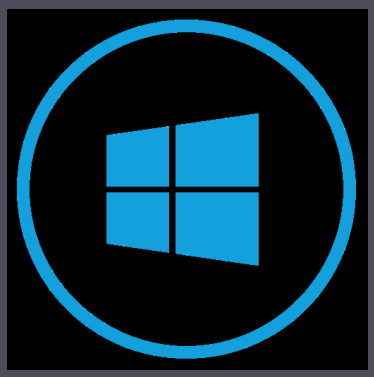
Attacktive Directory THM (Walkthrough)



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.

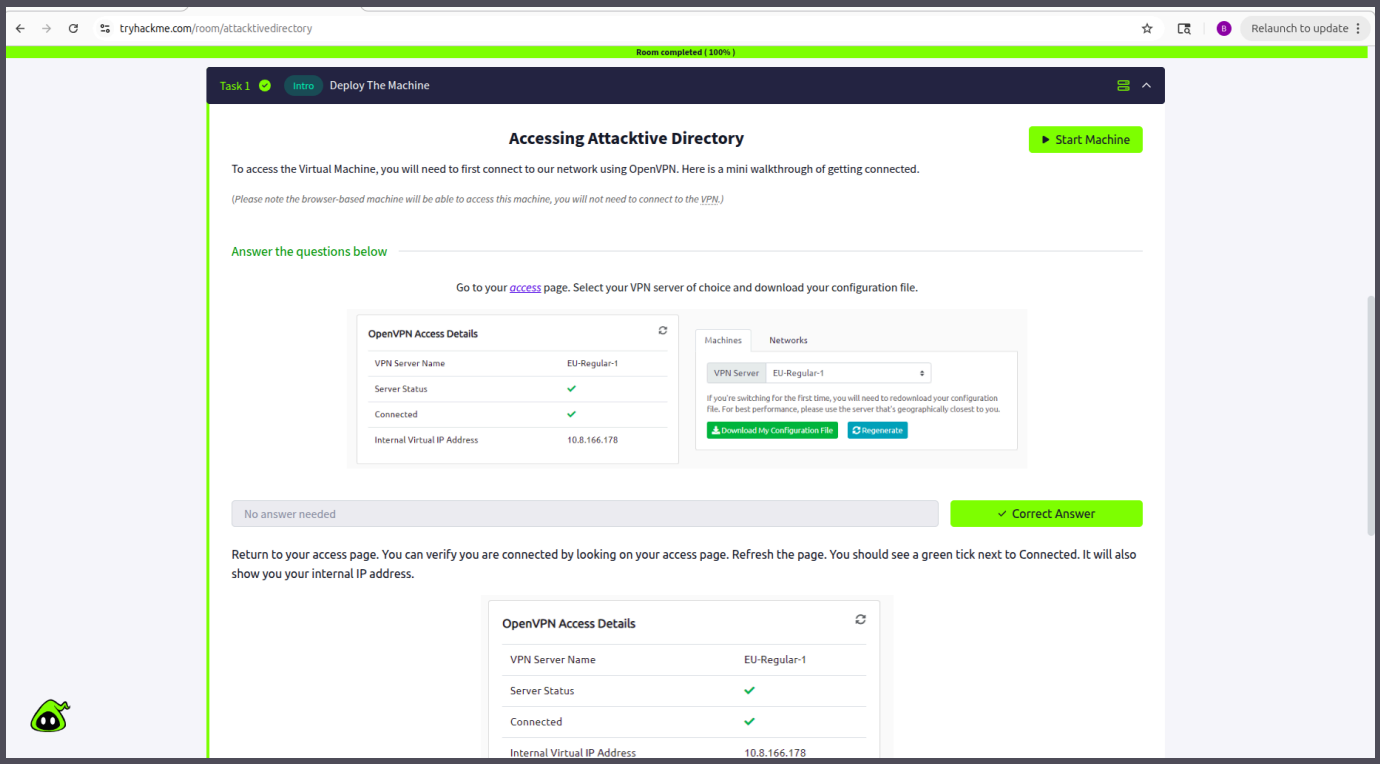
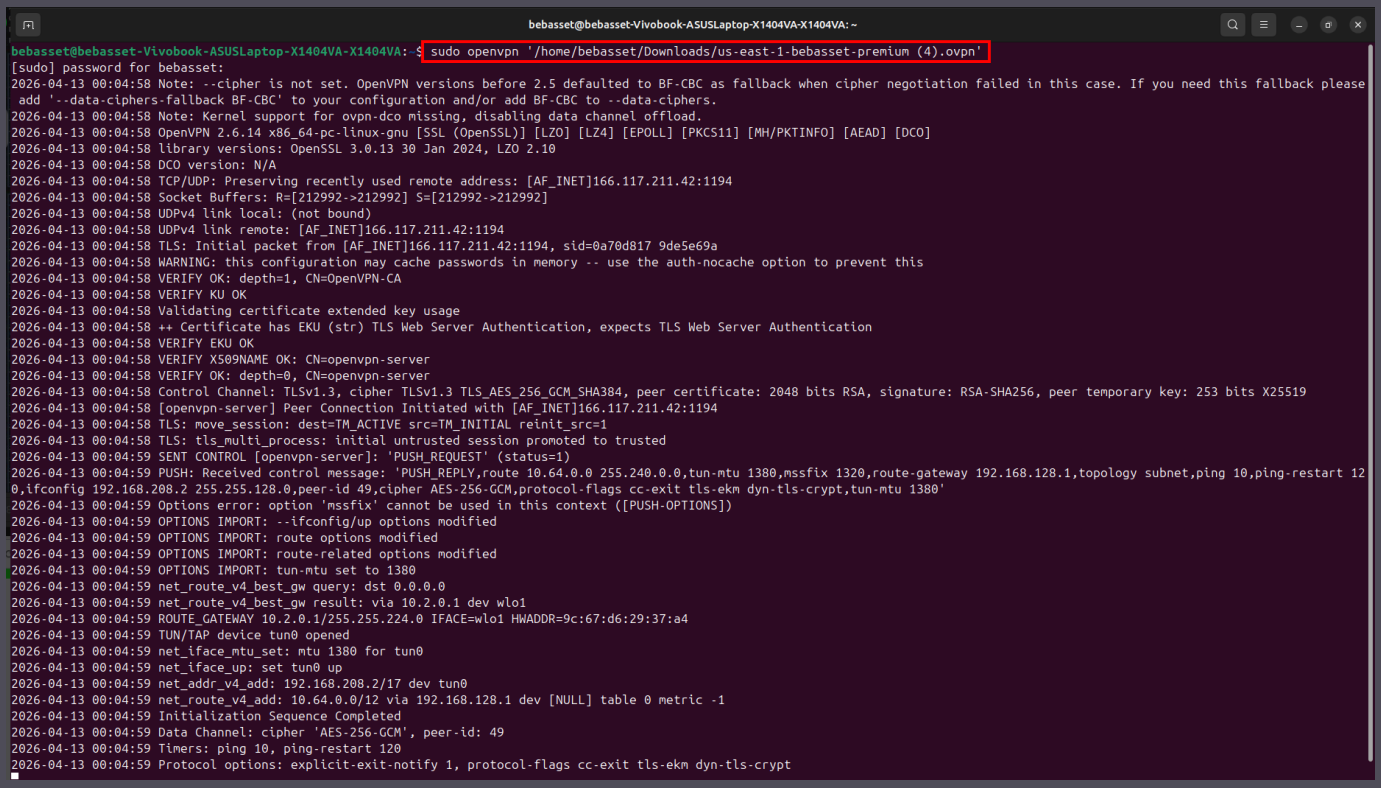


