Zack Heaton
PenTesting Unit 2
Denver University - Cyber Security Bootcamp

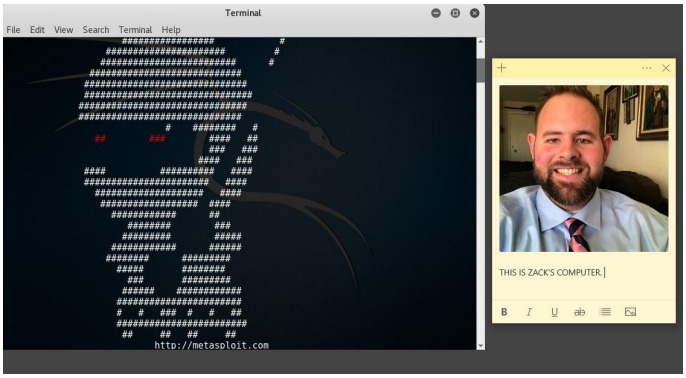
12-20 May 2020

It appears that the objective of this project is to learn and understand the functionality of a demilitarized zone (DMZ) and utilize that in order to further exploit a network. In this assignment I will begin by explaining what a DMZ is in terms of network security and will then move into how I was able to utilize the DMZ structure on a vulnerable machine to gain access to the network, cover my tracks, and create and functioning backdoor.

In order to understand a demilitarized zone it is crucial to understand the basic methodology on how a particular machine connects to the internet. This takes place when the network interface card (NIC) connects to the local area network (LAN). The LAN then connects to the internet and a connection is forged. Local Area Networks can have a firewall placed on them in order to sort through traffic coming in or going out but that is only one layer of protection. A demilitarized zone is an excellent idea to segregate a section of your network for personal use while important hardware is not located on the same part of the network infrastructure. An example of this might be a household where the security cameras are on the network because they require internet connectivity yet the system administrator does not want them on the same part of the network that friends and family use to browse Google during soccer matches.

This security architecture is intelligent but the appropriate measures must be taken to ensure the network's security. It is possible to use exploits in metasploit to essentially convert a DMZ into being a router without the knowledge of the system admin. Once the DMZ serves as an effective router it can then be used to tunnel into the subnet and even further to the LAN. The steps below will outline this process.

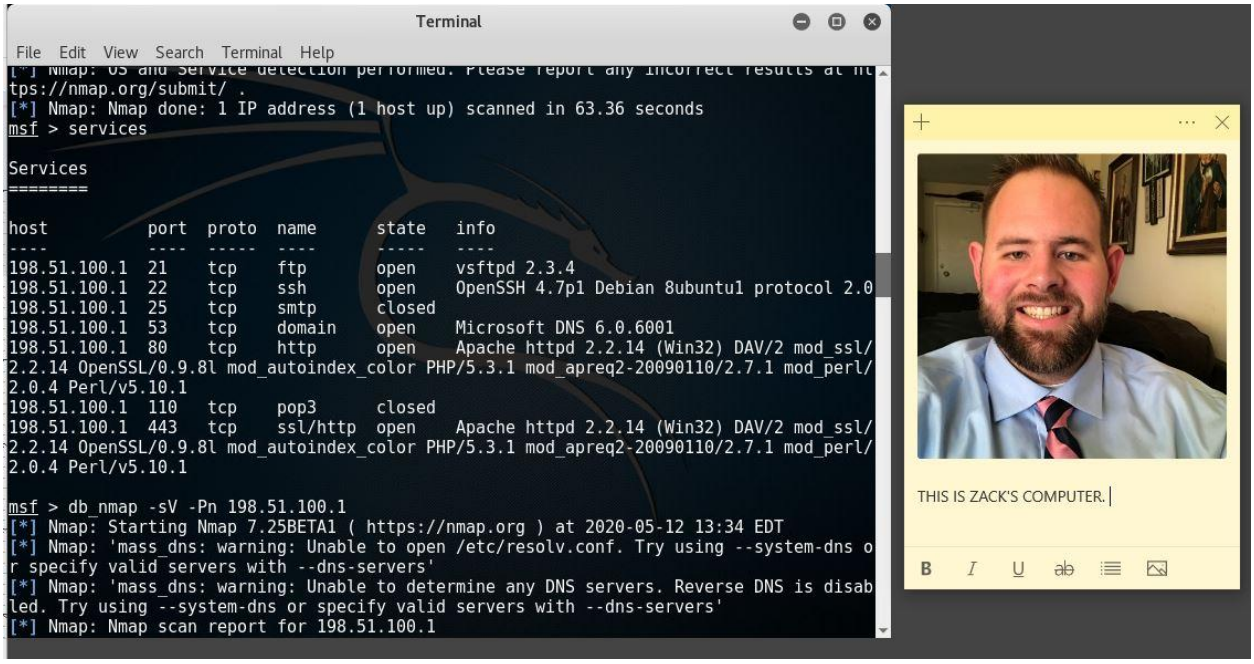
The first step in this entire process is to start up Metasploit and prepare the tools that we will need. In order to cut down on unnecessary screenshots I will not post several screenshots outlining the commands to get Metasploit operational. The below screenshot will serve as evidence that you have to get it running.



Screenshot 20-1

The next step to the process is to identify any target machines on the network 198.51.100.0/24. This revealed that there is a connected device to the address 198.51.100.1. In a real scenario we would have had to do some background research to determine the network that we wanted to target but in this case we knew that our target would be on this network. It should be stated at this point that there is not enough information to say exactly what 198.51.100.1 is or is not. That will come with further enumeration. At this point it is simply known that it exists and is connected to the LAN. Again,a screenshot has been redacted because it only shows that a scan took place.

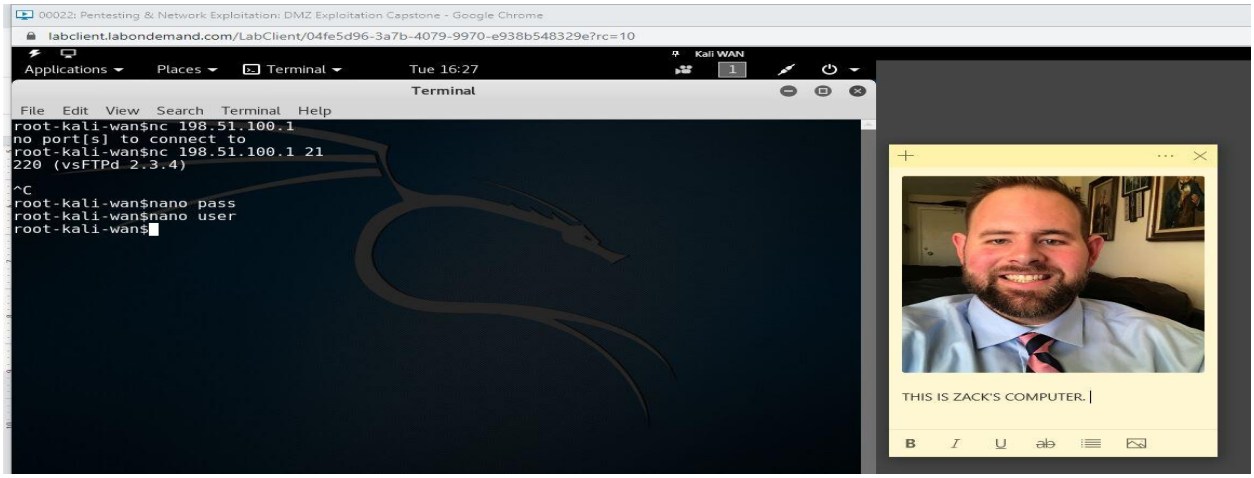
The next part of the task is to determine what services are going in and out of the device at 198.51.100.1. In order to do this the command [services] is issued and the result highlights some important information in Screenshot 20-2.



Screenshot 20-2

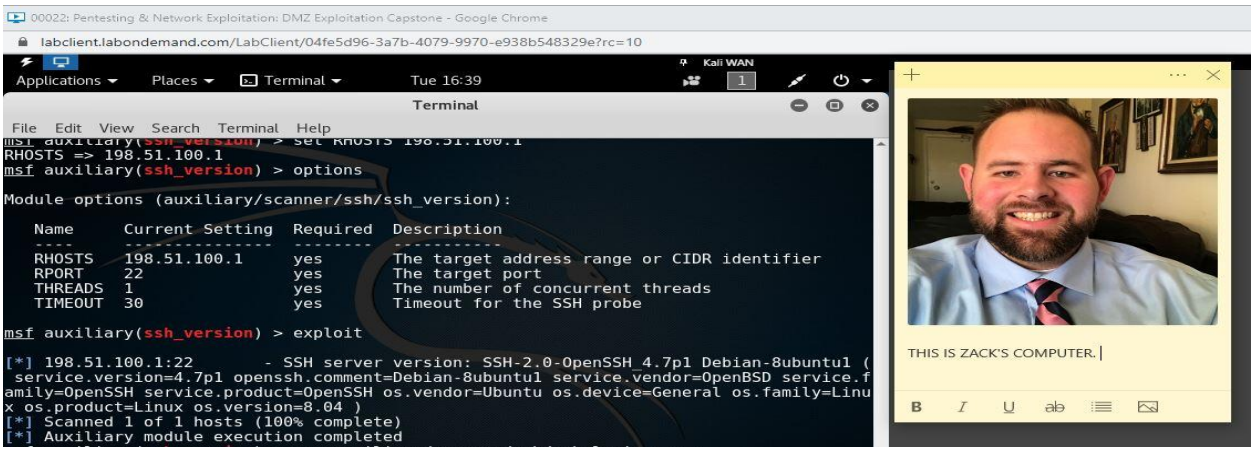
In screenshot 20-2 there is a fair amount of useful information. Beyond telling us which ports are open there is an interesting observation to be made. On port 22 we can see an Ubuntu machine, on port 53 there is something with what is likely a Microsoft server, and on port 443 there is an Apache server. The fact that there are two different operating systems and a web-based server tells us that there are most likely multiple devices connected to that one ip address. This could be a subnet or a DMZ. Considering that this is already a subnet within the wider structure this is most likely a DMZ.

Before any sort of exploit can actually be planned we have to make sure that Metasploit is providing correct information. We can do this by using Netcat to establish a connection in a different terminal. **THIS WILL LEAVE EVIDENCE.** The evidence it will leave will be left behind in logs which we can go back and modify if we so chose. Below in screenshot 20-3 you can see that using Netcat on port 22 where ssh is passing established a connection. This provides us the greenlight to continue exploiting.



Screenshot 20-3

The next part of our infiltration will be to choose how to actually exploit this DMZ. I see ssh is working and this is a protocol that we have learned to exploit using Metasploit. In order to find the best exploit the command should be issued within metasploit [search shh]. This command provides all the available ssh exploits. In a normal situation the process would be to do deep research on each of these exploits and determine the best one for this particular job. In this scenario we were provided the exploit to use which was [auxiliary/scanner/ssh/ssh_version]

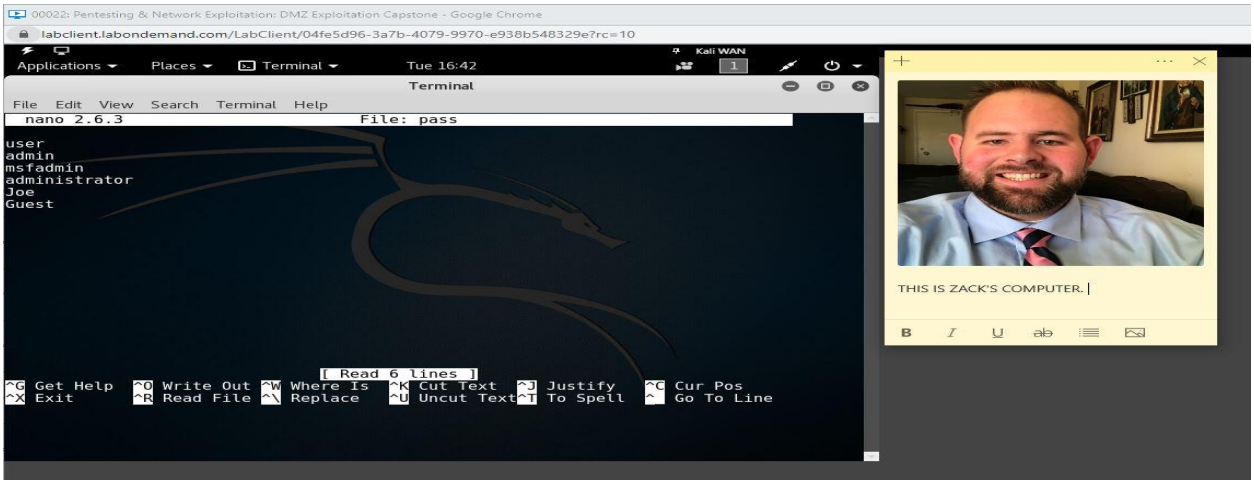


Screenshot 20-4

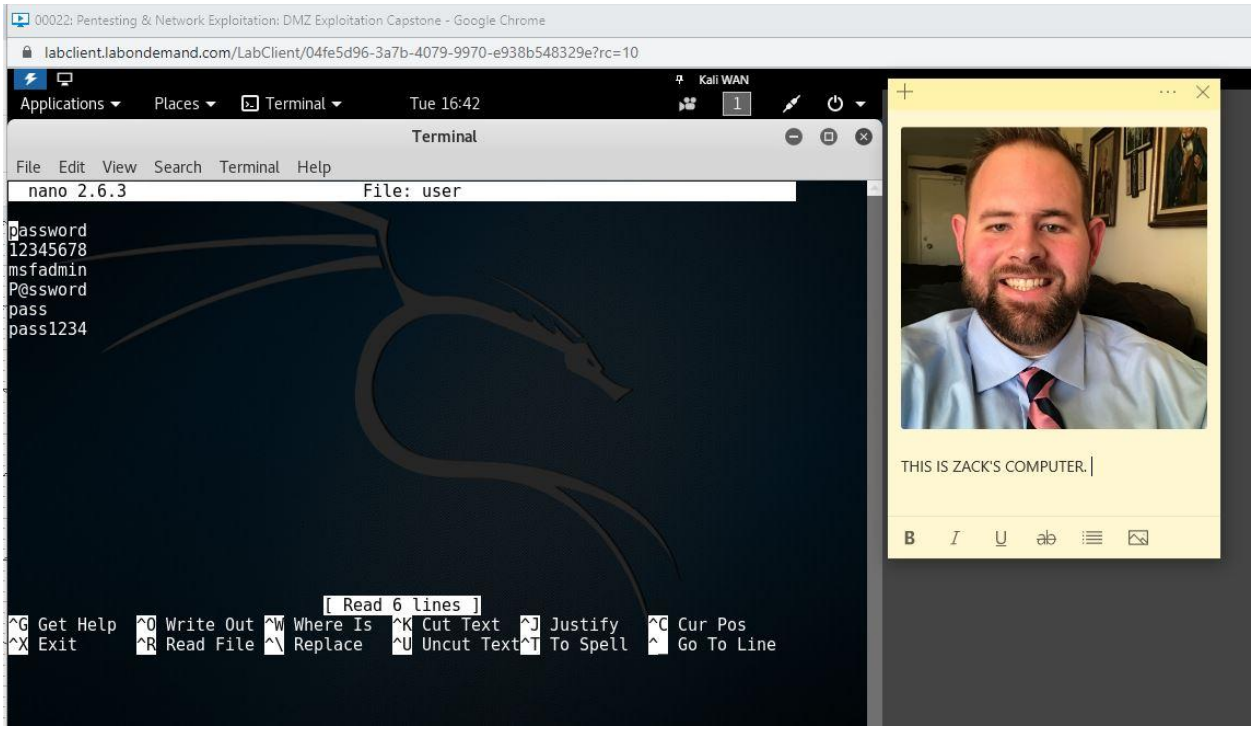
The exploit in screenshot 20-4 demonstrates that the machine is reachable not only by Netcat but also via Metasploit. Some might consider this step redundant but it seemed important to include here for the holistic explanation. The next exploit will attempt to get us logged into the server.

