel: 172.16.63.49:39941 -> 172.16.63.41:12440 (172.16.63.41) Via: exploit/multi/http/php_cgi_arg_injection Encrypted: No UUID: CheckIn: <none> Registered: No msf6 >

Right click on the Red Host machine image and select shell 1, from the drop down menu select interact. This should open a CLI shell	

In the shell I have opened I will first enter the home directory by typing:
 	
cd/home
 	
I will then check what is contained in the home directory by typing list function:
 	
ls
 	
when I type **ls /home** I can see a folder named msfadmin which would seem important and pique an attacker's interest I will access this folder to check it's contents by typing:
 	
ls /home/msfadmin
 	
I have found a directory named secretDirectory, finally I will list it's contents to see what it may contain:
 	
ls /home/msfadmin/secretDirectory
	`$ cd /home`
	`$ ls`
	`dav`
	`dvwa`
	`index.php`
	`mutillidae`
	`phpMyAdmin`
	`phpinfo.php`
	`test`
	`tikiwiki`
	`tikiwiki-old`
	`twiki`
	`$ ls /home`
	`ftp`
	`msfadmin`
	`service`
	`user`
	`$ ls /home/msfadmin`
	`hello`
	`secretDirectory`
	`vulnerable`
	`$ ls /home/msfadmin/secretDirectory`
	`passwords.txt`
	`superSecretDirectory`
	`$ cat /home/msfadmin/secretDirectory/passwords.txt`
	`Password01`
	`4BlindMice`

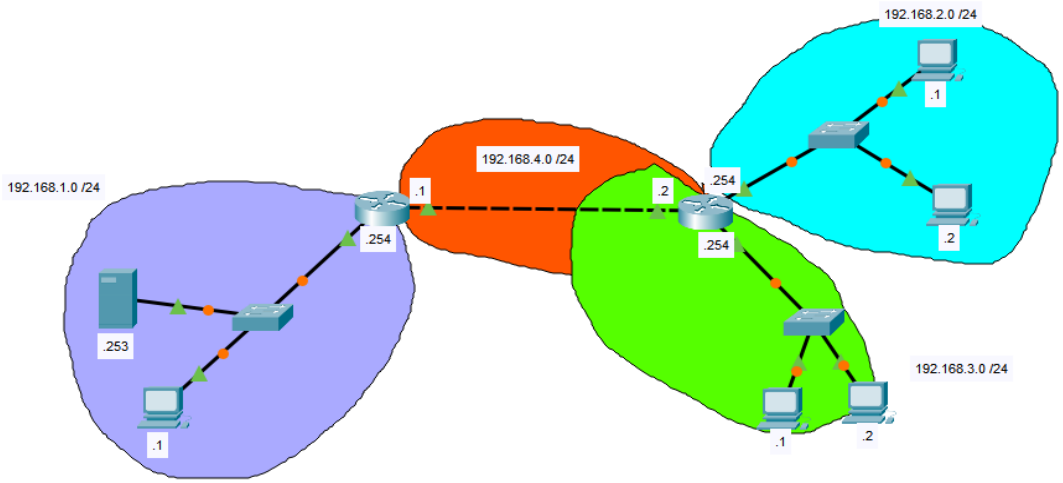
In this secret directory I have found a file named passwords.txt Our searching seems to have paid off and I will view the contents by typing: **Cat/home/msfadmin/secretDirectory/Passwords.txt**	
After I cat the passwords.txt file I was met with the following: **Password01** **4BlindMice**	

I have successfully performed a penetration test on our remote host machine using Armitage, I scanned for open ports, ran a Hail Mary and searched the directories for any hidden information, upon doing this I managed to find a secret directory with a text file containing passwords stored in plaintext.