Figure 2. OpenVPN Connection Established from the Attack Host:

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.



Command Used:

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

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.

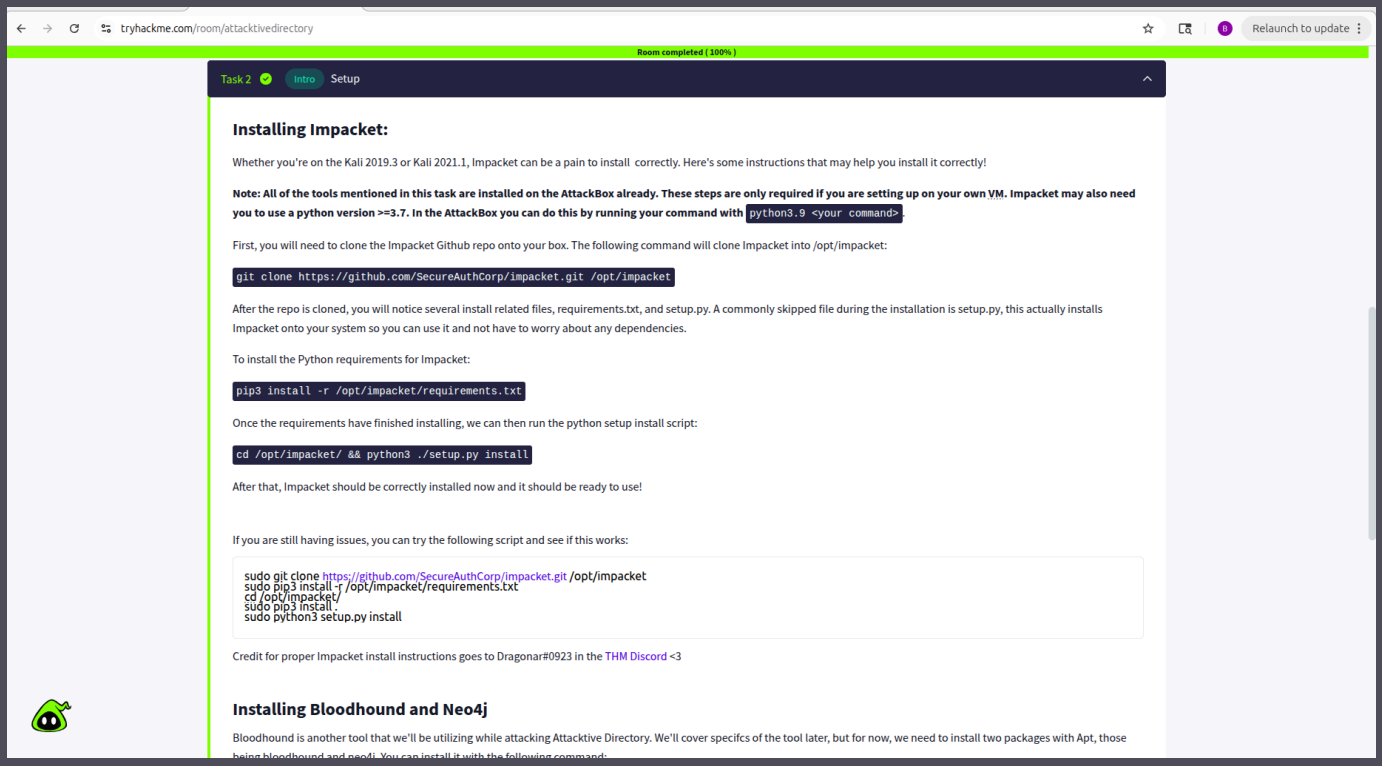
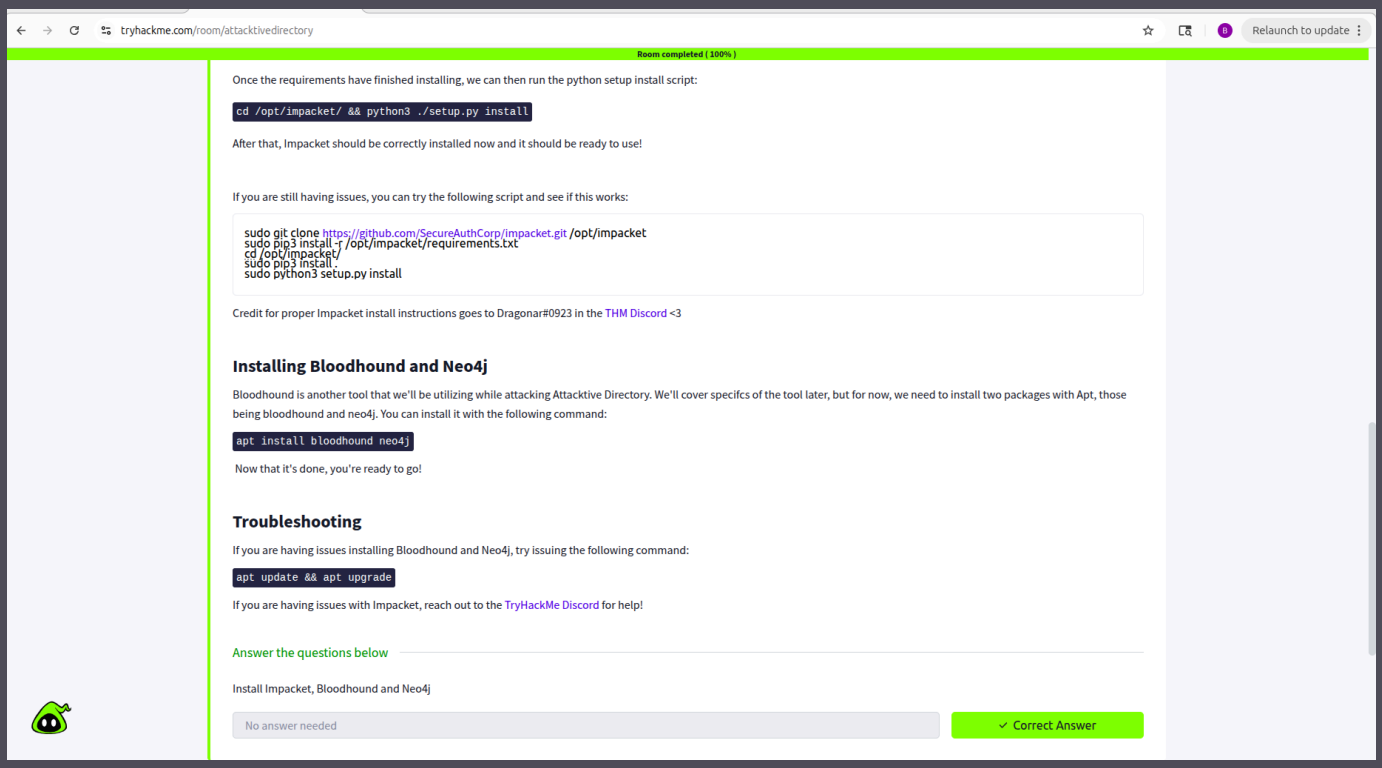


Figure 4. BloodHound, Neo4j, and Troubleshooting Setup:

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.