This next exploit is called [auxiliary/scanner/ssh/shh_login]. This exploit uses a John the Ripper like exploit to attempt a brute force entry to the server. In order to do this the RHOSTS must be set at the target IP address but there is also the setting of pass_file and user_file that must be set in order to exploit successfully. These files can be created using nano or any other text editor and in each of them you can simply place a list of common usernames and common passwords. These files should be created in a different terminal so you don't have to close Metasploit.

You can see in screenshots 20-5 and 20-6 that those files were created. A very important note to make in this particular write-up is that due to my own carelessness I accidently named the files in reverse. **That is to say that the user_file is actually the passwords and the pass_file is actually the users.** Normally the settings would be set according to the same name but in this case the screenshots will indicate the opposite.

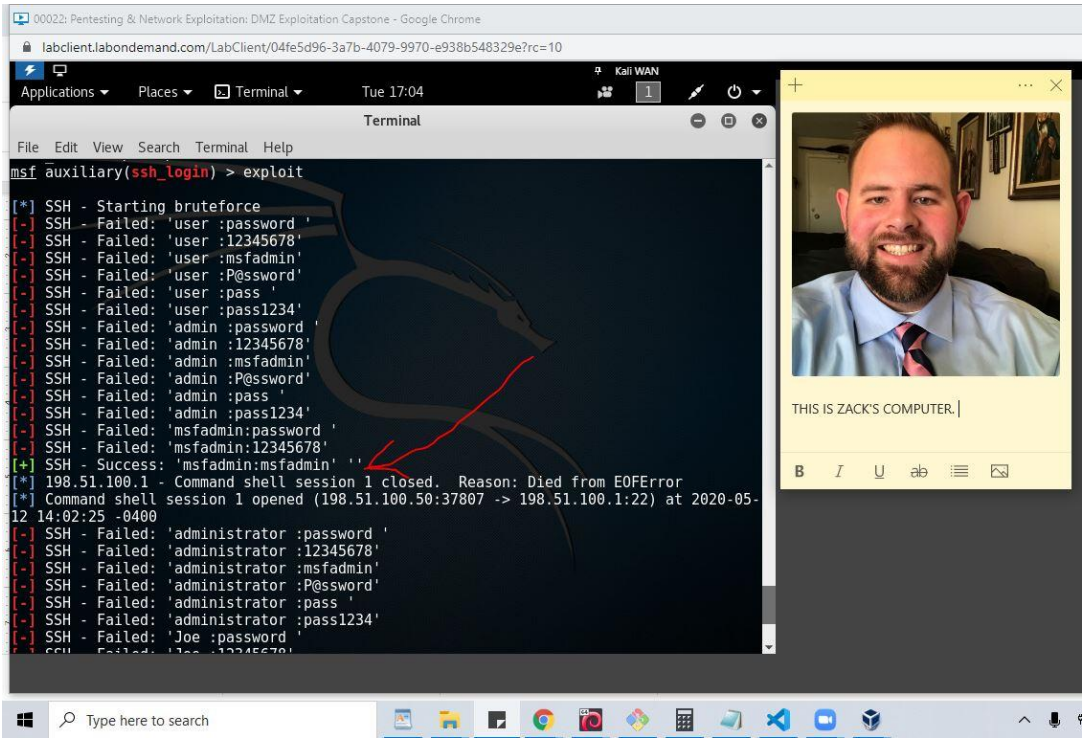


Screenshot 20-5



Screenshot 20-6

Now that these files have been created they can be set using their file paths in metasploit. When they are set the ssh_login can run and the results are shown below in screenshot 20-7.

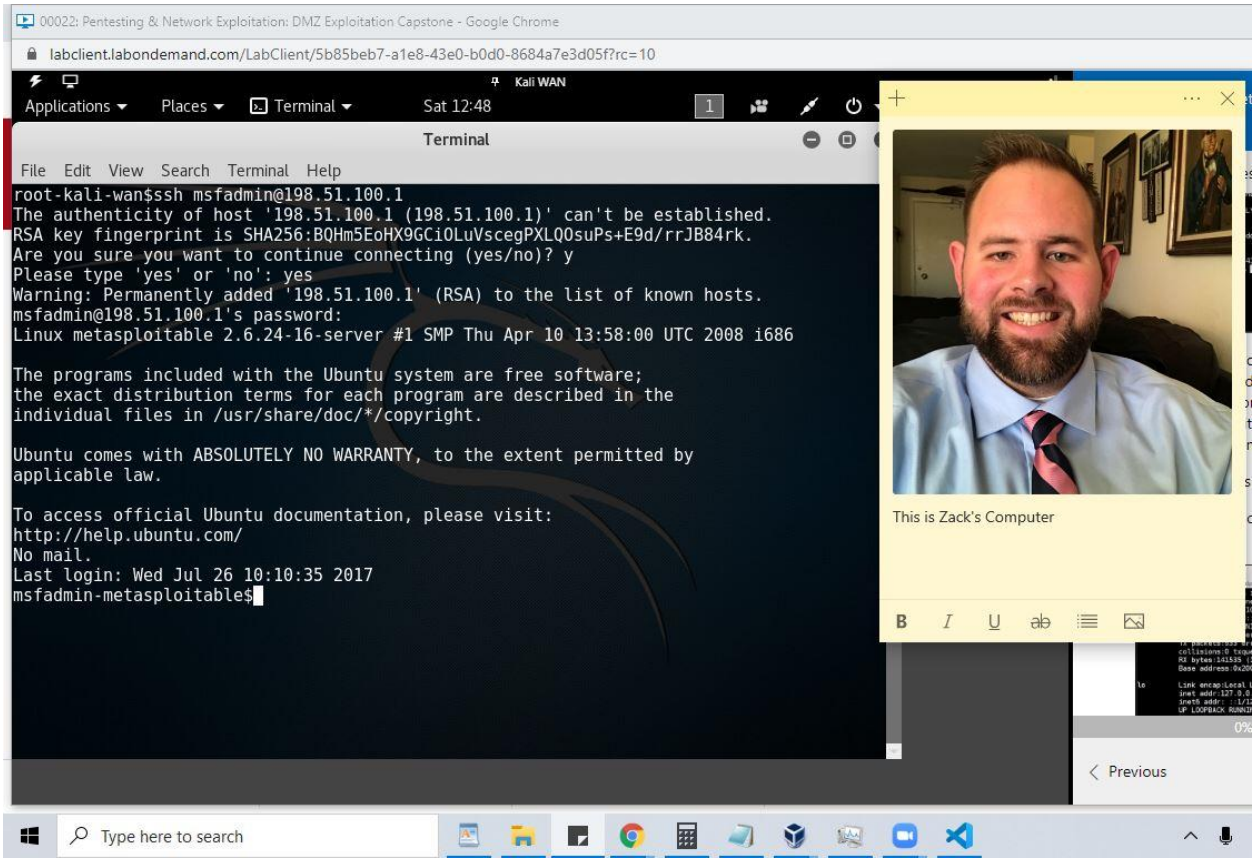


Screenshot 20-7

You can see that the exploit was successful and provided legitimate login information for the server. The username “msfadmin” and “msfadmin” will work to log on to that machine which we can then use to continue our exploitation and conversion of the DMZ. Another note that may be strange in this particular report. Given that the user_file and pass_file were named in reverse it is unknown which msfadmin is the password

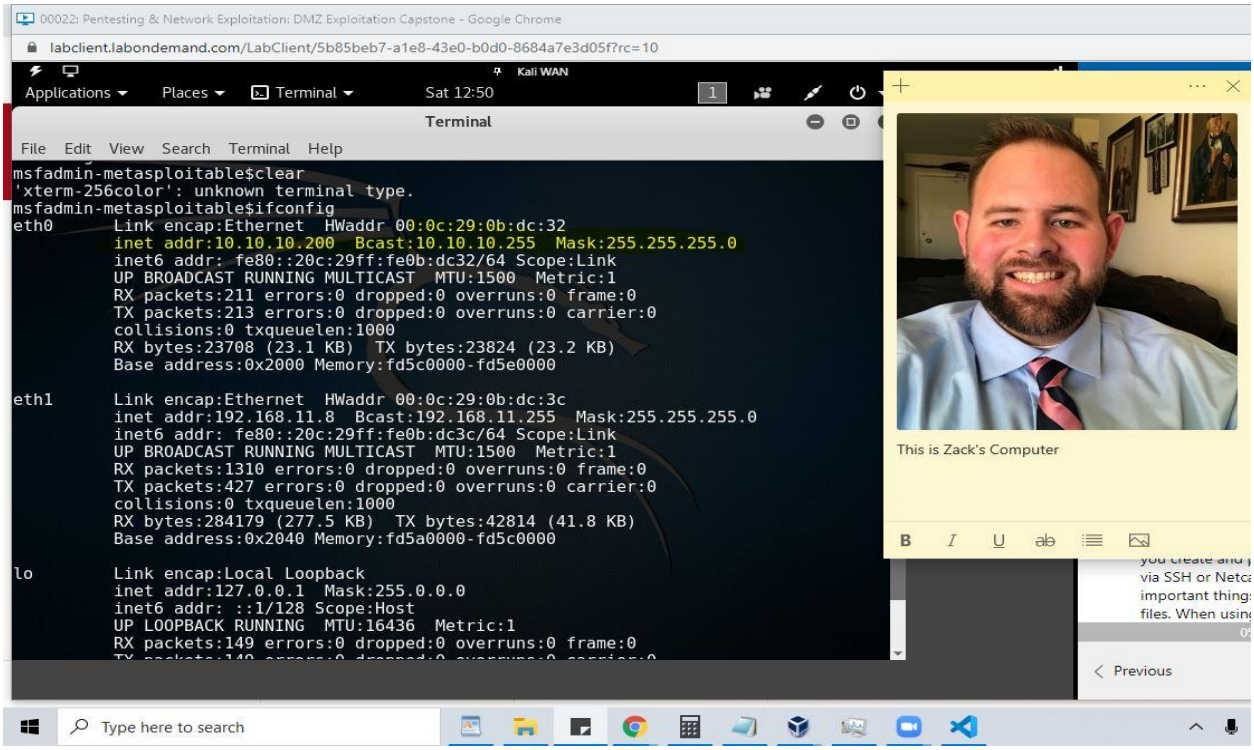
and which is the username. Going back and redoing this would provide that answer but the shorter message is don't use amateur credentials.

The line above is a great location marker if you are looking to read part and come back later.
Now that we have the credentials we are able to use ssh in order to get onto the DMZ and start collecting some information. In order to accomplish this you feed the command [ssh msfadm@198.51.100.1] to which it will ask for a password and we feed in the correct [msfadmin]. Screenshot 20-8 shows this below.



Screenshot 20-8

The next step is get our bearings in terms of network connections. We feed the command [ifconfig] which confirms we are actively on the DMZ! Screenshot 20-9 below shows in yellow that we are on the DMZ.



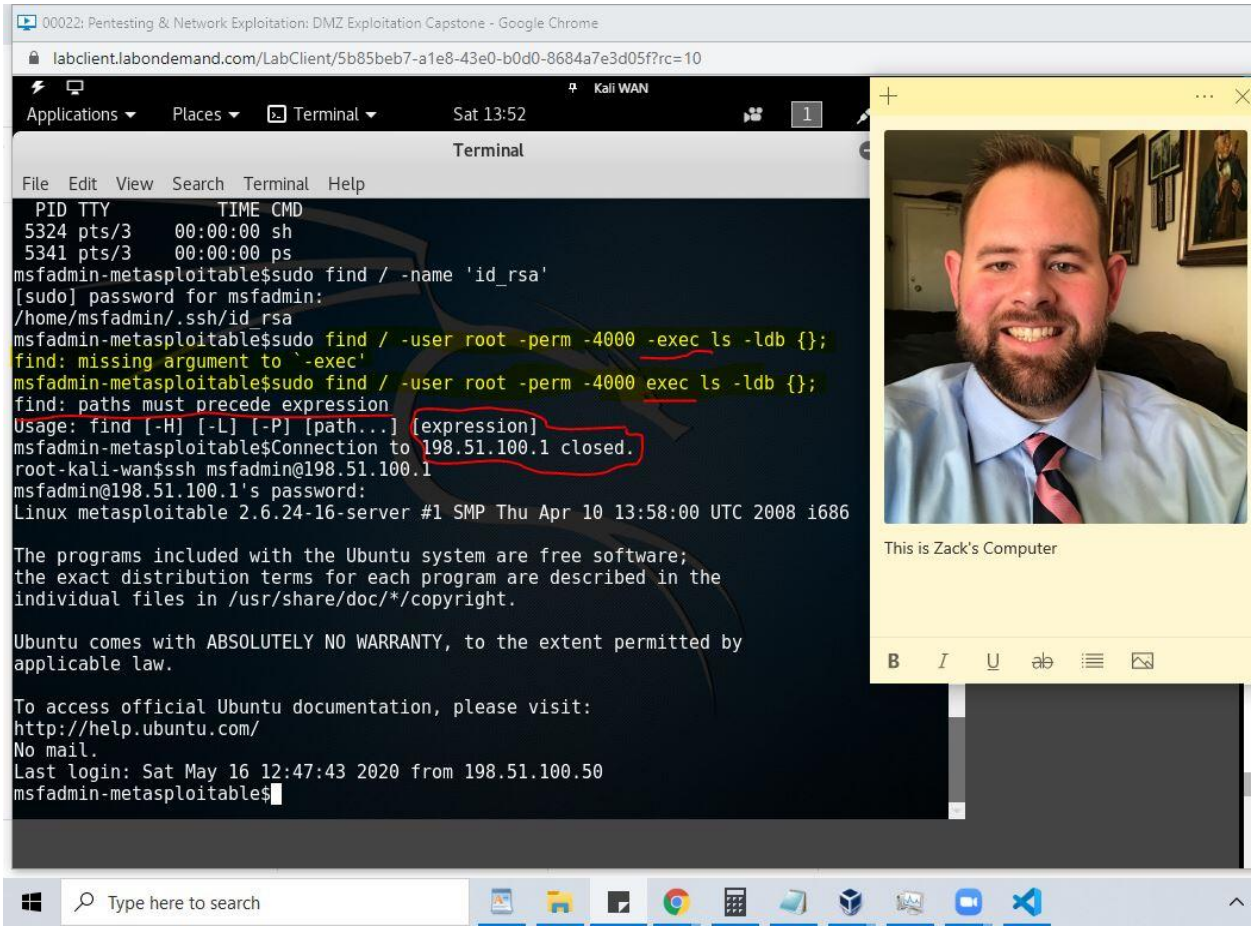
Screenshot 20-9

Now that we are for sure on the DMZ it is time to start mining for critical information. The instructions provided us with a list of commands to run and research what information they were yielding. All of this information was stored in a file called [“meta_victim.txt”]. Clicking on that link will provide all the information that was pulled out of the exploited machine. Below in Table A you will see a list of the commands and a description of what information they yielded in that text file

TABLE A	
uname -a	This command gives us the name of the machine.
dmesg | grep Linux	This gives us information on the kernel.
Ps -ef	This provides a list of all running services -ef basically makes it a full listing.
sudo find / -name ‘id_dsa’	Finding the private key within the ssh connection.
sudo find / -name ‘id_rsa’	Finding the private key within the ssh connection.
sudo find / -user root -perm -4000 -exec ls -ldb {} ;	This command I believe was trying to figure out which files run as root regardless of logged in as user.
msfadmin-metasploitable $ **cat /etc/passwd	This provides information on the users.
sudo cat /etc/shadow - this is finding passwords	This provides passwords for the users.
sudo netstat -tulpn	Netstat tells which ports are active and -tulpn makes it tell us which protocols and actual devices are listening.