The passwords are as follows:

Password01

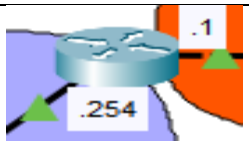
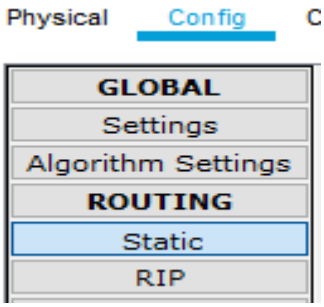
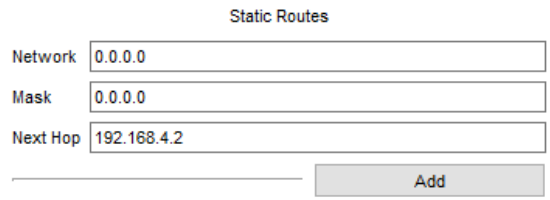
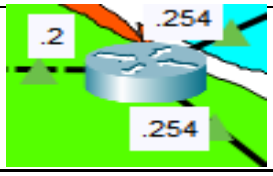
4BlindMice

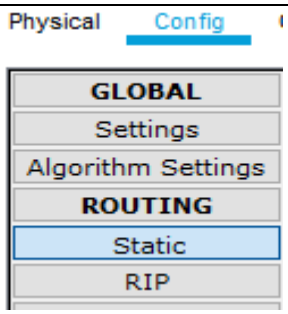
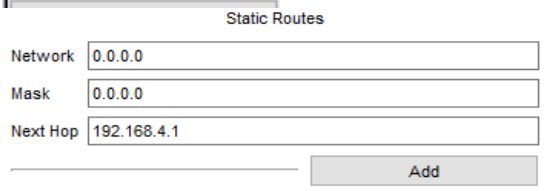


Access Control Lists

In this segment using this topology that I have set up I will be configuring static routing betIen both Routers (4.1 and 4.2), I will then be creating an access list in order to deny access to the Web server and ICMP from the 192.168.3.0 network and allowing access from everywhere else.

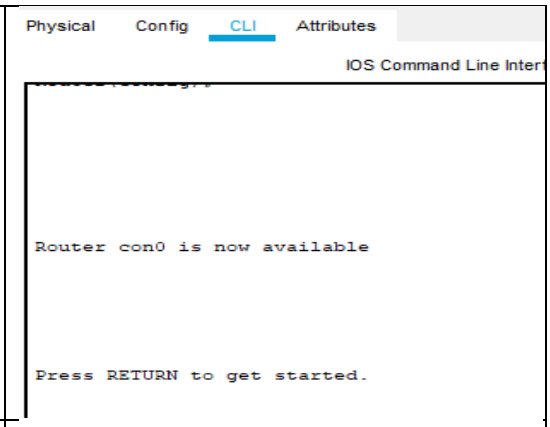
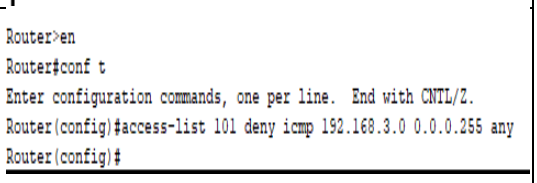
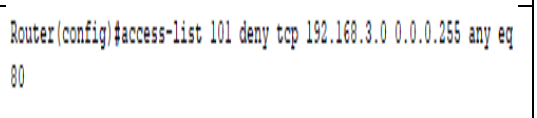
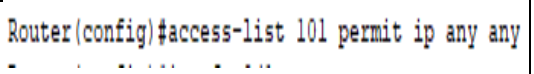
Setting Up Static Routing

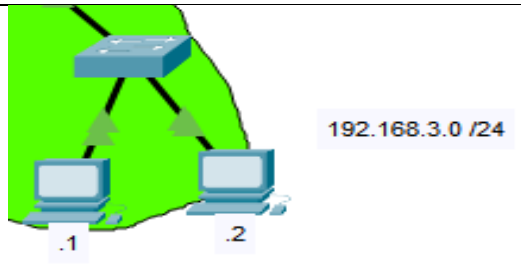
I will begin by entering the first Router addressed 192.168.4.1 and navigating to the "Config" tab.	
Once I am in the config tab I will select "Static" from the drop-down menu, this will bring us to a menu where I can set up and configure static routing to connect both routers in our topology.	
I will now fill out the dialogue boxes as follows. Network: 0.0.0.0 Mask: 0.0.0.0 Next Hop: 192.168.4.2 This will connect router 1 to router 2	
I will now repeat the same steps for the second Router addressed 192.168.4.2, starting off by navigating to the "Config" tab.	