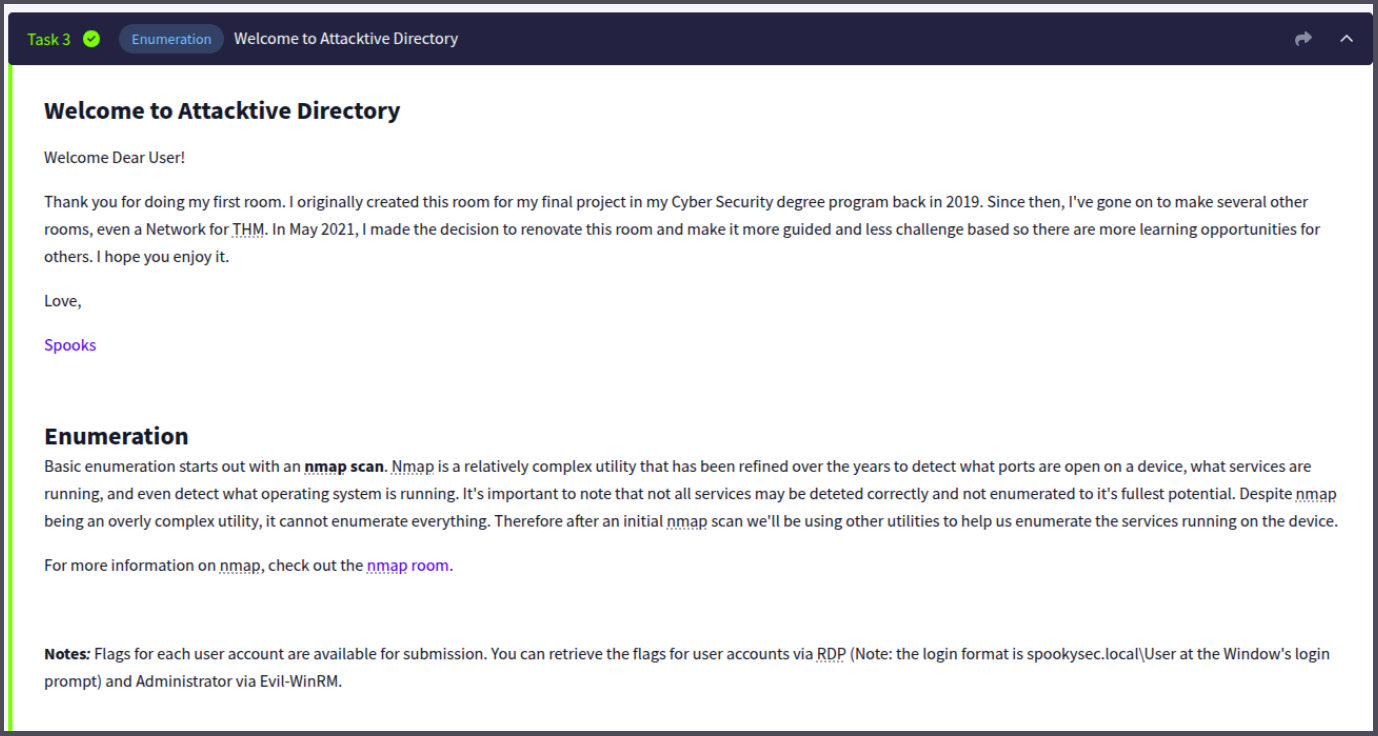
(I didn't use Bloodhound for this lab. Although it's are very useful for things like AD mapping and such 🙂)

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:

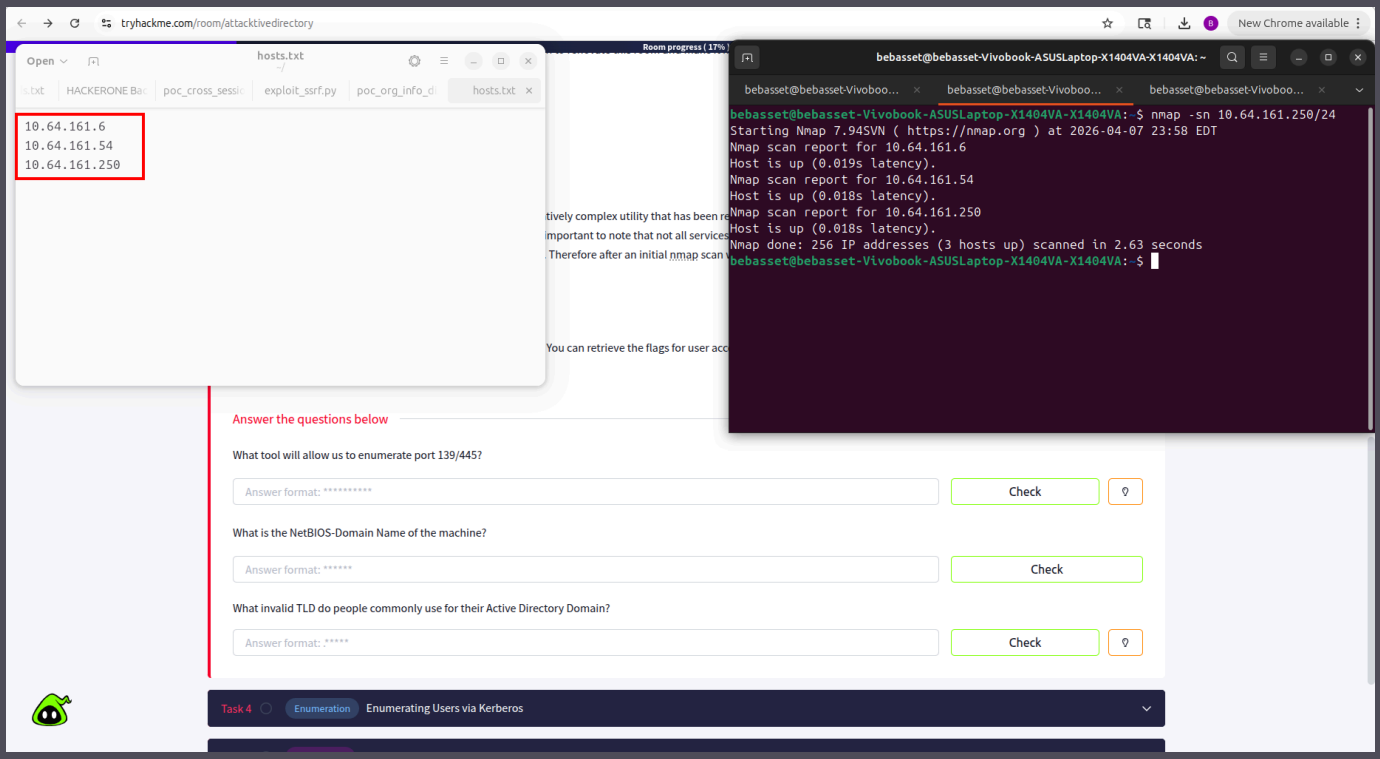
Here we review the room instructions and understand the recommended starting point for enumeration.