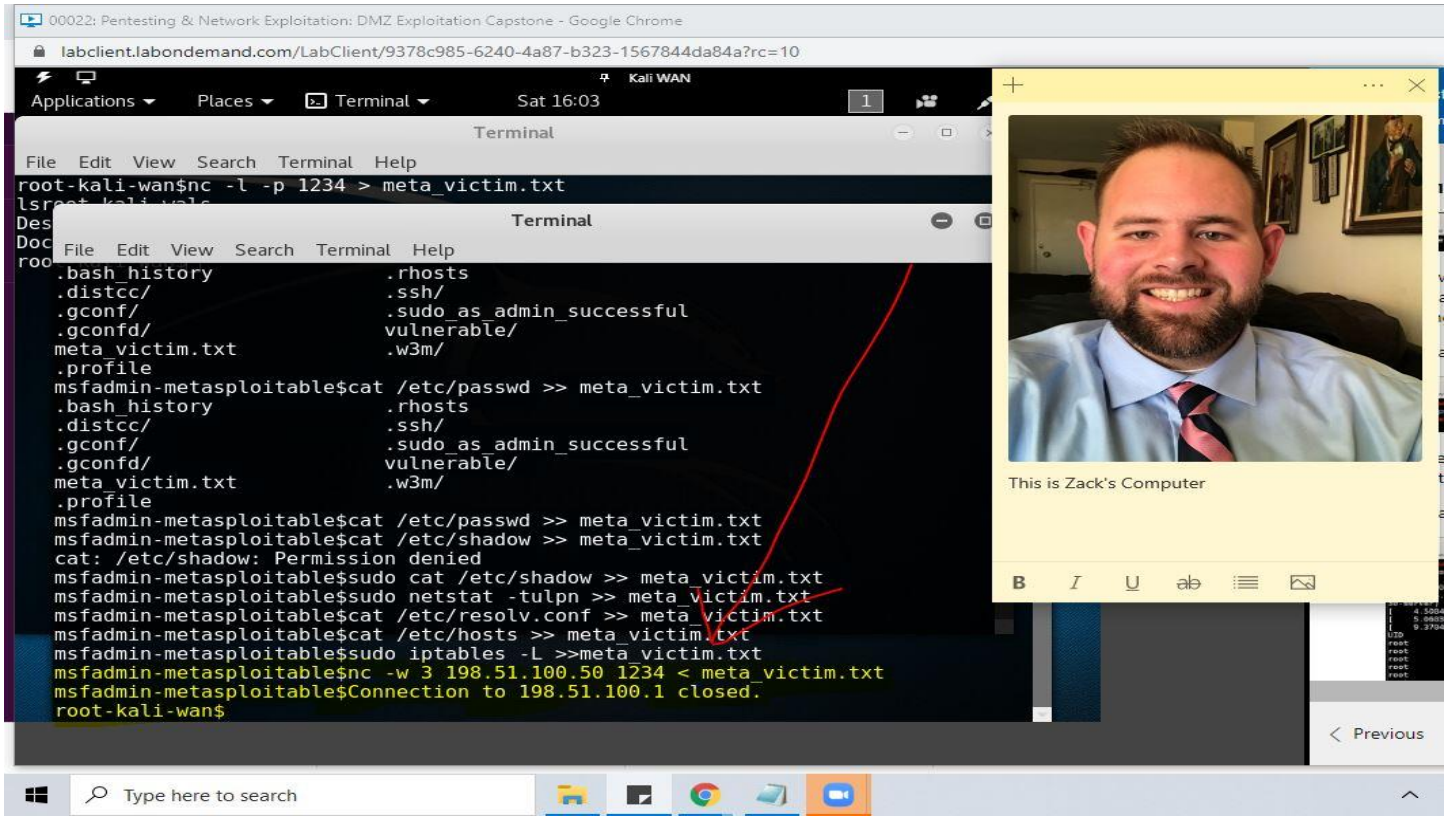
cat /etc/resolv.conf	This file contains the configuration information for the DNS server.
cat /etc/hosts	This file maps out hostnames to ip addresses.
sudo iptables -L	This lists out information about the firewall.

In the above table there was only one command that had trouble running. The command **[sudo find / -user root -perm -4000 -exec ls -ldb {} ;].** This command appears to attempt to investigate information based on the root user located within the light weight database. When this command was attempted it had issues with the [-exec] part of the command not feeding properly. **The more important part and reason it brought up in this brief paragraph is that playing with the command gets you kicked off the DMZ and breaks the connection.** It would be too far out of scope of this write up to address that here but it is for sure an interesting tangent/rabbit hole to travel down as to how the security settings are configured. Screenshot 20-10 below shows all of that happening.

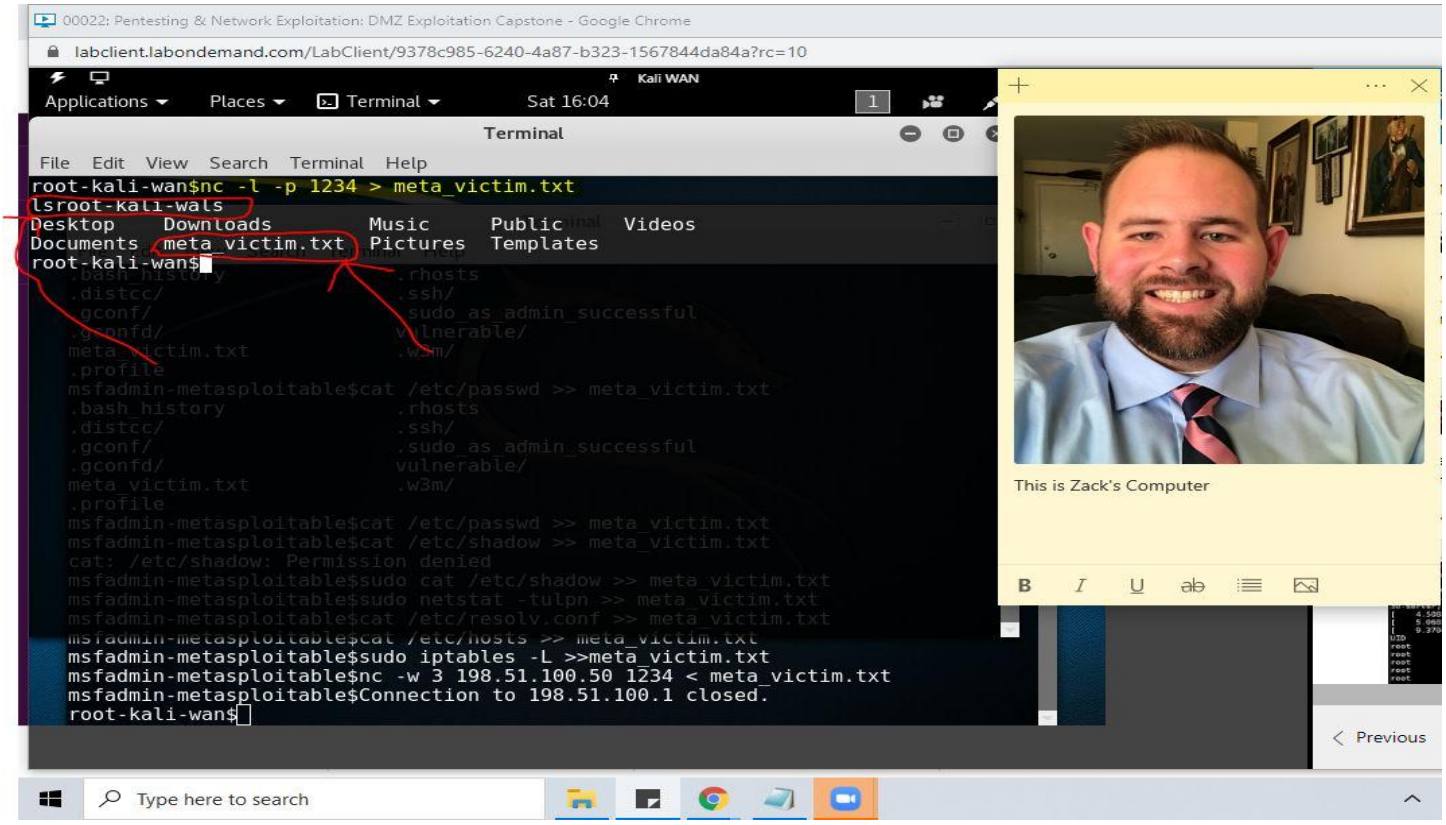


Screenshot 20-10

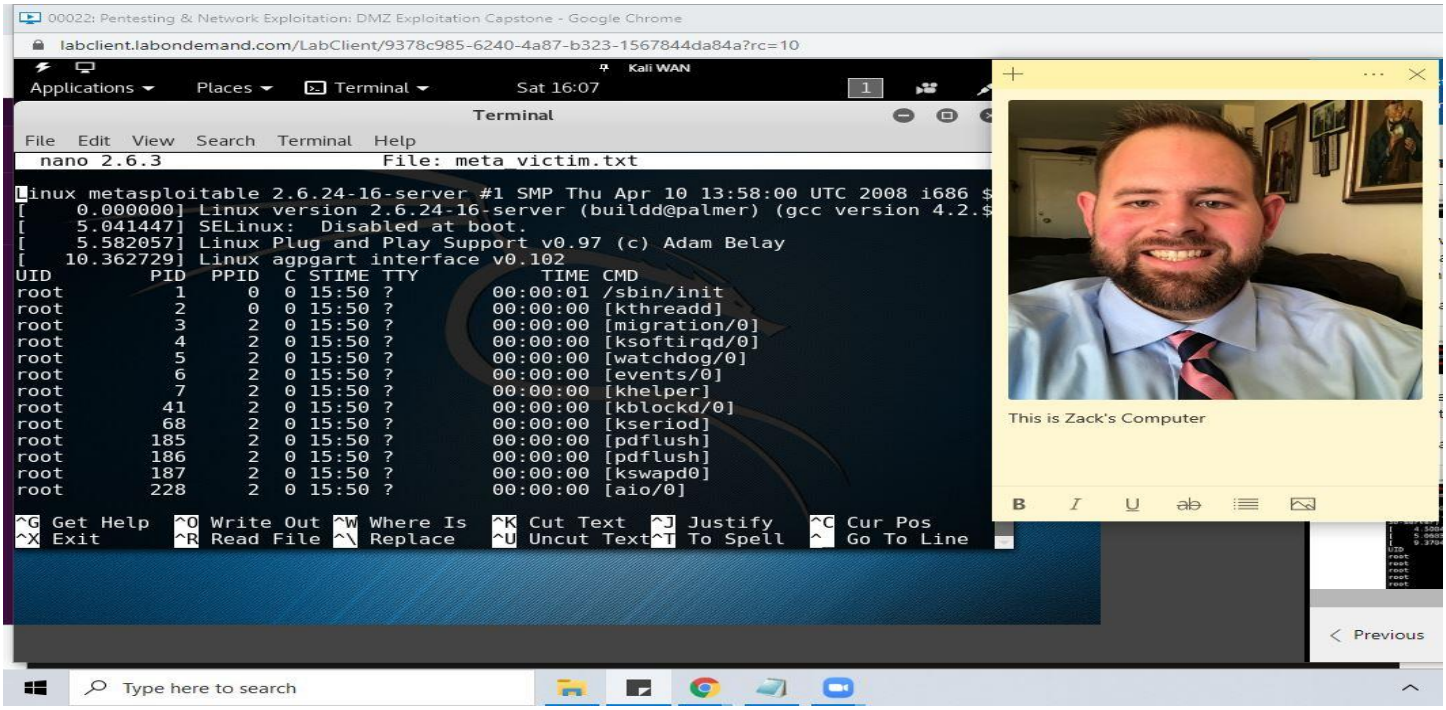
Once all of that information was placed into the meta_victim.txt file it was possible to netcat that out of the victim DMZ and onto our Kali Linux platform. That is shown below in screenshots 20-11 and 20-12. This ensures that the raw data is saved in a safe place if it is needed later. Screenshot 20-13 is proof of what that text file looked like when nano was used to view the contents. The entire contents of that file have been conglomerated in this additional file.