Once I have located the config tab I will once again select the static option so that I can begin setting up static routing on our second Router.	
I will once again fill out the dialogue boxes as follows: Network: 0.0.0.0 Mask: 0.0.0.0 Next Hop: 192.168.4.1 This will connect Router 2 back to Router 1 and ensure that static routing is set up both ways	

Setting up and Implementing Access lists

In this segment I will be setting up an access list on our router 1 interface in order to prevent our 3.0 network from sending ICMP and accessing the Web server (192.168.1.253) whilst allowing Web and ICMP access to the 1.0, 2.0 and 4.0 Networks.

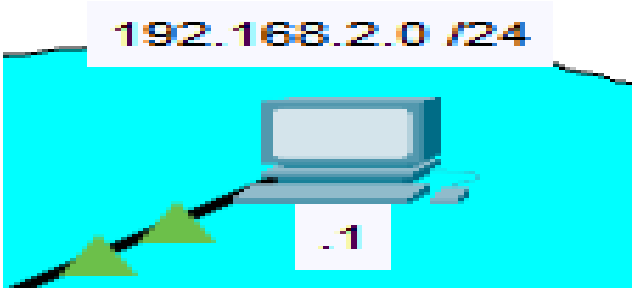
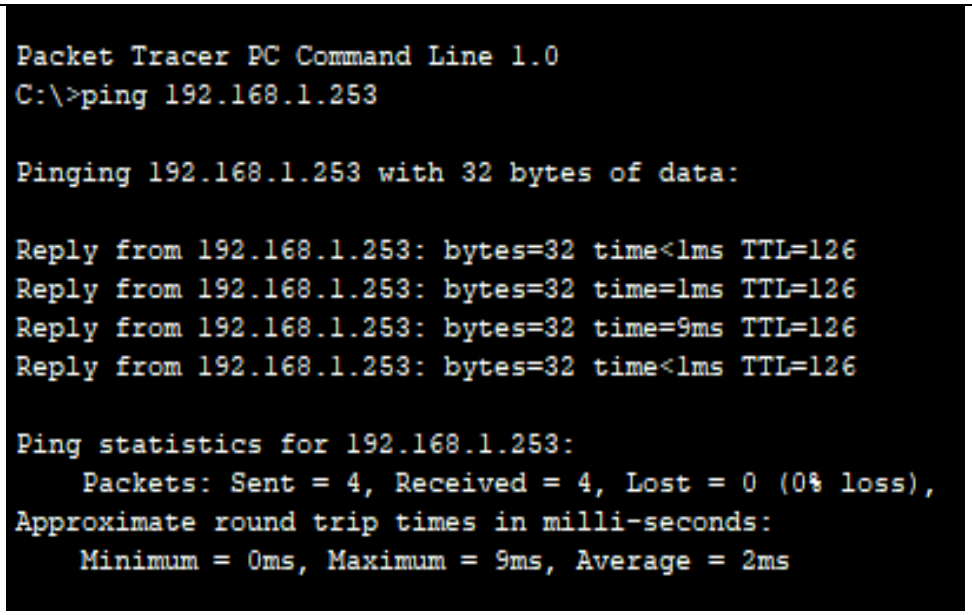
Using the CLI (Command Line Interface) I will set up an access list on Router 1 that will aim to block ICMP and Web access from the 3.0 Network.	
Once I have opened the CLI I will type the following in order to set rules for our access list and block ICMP from the 3.0 Network: **en** **conf t** **access-list 101 deny icmp 192.168.3.0 0.0.0.255 any**	
Next, I will be blocking TCP from the 3.0 network by typing the following: **access-list 101 deny tcp 192.168.3.0 0.0.0.255 any eq 80**	
Our final rule in this access list is to allow IP from any source in our network whilst simultaneously blocking ICMP and TCP from 3.0. If successful, the rest of our network should still have access to ICMP and the Web server whilst 3.0 is excluded. To allow ip access: **access-list 101 permit ip any any**	