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.

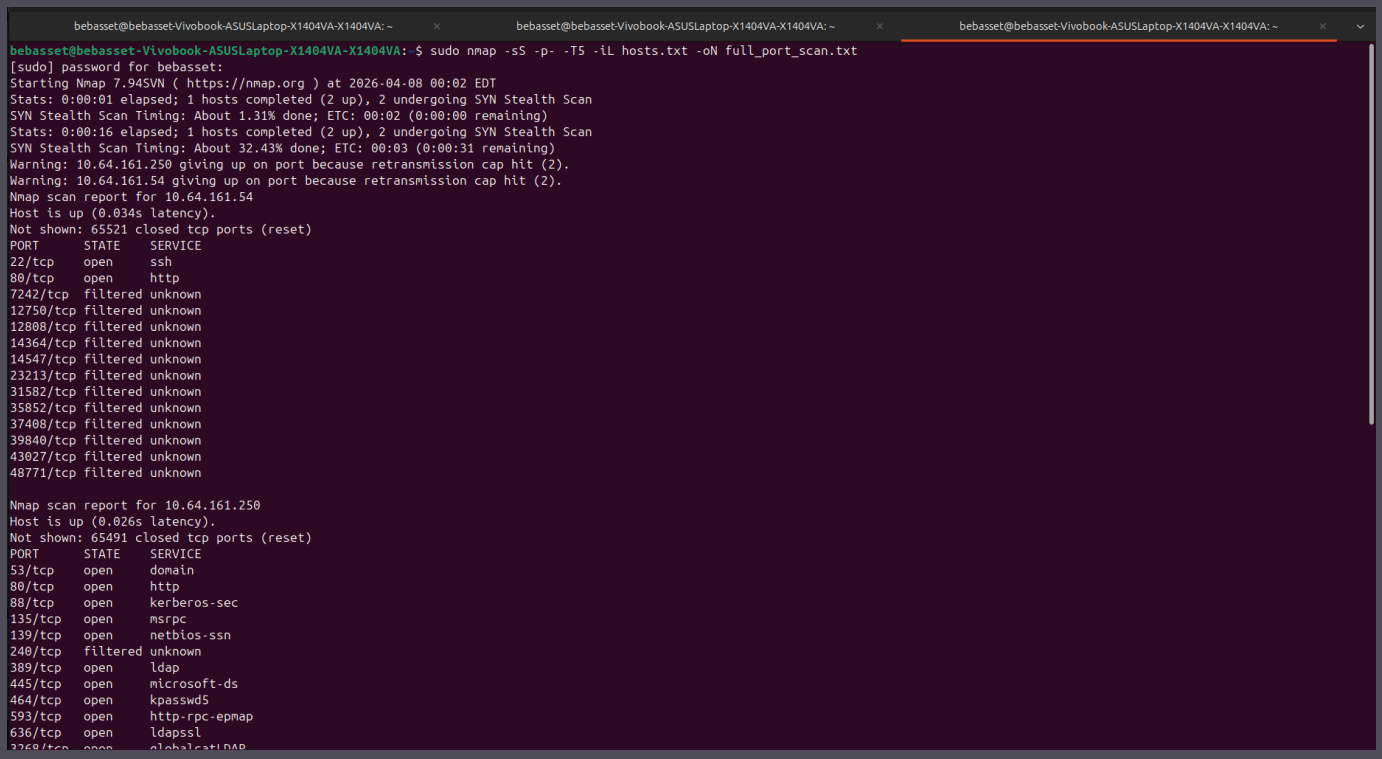


This screenshot shows the initial host discovery scan used to identify active systems on the lab network. The scan revealed three responsive IP addresses, which were recorded for further investigation. This allowed me to narrow my focus to the systems that were actually online.