Screenshot 20-11

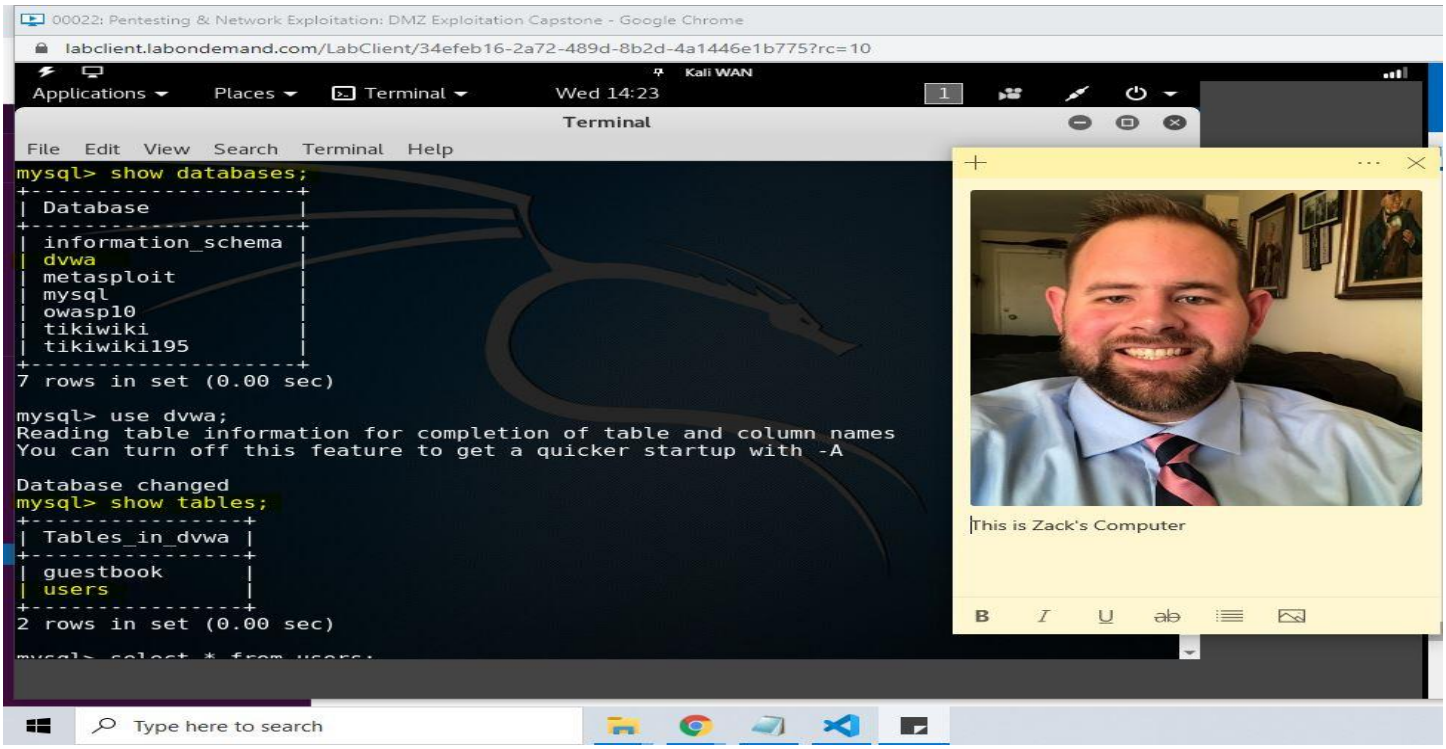


Screenshot 20-12



Screenshot 20-13

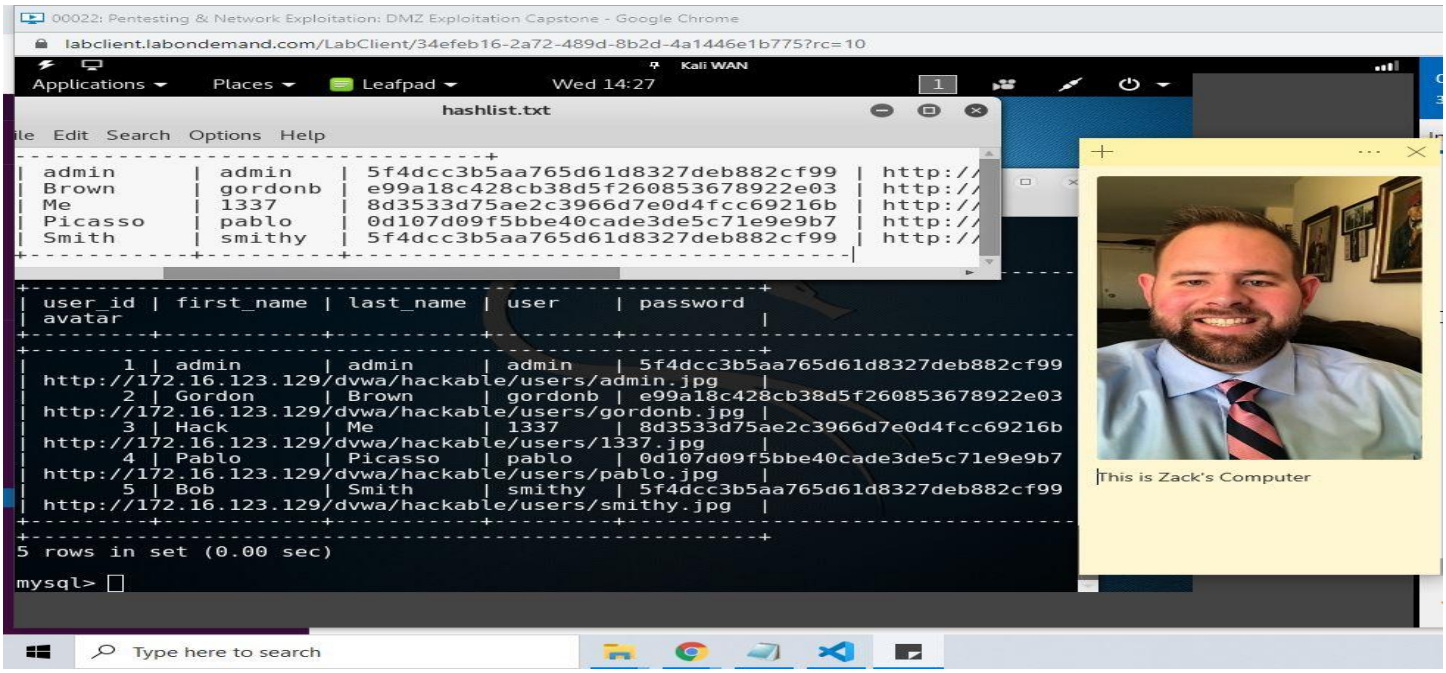
The next step is to use the victim machine against itself. We did this by using its mysql database to produce information. The steps to do this are in screenshot 20-14 below.



Screenshot 20-14

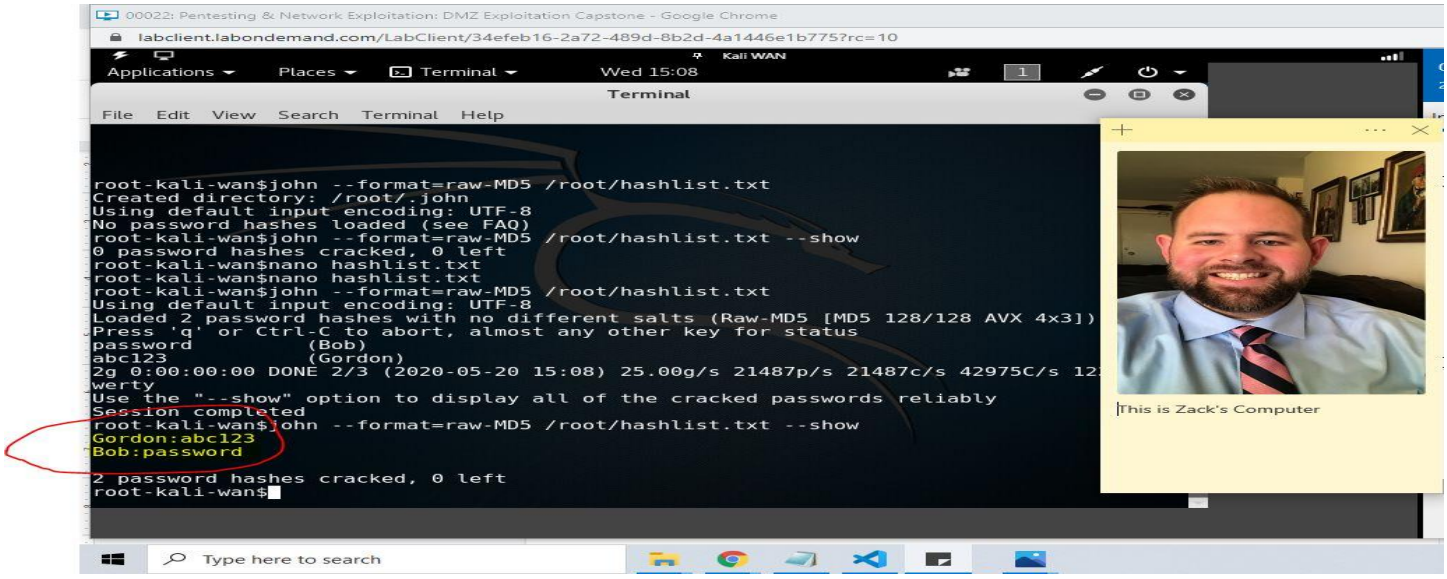
Once the users table in the dvwa database on mysql was displayed; it was possible to see all the users and their password hashes. The concept of this information being useful is that the same users are using the same passwords across multiple platforms. Once these hashes are resolved into plain text it would provide great information in terms of a forward attack. In order to make life a bit easier the contents from the mysql

database were copied and pasted into a hashlist.txt file on the Kali machine. This will make it far easier to resolve those hases. Screenshot 20-15 displays this below.



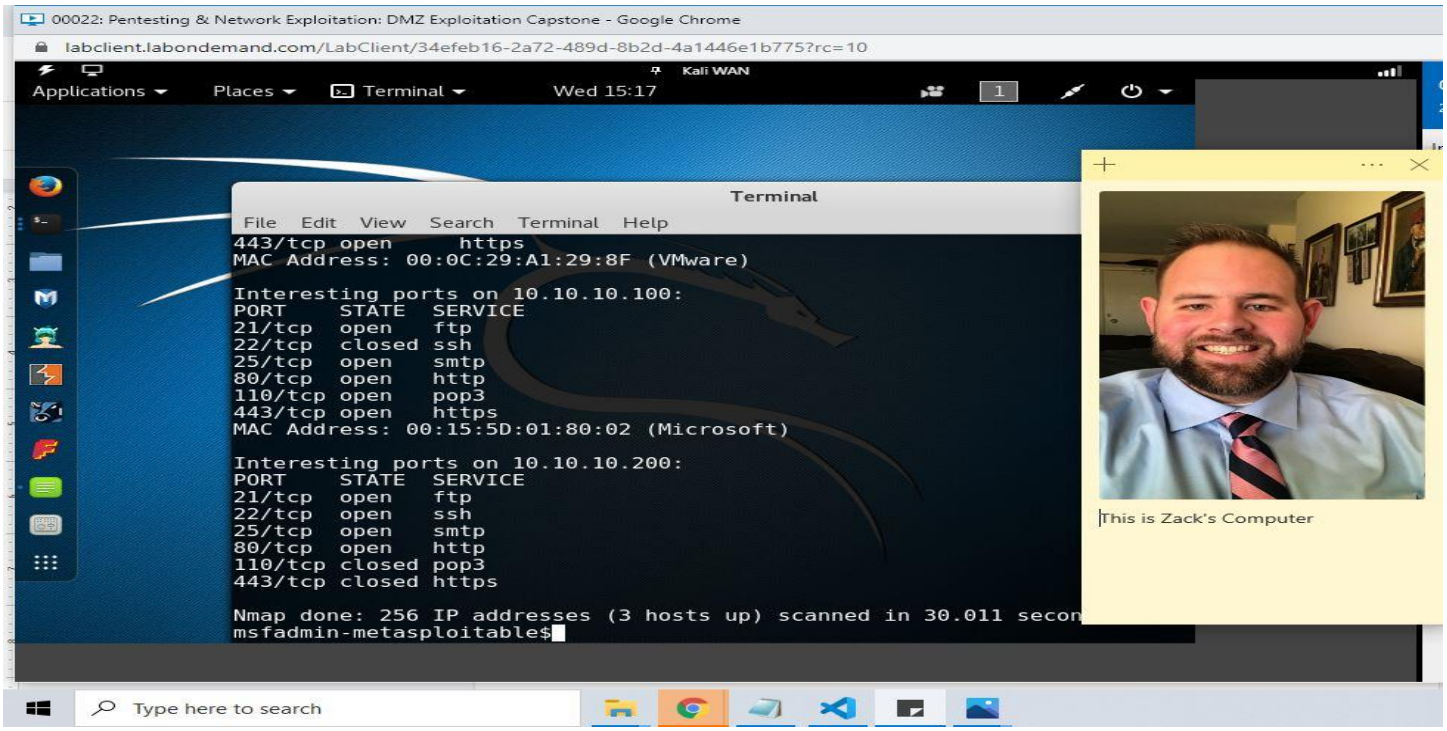
Screenshot 20-15

Having those password hashes allows the use of a tool like John the Ripper. In screenshot 20-16 you can see how John was able to resolve the passwords of both Gordon (abc123) and Bob (password). This information is helpful moving forward off the DMZ and onto machines that are connected to it.



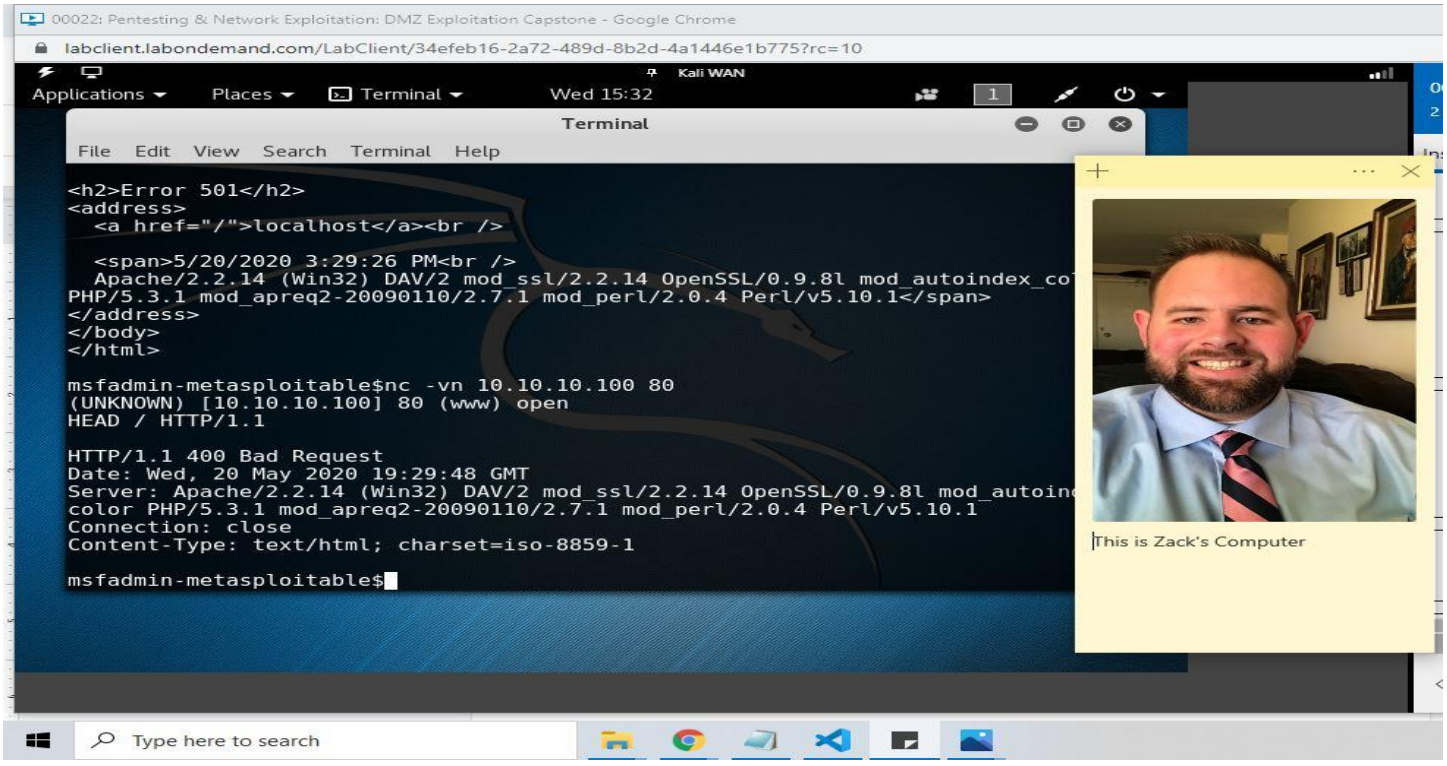
Screenshot 20-16

With that information obtained it was then appropriate to move out of mysql and begin to use nmap on the internal DMZ to see what machines were connected and which ports were left vulnerable. Screenshot 20-17 displays the results of that scan.