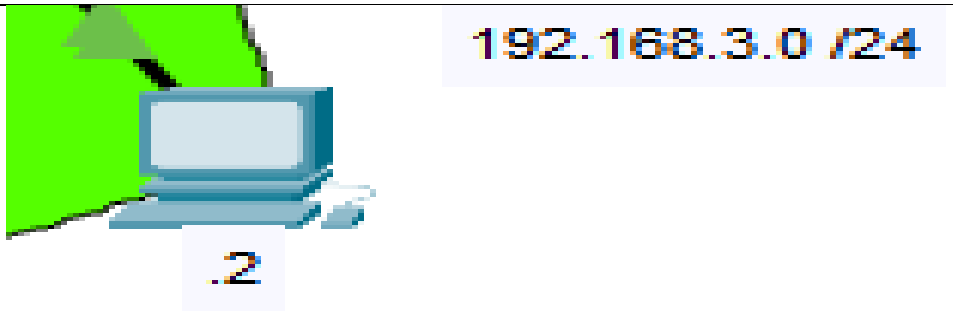
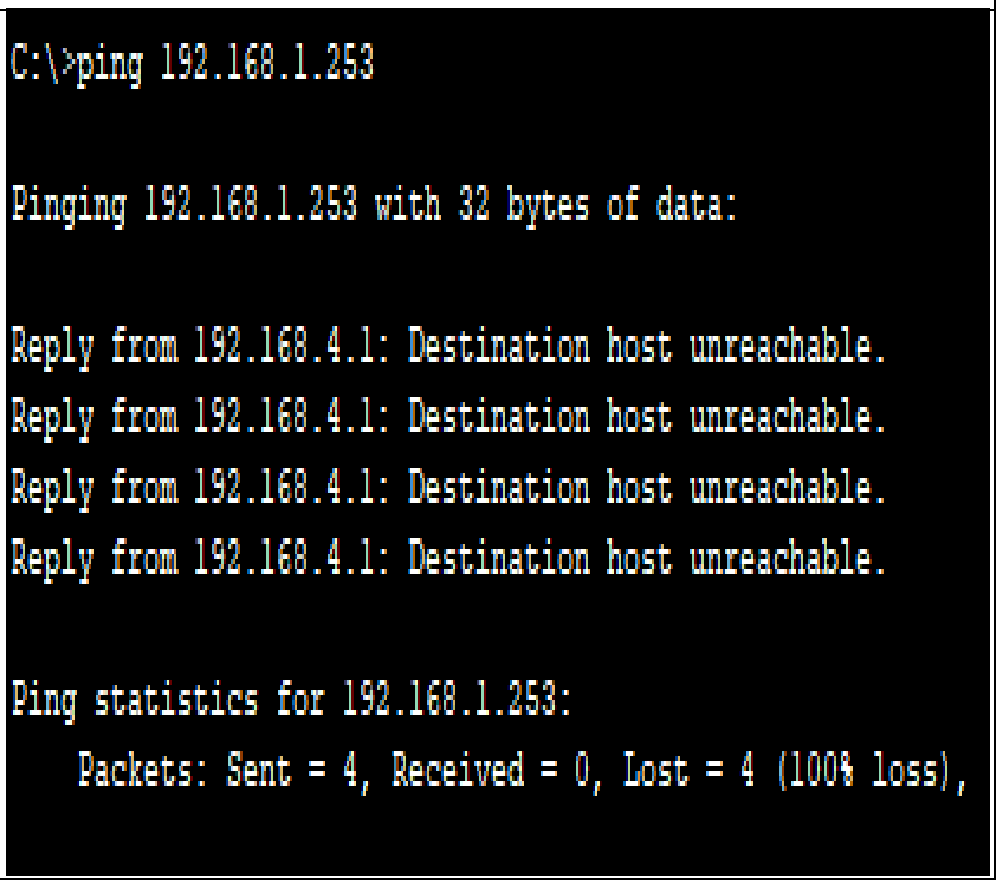
access-list 101 permit ip any any	
Now that I have established our rules our last step is to implement our access list into our network to do this I will type: **Int fa 0/1** (I are entering the interface on our router where our access list will be set up) Once in our interface I will type: **ip access-group 101 in** **exit** **exit**	Router(config-if)#ip access-group 101 in Router(config-if)#exit Router(config)#exit
Our access list has been set up successfully and it is now time to test if it has been implemented correctly. If successful, our 192.168.3.0 network will be unable to access ICMP pings and our Web server whilst the rest of our server still has access due to the access list specifications/guidelines	