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

Figure 3. Full TCP Port Scan Against Discovered Hosts:

Here we perform a full TCP SYN scan against the discovered hosts to identify exposed services.

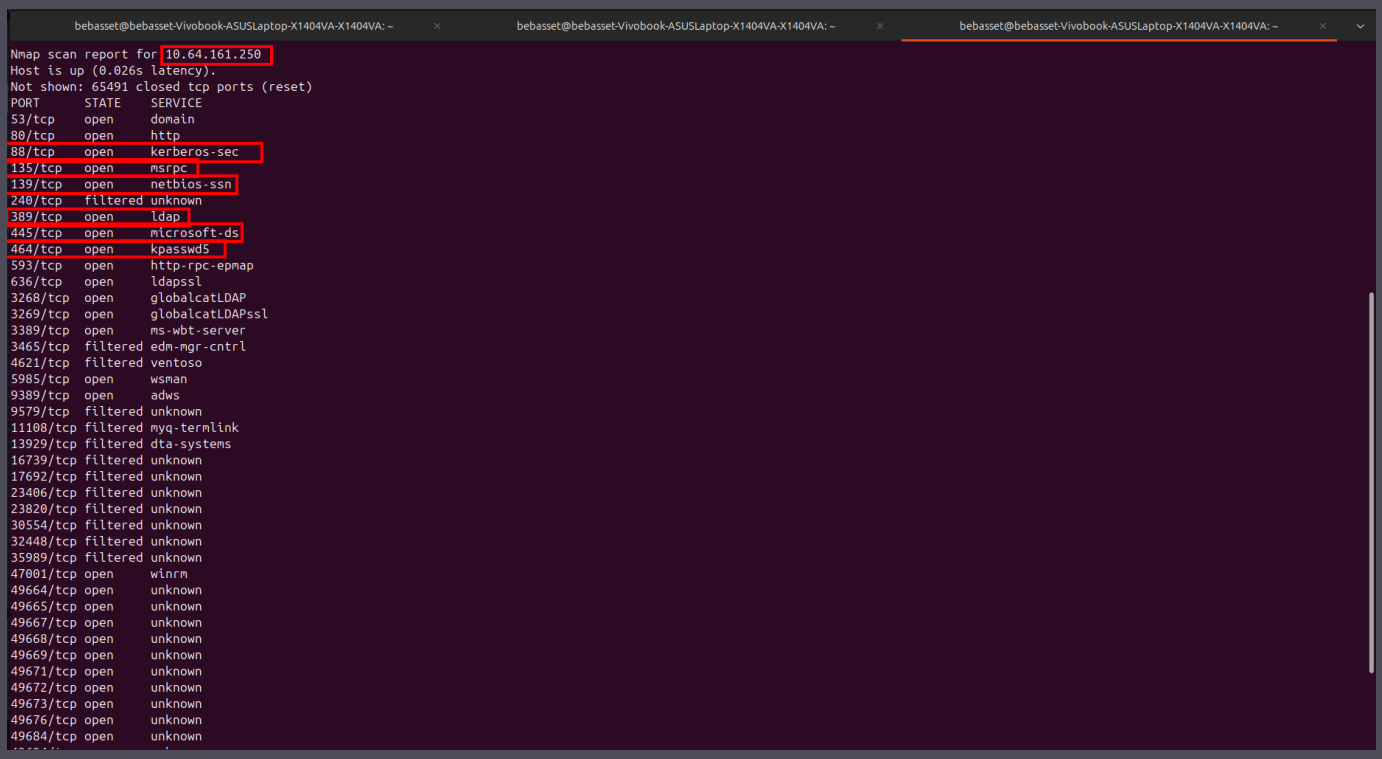


This screenshot captures the full TCP port scan conducted against the live hosts listed in hosts.txt. By scanning all ports, I was able to compare exposed services across systems and begin determining which host was likely providing Active Directory services.

```
sudo nmap -sS -p- -T5 -iL hosts.txt -oN full_port_scan.txt
```