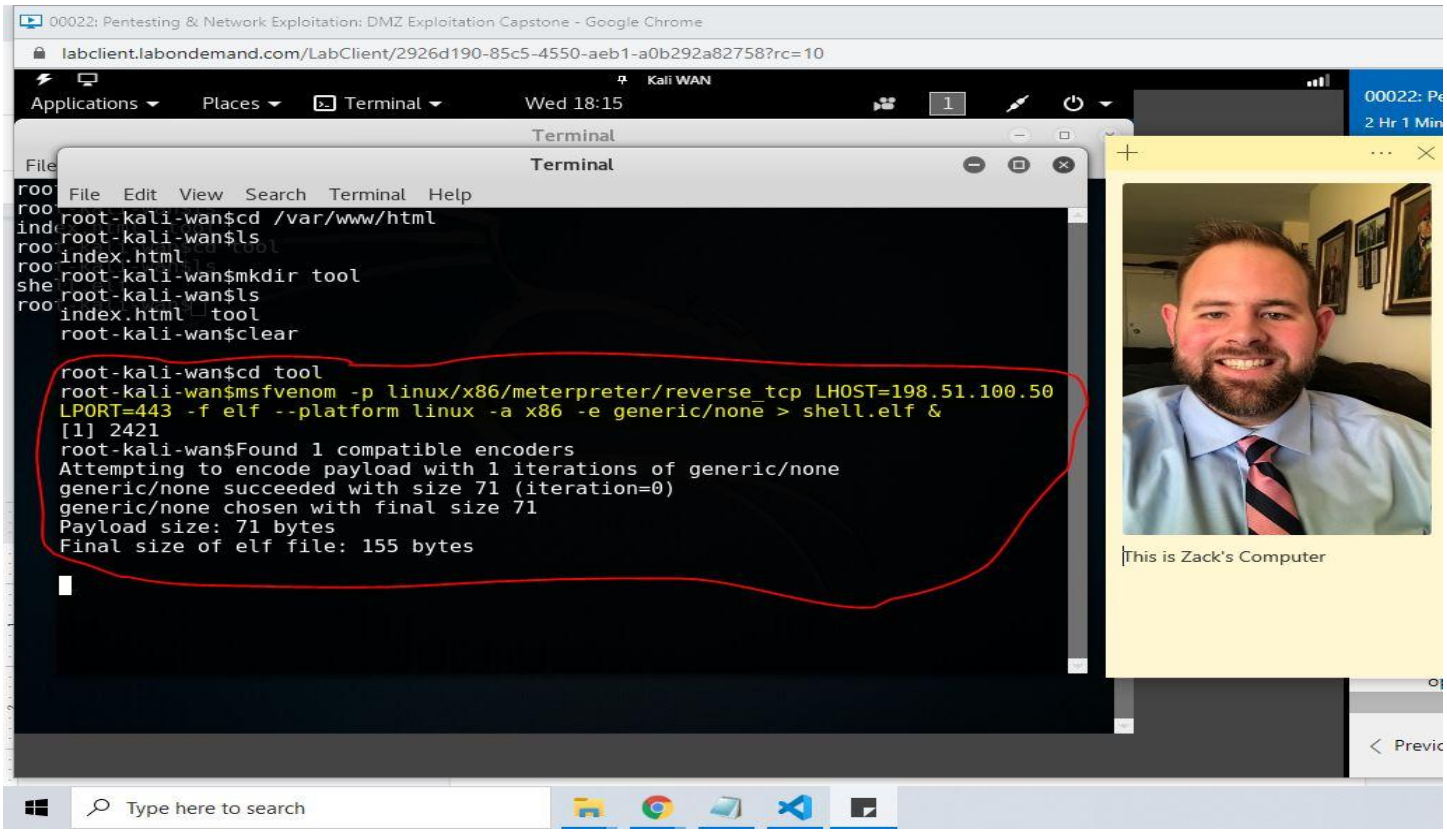
Screenshot 20-17

Screenshot 20-17 reveals two machines 10.10.10.100 and 10.10.10.200 with the displayed ports open. In order to continue it is necessary to use netcat in order to banner grab any information that is available. Screenshot 20-18 below shows that an Apache Version 2.2.14 server is operating and it is using DAV/2. It also looks like it was set up using Perl. That information is useful to tailor a unique exploit.

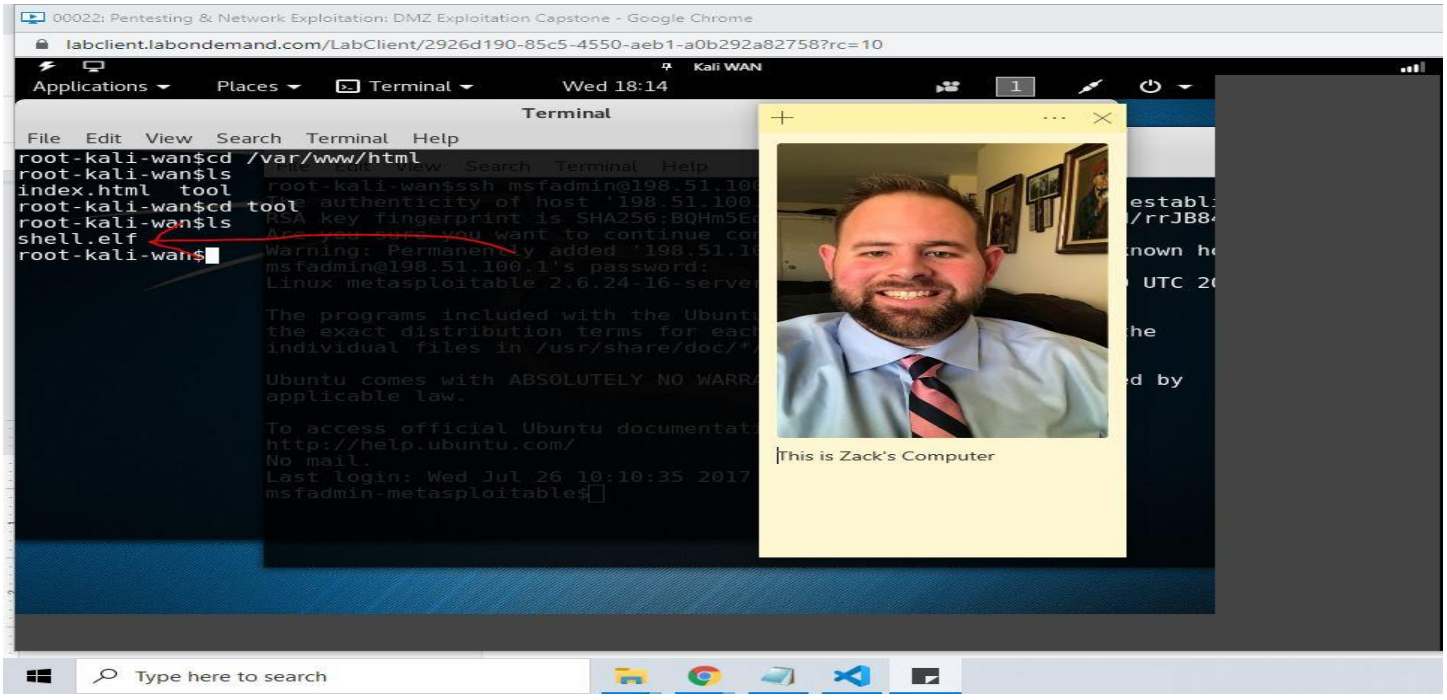


Screenshot 20-18

The next stage is to actually develop the exploit. In order to do this it is critical to have the ssh connection established in a background secondary terminal. At this point would normally be planning out the different parts on building a binary to exploit the internal segment. The class was not quite to that level and the commands to build the exploit were provided. Screenshot 20-19 shows the construction of that exploit and screenshot 20-20 shows that it was in fact created.



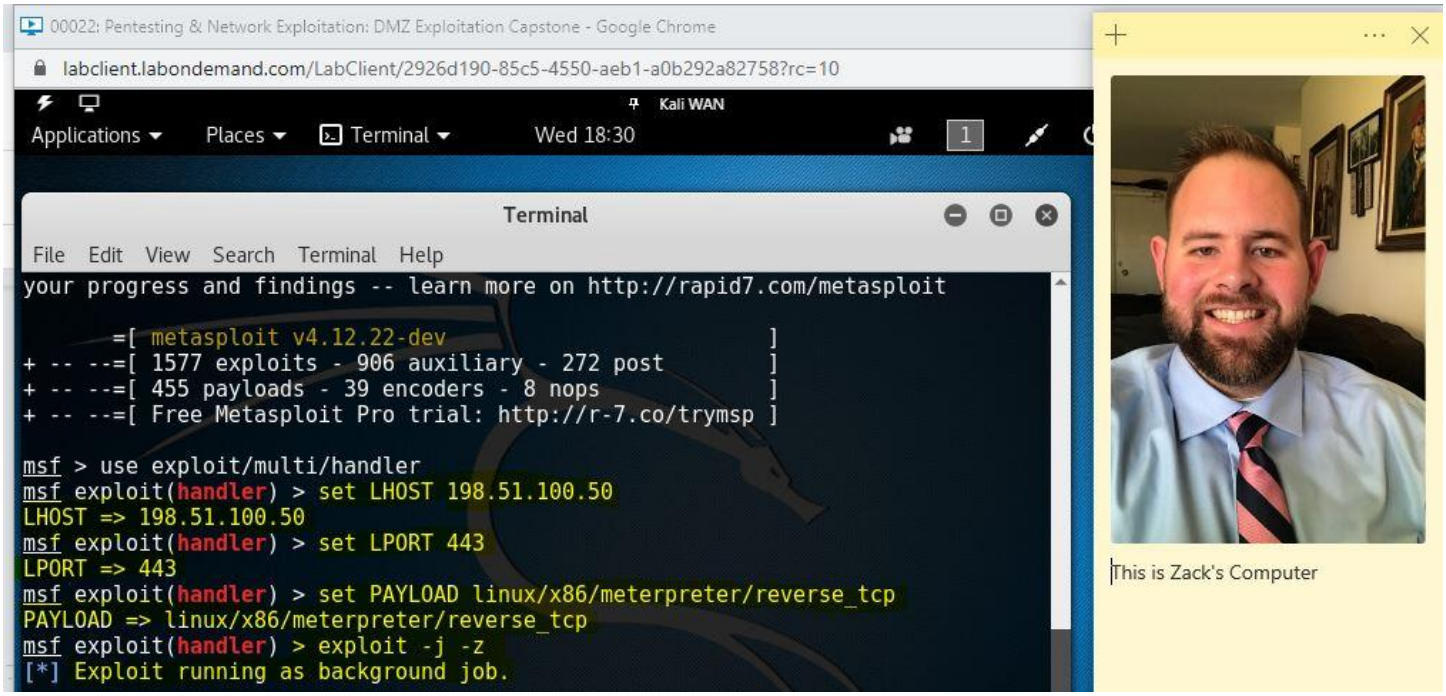
Screenshot 20-19



Screenshot 20-20

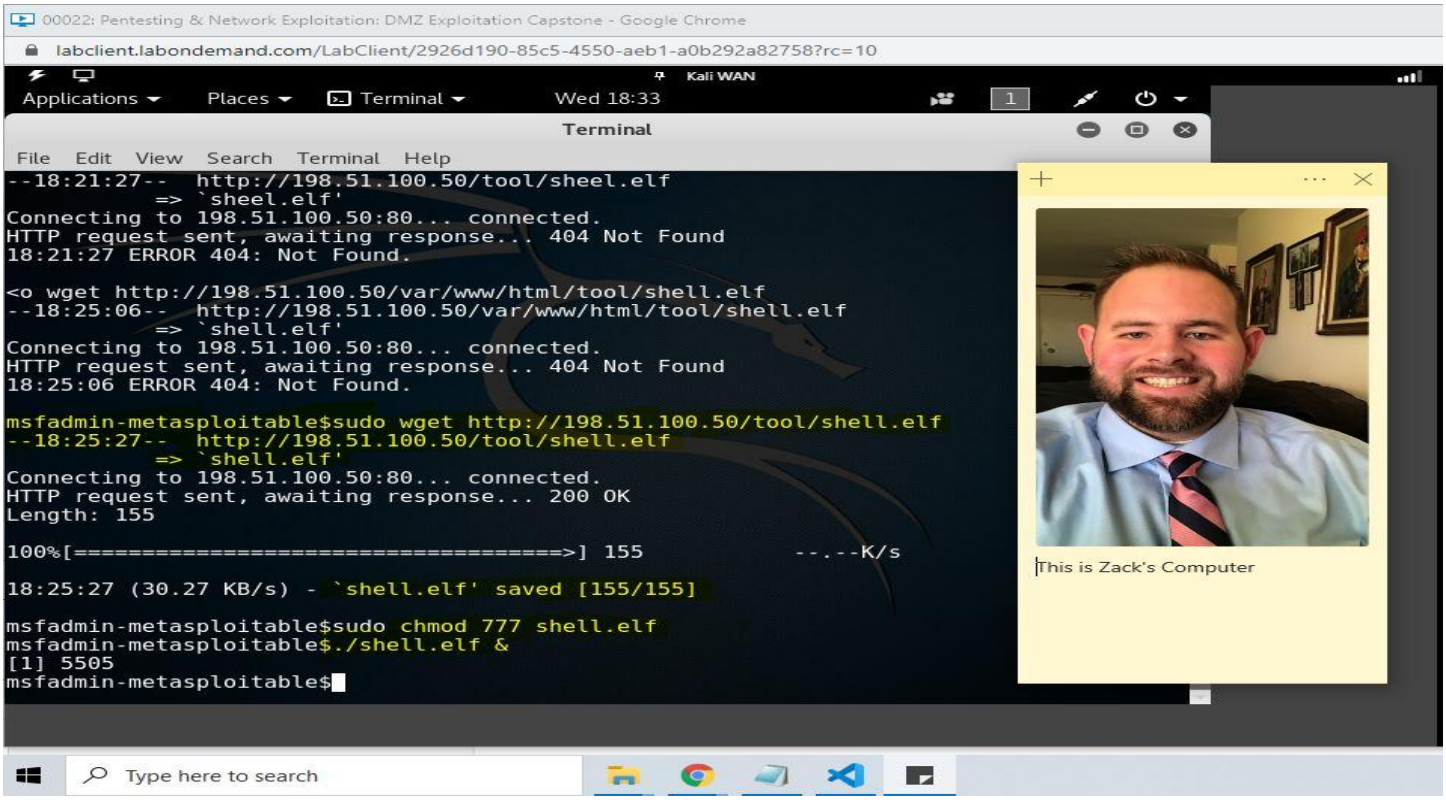
Once that binary script is ready the next step is to switch into another blank Kali terminal and start up the Apache Server. This next step is good to read a few times because it is literally a pivot moment. With the Apache Server now started there is a possible way to transfer the shell.elf executable. In yet another terminal Metasploit has been configured to run the handler. In order to recap there should be SEVERAL terminals now open. The should be (a)a plain kali terminal, (b) an ssh connected terminal to msfadmin@198.51.100.1, (c) the terminal where shell.elf was created and running in the background, (d) a metasploit connect terminal using the handler exploit.