Testing our Access List

Now that I have set up our access list in the topology, I will test its effectiveness.

I will begin by checking if I can ping using ICMP on our 2.0 network that should have permissions I will also check if I can access the server's Web address.

I will test the 2.0 network using PC 2.1 in order to ping our 1.253 server on the other side of our topology	
Once in the PC 2.1 I will enter the terminal and ping our server's address by typing: **Ping 192.168.1.253** I have obtained a reply telling us our ICMP ping was successful	
Next, I will test our Web server access by using the PC 2.1 GUI to type the address of our server where a page should be displayed.	

Once in the Web browser I will type the server address: **192.168.1.253** A page has been displayed telling us that Web access has been permitted successfully	
Next, I will check our 3.0 network that should not have access to the Web server and ICMP pings using PC 3.2.	
Once in the PC 3.2 I will enter the terminal and ping our server's address by typing: **Ping 192.168.1.253** I have obtained a reply from our 4.1 router that has our access list telling us that the destination host is unreachable	

meaning our access has been denied due to our access list.	
Finally, I will check our Web server access by using the PC 3.2 GUI to type the address of our server where a page should be displayed.	
Once in the Web browser I will type the server address: **192.168.1.253** I should be met with a dialogue telling us "Request timeout" due to the fact that our 3.0 network does not have access to our Web server like the rest of our network does.	

Finally, if I enter the CLI on Router 1 and type sh run I can show the run of the router through our CLI. Here if I navigate to our 0/1 interface, I can see our access list is displayed and each of its specified commands alongside it.	interface FastEthernet0/1 ip address 192.168.4.1 255.255.255.0 ip access-group 101 in duplex auto speed auto ! interface Vlan1 no ip address shutdown ! ip classless ip route 0.0.0.0 0.0.0.0 192.168.4.2 ! ip flow-export version 9 ! ! access-list 101 deny icmp 192.168.3.0 0.0.0.255 any access-list 101 deny tcp 192.168.3.0 0.0.0.255 any eq www access-list 101 permit ip any any ! ! ! --More--

This tells us that our access list has been set up and implemented successfully and that I have isolated our 3.0 network from the rest of our topology and limited its access to ICMP and the Web server.

Conclusion

In conclusion in this skills demo I successfully performed an audit on a remote host machine, I used NMAP to scan for any open ports and used online exploit databases to identify outdated software along with security concerns that may arise from them. I explained how they could pose a threat to the security of the machine and the server as a whole along with possible solutions in each scenario, I then used Armitage to run a penetration test on the remote host machine and exploited the outdated software in order to retrieve the contents of the passwords.txt file, I learned how important it is to keep software up to date in order to help prevent security risks. I then used the packet tracer file that I was provided to set up static routing between 2 routers along with an access control list that limits access to the web server and ICMP from the 192.168.3.0 network in the topology while allowing access from the other networks (1.0,2.0,4.0) .