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

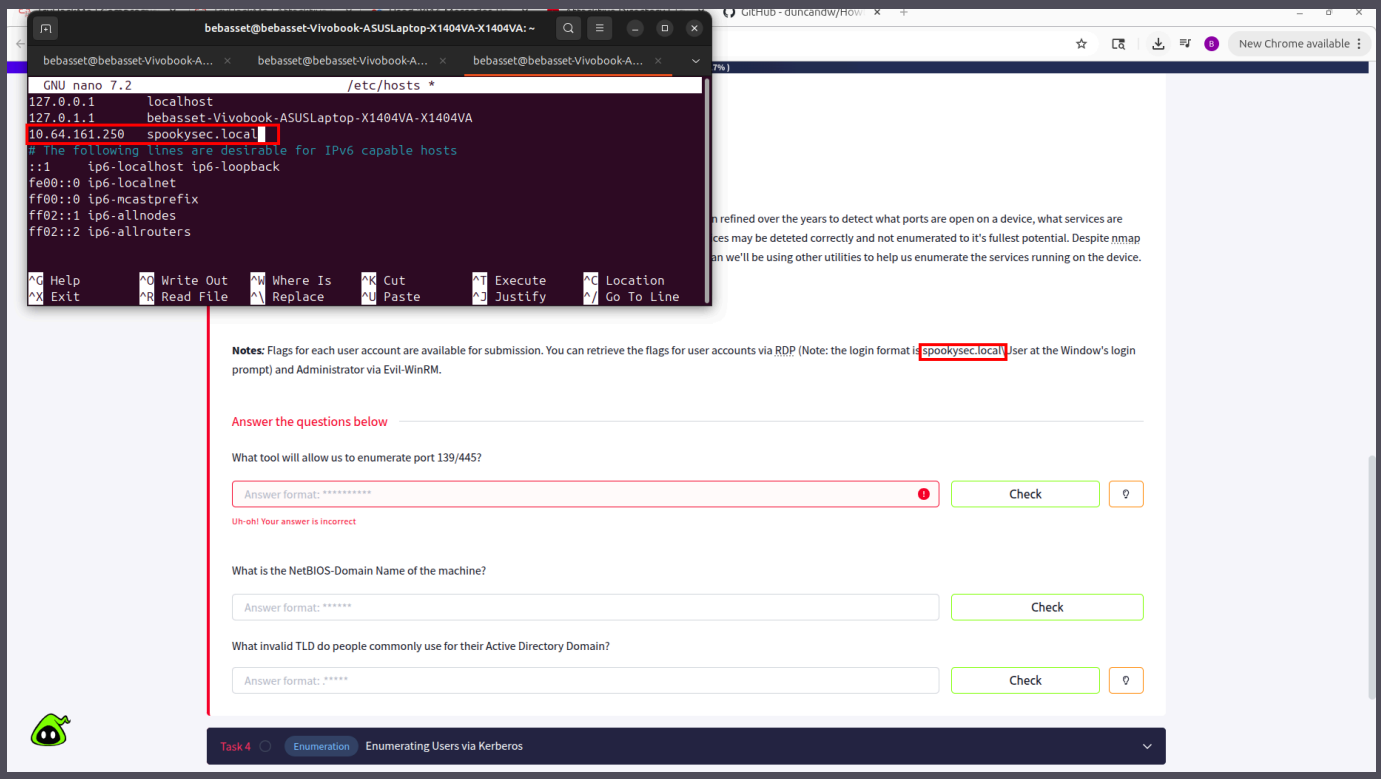
Our purpose here is to confirm which host was acting as the domain controller based on the services exposed.



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.



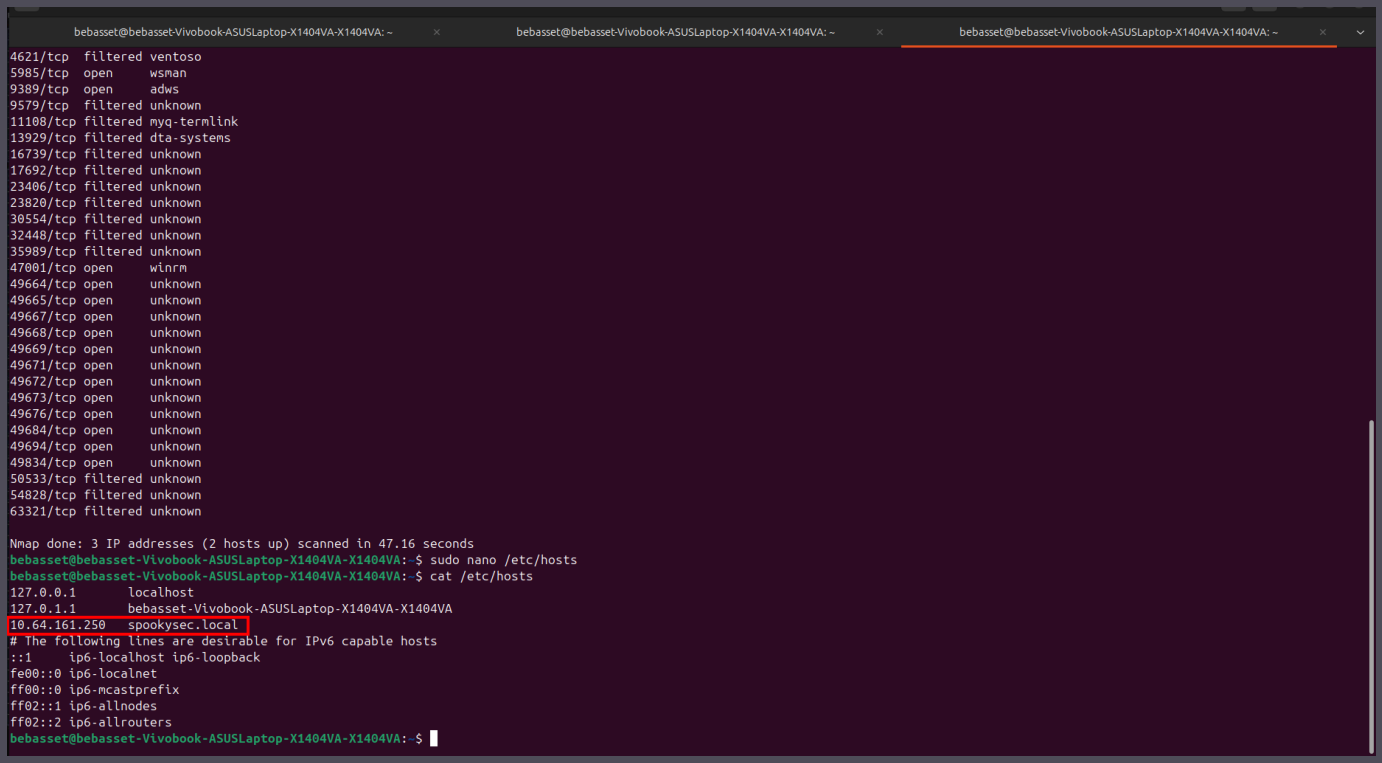
This screenshot shows the local `/etc/hosts` file being updated to map **`10.64.161.250`** to **`spookysec.local`**. Adding this entry made it easier to reference the target by name during domain-based enumeration activities.