Screenshot 20-21 below shows that metasploit in terminal (d) is running handler. Handler is running as a background because what this is doing is listening for communication from a matching IP address and port that were assigned in the options. When it hears that connection it will connect. This connection will eventually come by running the shell.elf binary executable.

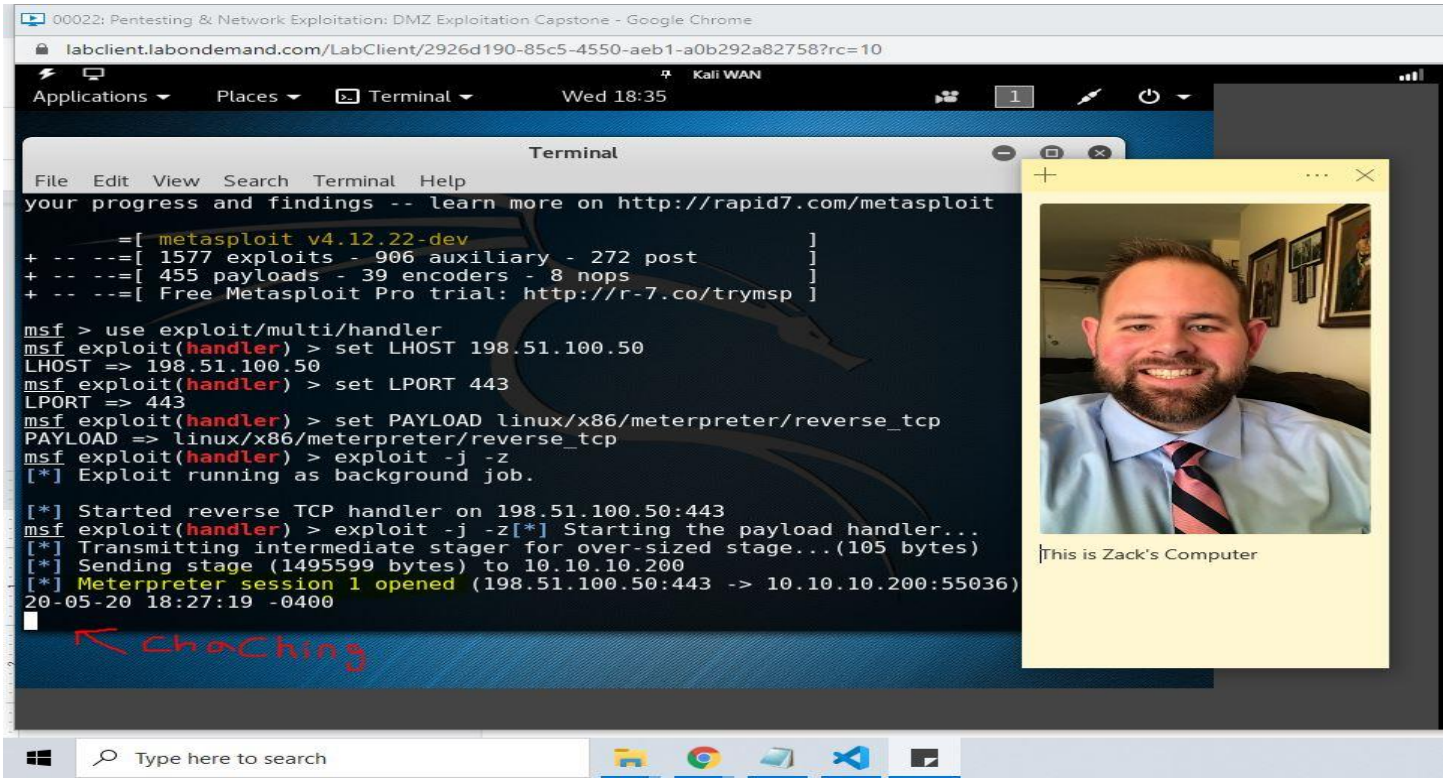


Screenshot 20-21

While that is listening it is now necessary to switch to terminal (b) which is running the ssh connection to msfadmin. A wget command is issued which tells the Apache Server to transfer the created binary file to the exploited msfadmin account. It should be clear that at first the shell.elf was created on the KALI terminal (a) and is now transferred onto the MSFADMIN terminal (b). On this terminal a simple chmod 777 is done to make it executable and then a ./shell.elf to run the script. Screenshot 20-22 below shows this and below that screenshot 20-23 shows the result of this binary. Upon return to the metasploit terminal it is clear that a meterpreter session has been opened within metasploit. If it is not obvious this means a tunnel has been forged. Rather than having terminals (a), (b), and (d) running separate from each other; a is controlling d which is in turn controlling (b). Terminal (c) is only necessary to run in the background to allow shell.elf to operate. On the surfact this simply appears that everything is now controlled from metasploit. Screenshot 20-23 below also demonstrates how meterpreter has been opened within metasploit.

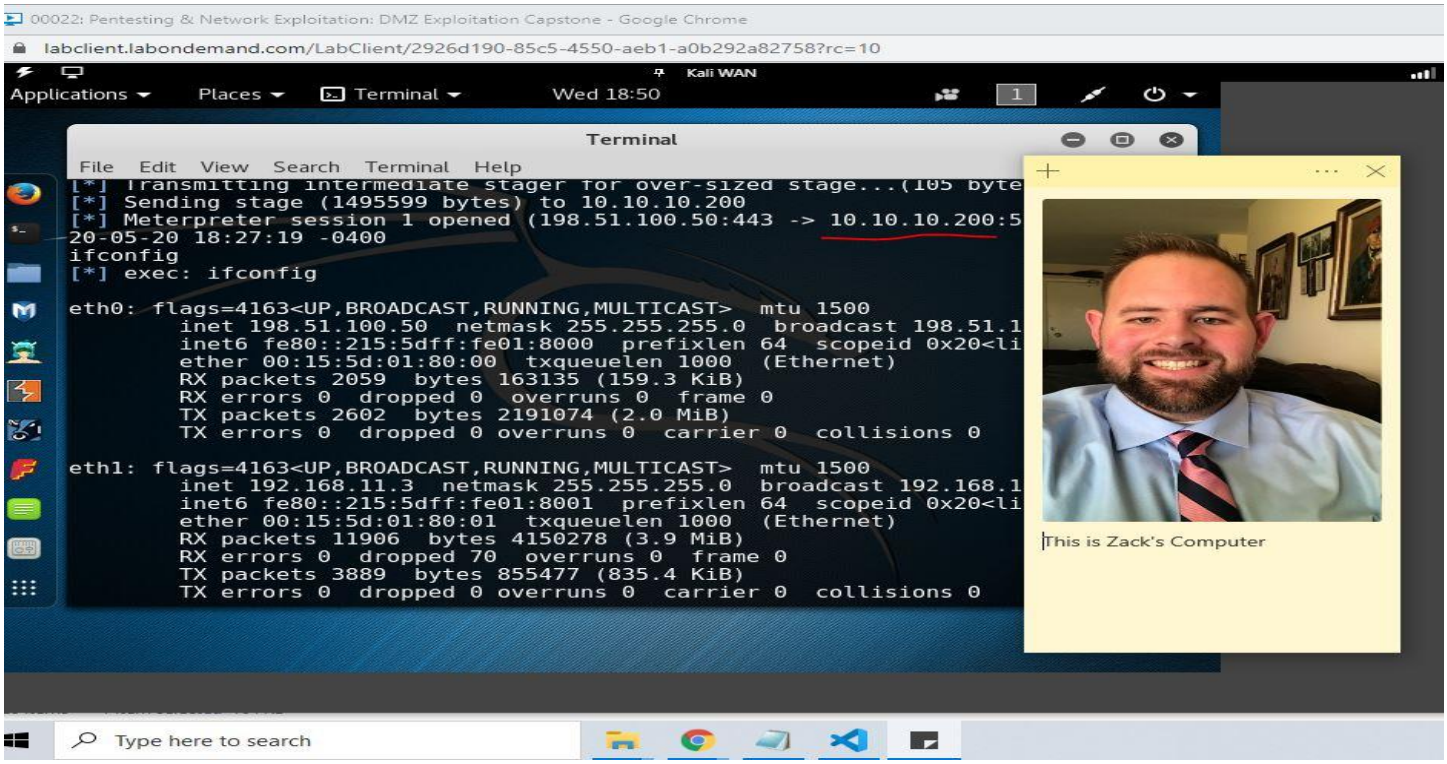


Screenshot 20-22



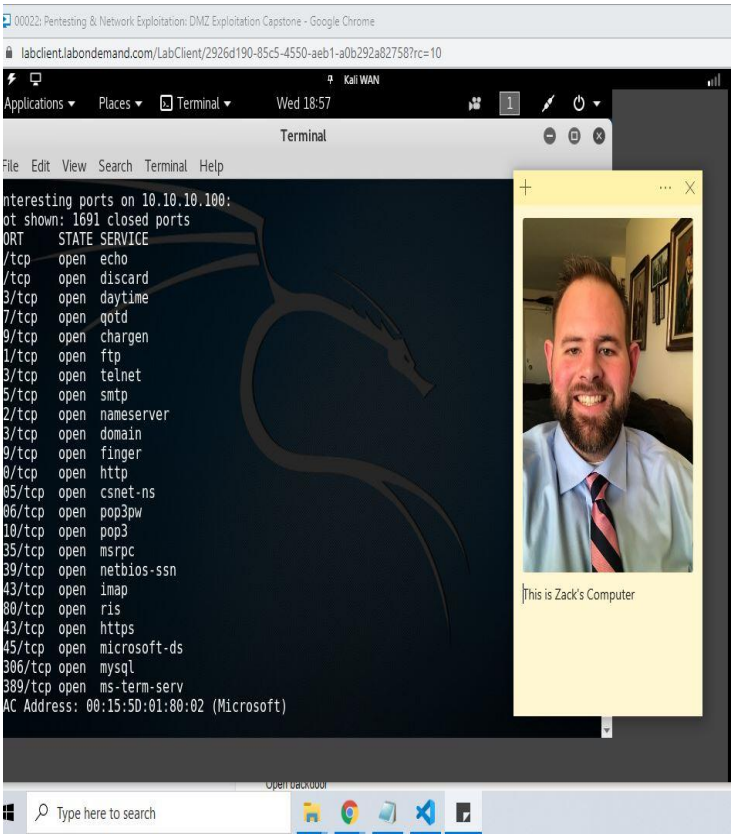
Screenshot 20-23

Using meterpreter it is possible to run ifconfig on the internal segment (the former terminal b). The results of the ifconfig are not that telling but what is telling is observing what ifconfig connected to within the internal segment. In screenshot 20-24 it is clearly connected to an IP address 10.10.10.200.

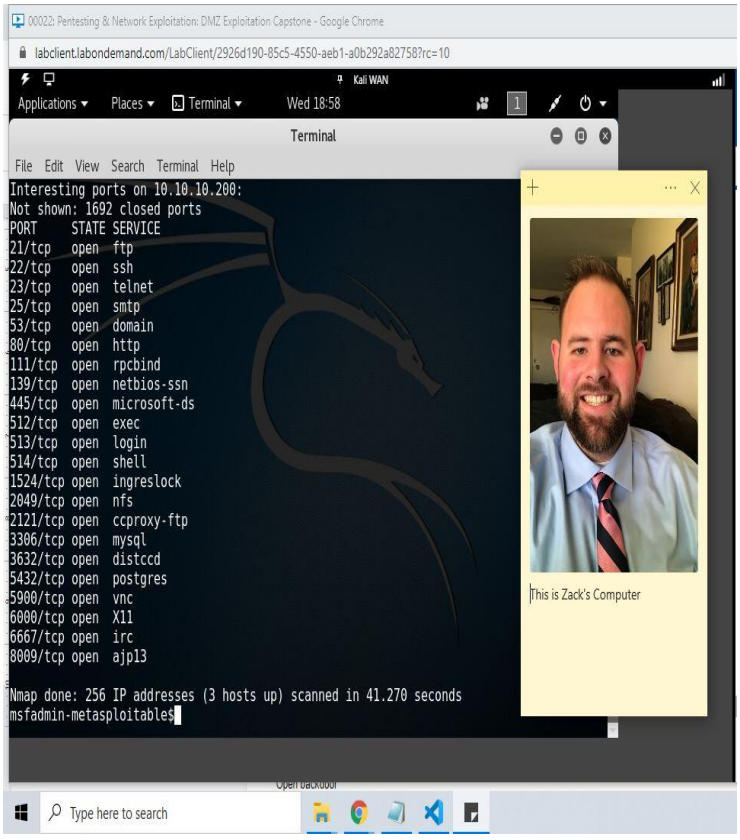


Screenshot 20-24

The connection to 10.10.10.200 suggests that running nmap would be smart to run and determine what other objects are on that IP address. Screenshots 20-25 and 20-26 below show the results of that NMAP Scan.



Screenshot 20-25

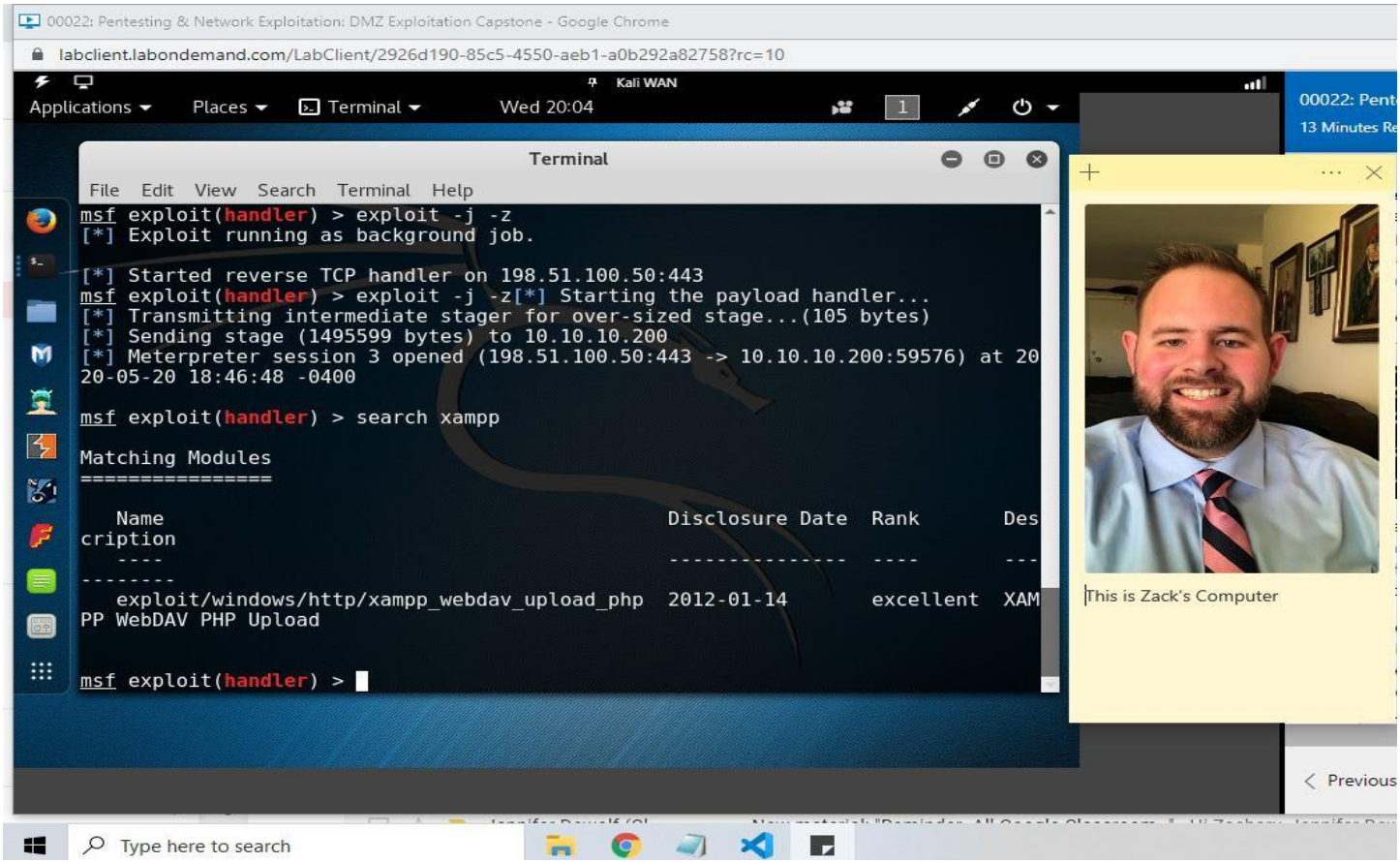


Screenshot 20-26

Now that IP addresses 10.10.10.100 and 10.10.10.200 have been identified and their vulnerable ports revealed it is possible to start working on which exploit will work best for this particular situation. In order to develop clarity on what this is doing from start to beginning. This project started on the Kali machine 198.168.11.3 and then used metasploit to find login credentials and login via ssh to an internal linux server on IP address 198.51.100.50. At that stage several steps were taken including the build of a custom binary executable to utilize the internal linux server on the DMZ almost like a router to reverse connect to the other devices on the DMZ. These other devices 10.10.10.100 and 10.10.10.200 responded. The steps were taken to consolidate the work so that the Kali machine could be in control via metasploit and the steps below are now utilizing metasploit which is already inside the DMZ to get through the firewall to devices on the other side.

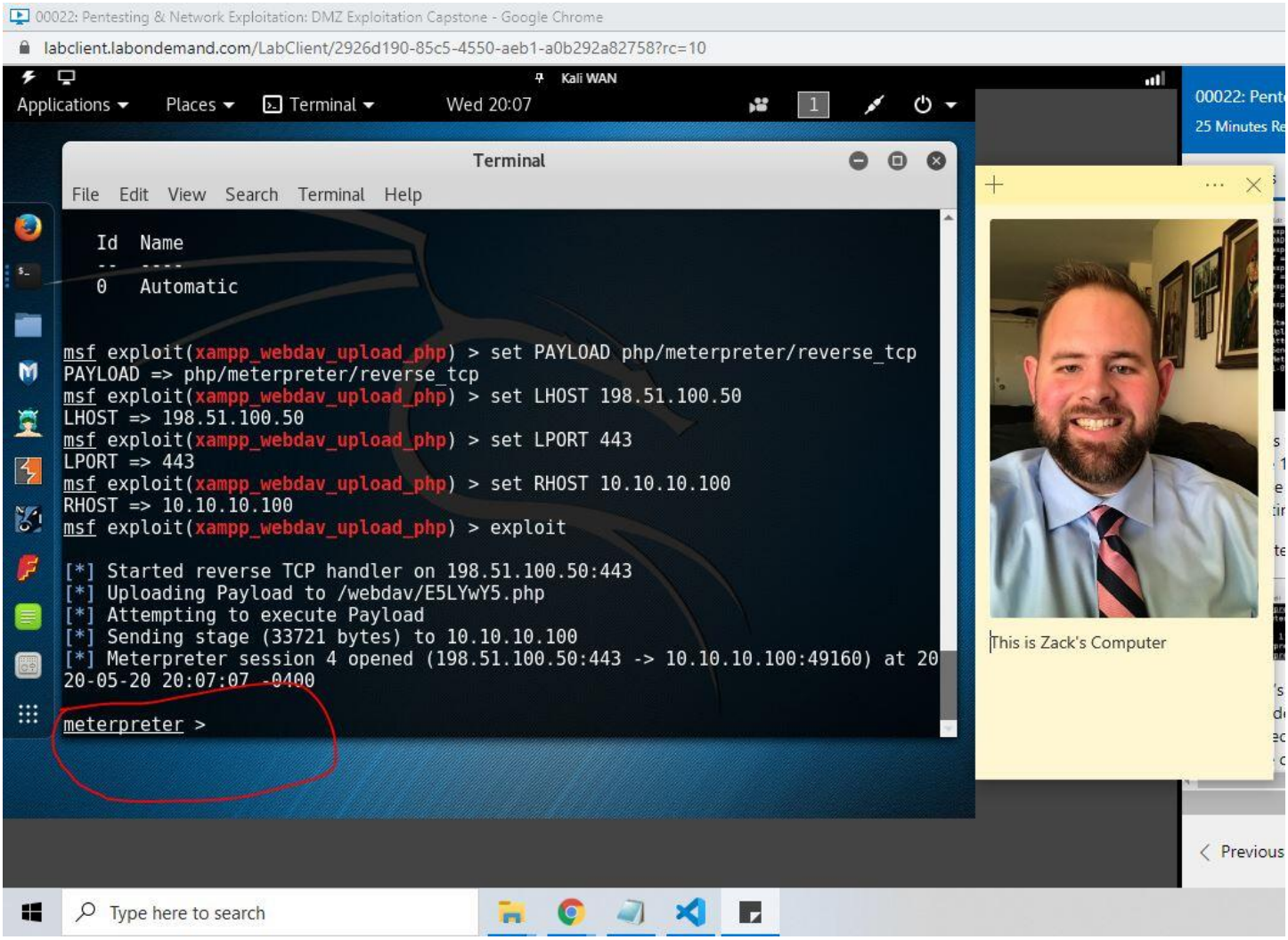
The process of getting to devices on the other side will require a well chosen exploit. The best approach to this is to know what vulnerabilities already exist. It is known that the Apache server we are currently controlling has DAV2. DAV2 is a web application plugin that allows users to share, copy, edit, and move files. A citation for that information and some light bedtime reading can be found here.

WebDAV, according to some online blogs, is often left unsecured because it is an overlooked feature that is taken for granted. This blog discusses the issue further and provides enough rationale to attempt an exploit using XAMPP and DAV. An attempt to search for XAMPP exploits with Metasploit made the choice quite easy as there was only one choice. Screenshot 20-27 shows this below.



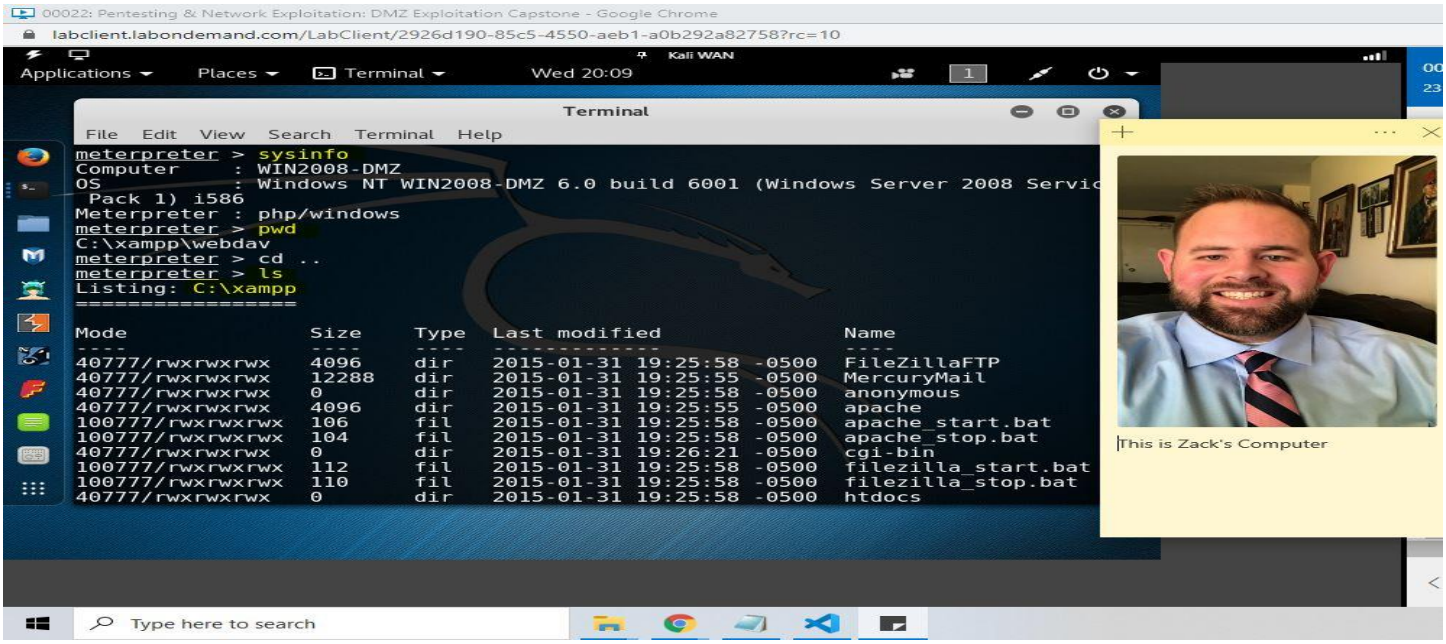
Screenshot 20-27

The next steps were fairly straight forward. The options must be set and those can be seen in screenshot 20-28. At the bottom of the same screenshot it is visible that a meterpreter session has been opened on the other side of the DMZ giving access to the final target's machine.



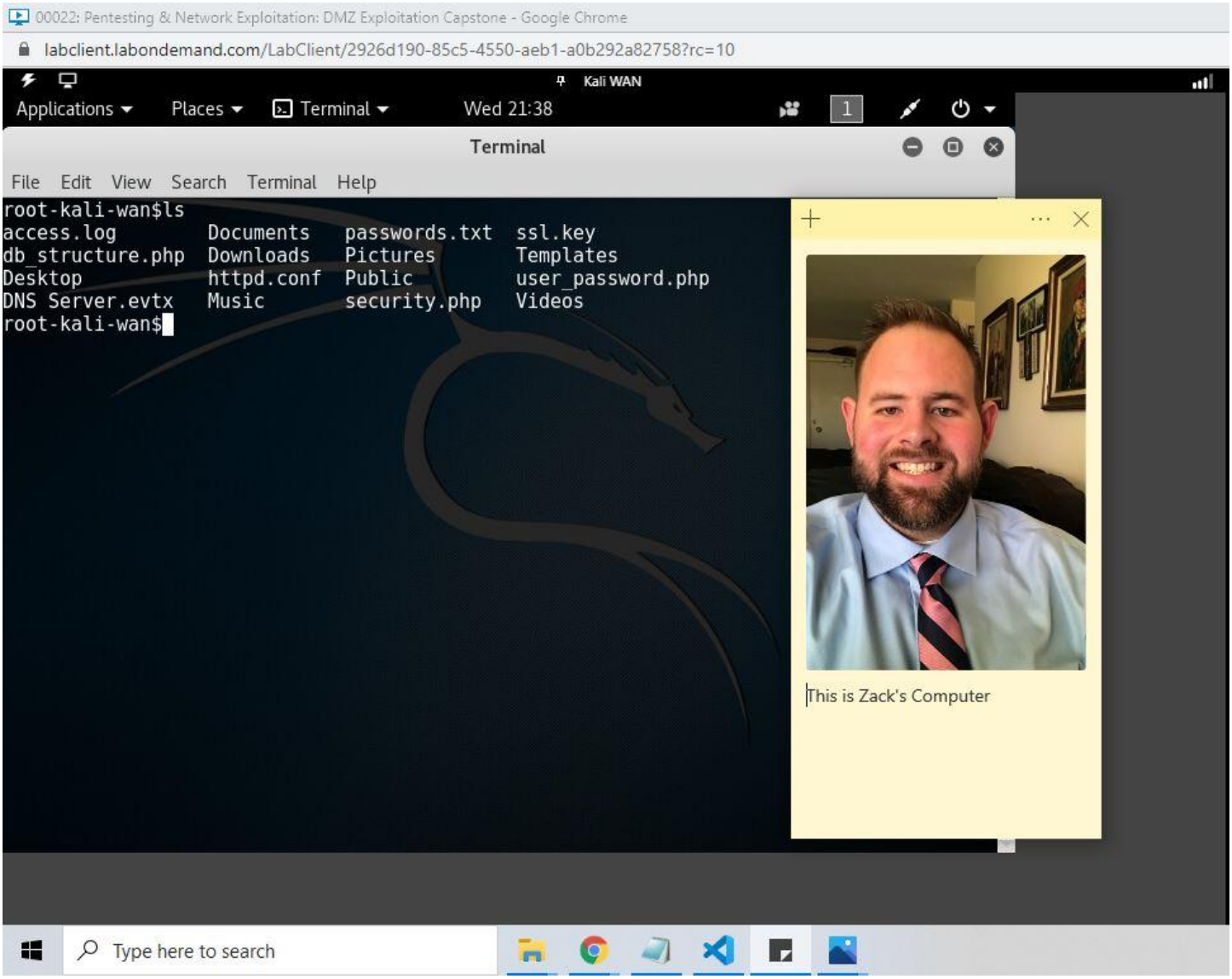
Screenshot 20-28

Screenshot 20-29 shows the starting position on the xampp server upon break in.



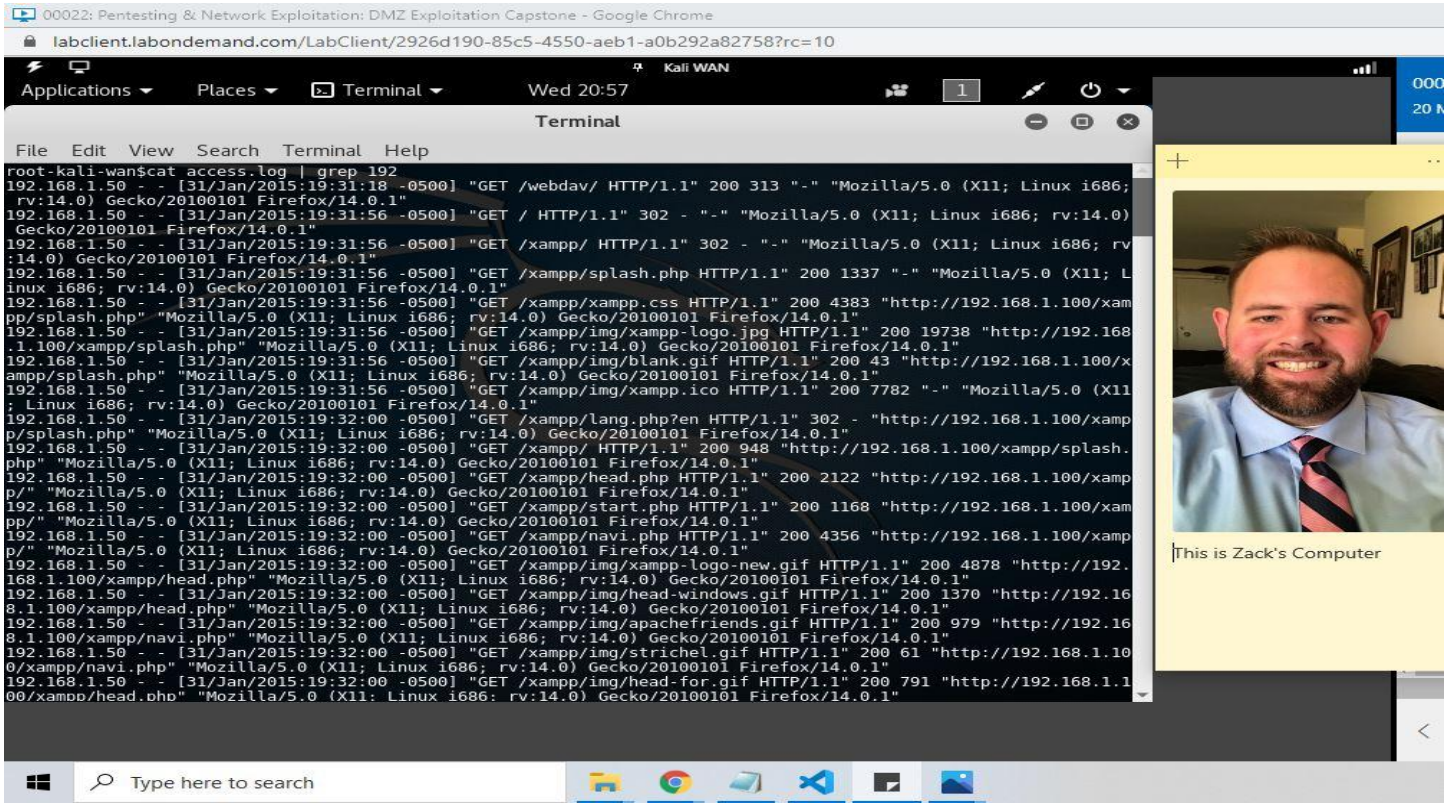
Screenshot 20-29

The assignment called for a list of specific files to be downloaded from the victim. There were other files analyzed, of course, but these are the primary files requested. The files requested were: security.php; httpd.conf; and ssl.key, db_structure.php, and user_password.php. Screenshot 20-30 shows evidence that this was performed successfully.



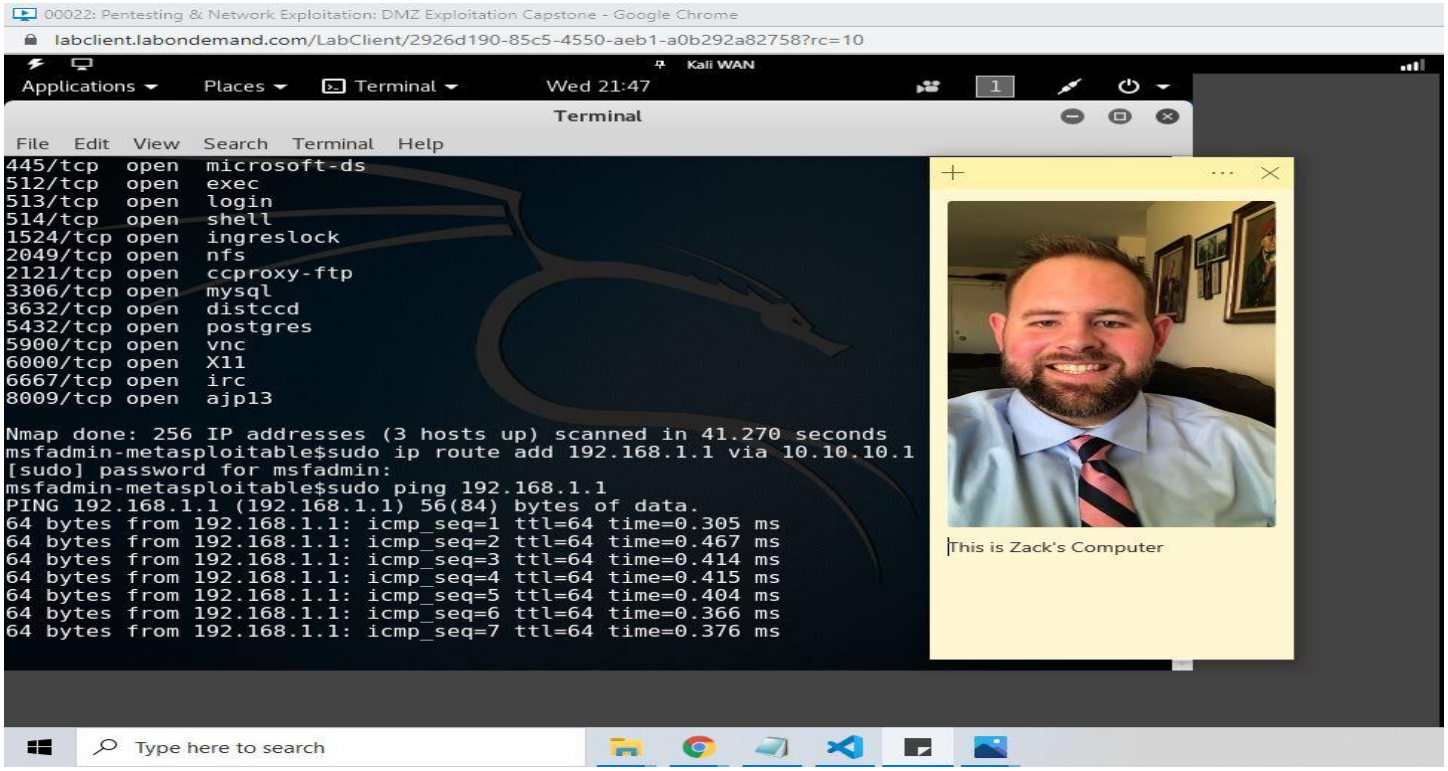
Screenshot 20-30

Screenshot 20-31 shows evidence that those files open. The particular displayed in the screenshot is the access logs which show evidence of an entry break. This will be addressed below.



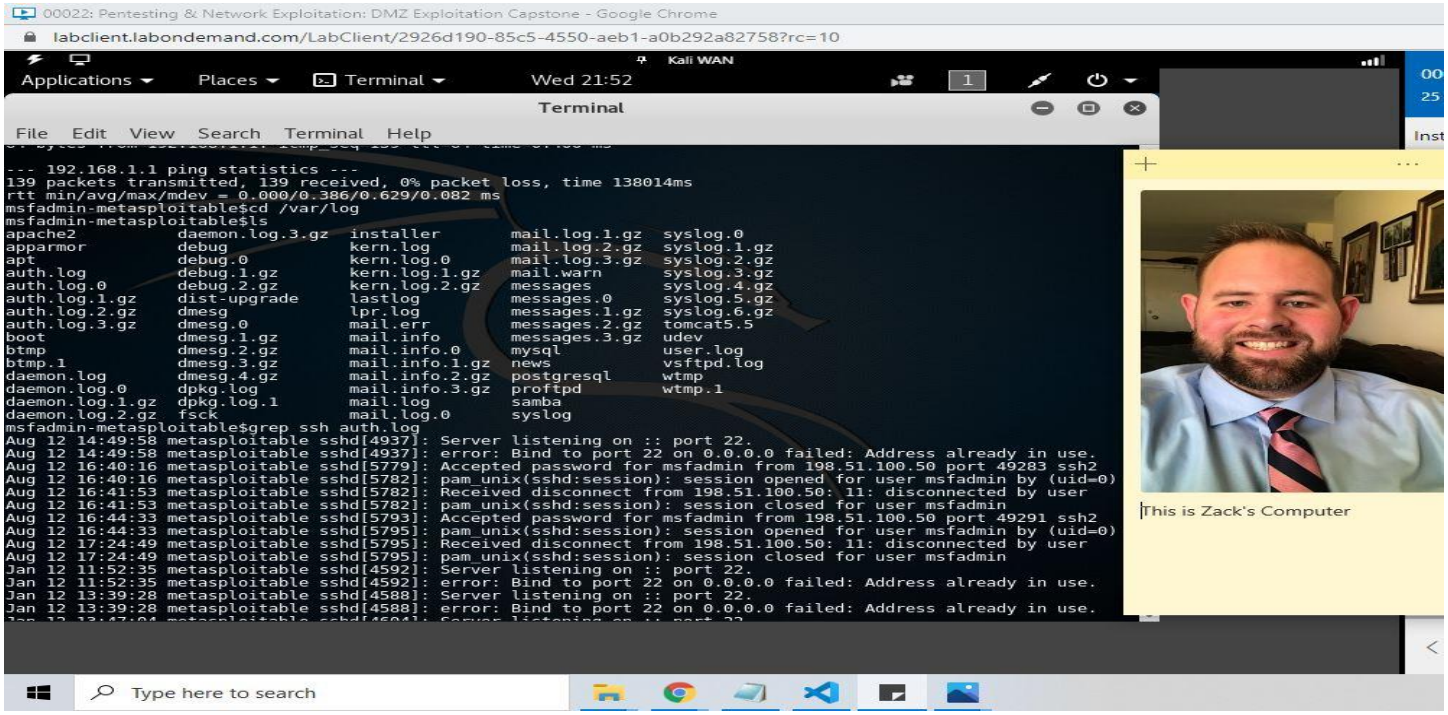
Screenshot 20-31

Another wise step that was taken was to build a route between 192.168.1.1 and 10.10.10.1 so that it can all be accessed more easily from the outside. This saves time in the future rather than repeating these steps. It is accomplished by returning to the ssh exploited account under msdadmin and using some simple commands at that location. Screenshot 20-32 shows the creation of this path and successful ping attempts.



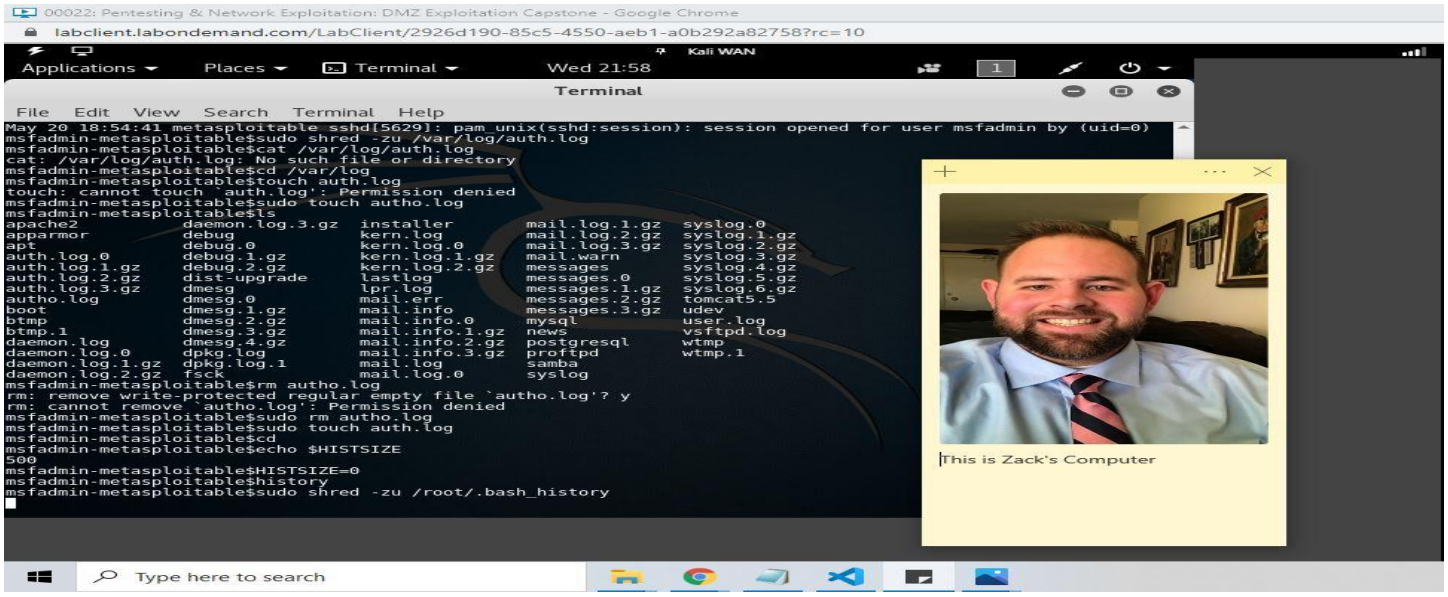
Screenshot 20-32

The last part of this project is to go back and cover any tracks that were created by all the work. Eliminating all evidence is challenging but there are ways to make it harder for someone to find clues. On the exploited Linux host the most important evidence to trash would be the ssh logs. This can be done by switching into var/log and analysing what is in the auth.log. You can see in screenshot 20-33 that there is plenty of evidence to be had.



Screenshot 20-33

The final screenshot, 20-34, shows that the auth.log file has been completely wiped clean and that the history of bash commands has also met its maker. With the tracks covered a successful DMZ exploitation and breakin to distant device has been accomplished.



Screenshot 20-34