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.



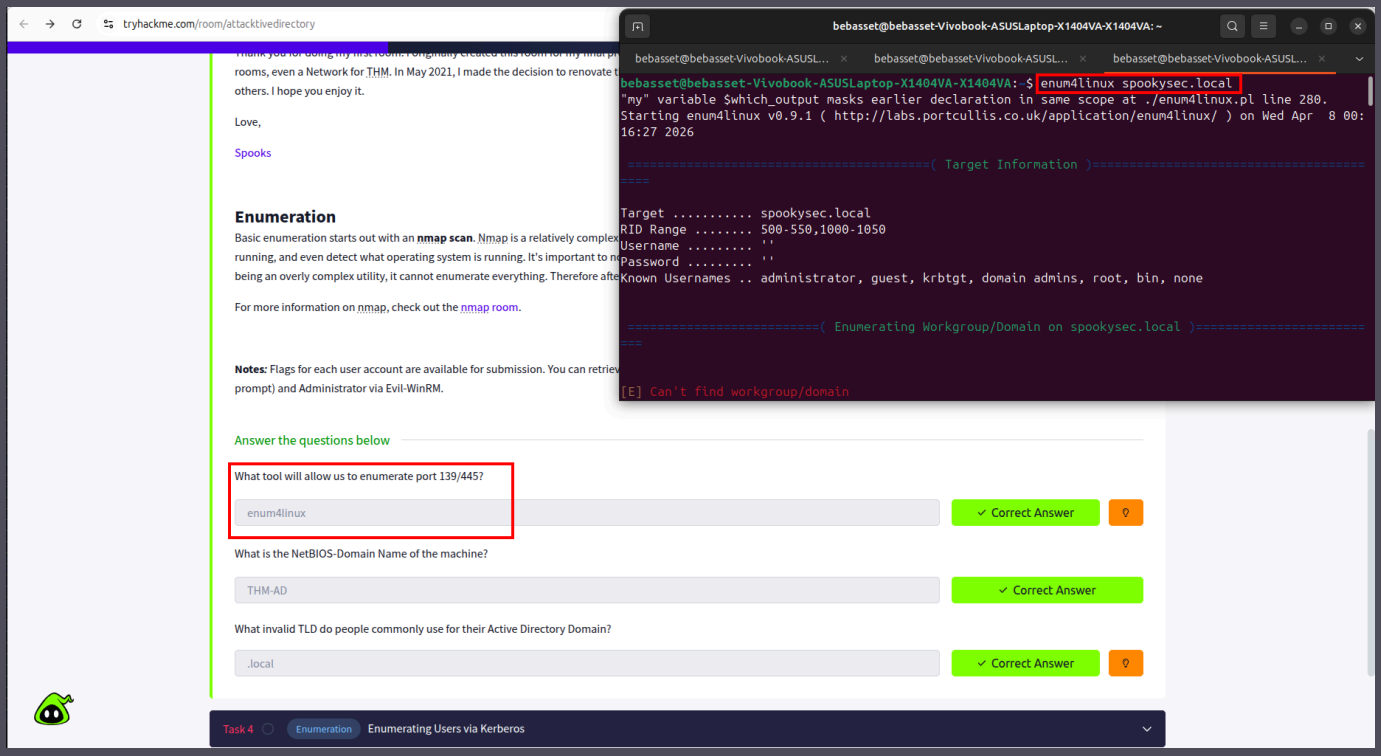
This screenshot shows the successful verification of the `/etc/hosts` update. The entry mapping **`10.64.161.250`** to **`spookysec.local`** was confirmed in the terminal, ensuring that later tools could properly resolve the target domain name.

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.



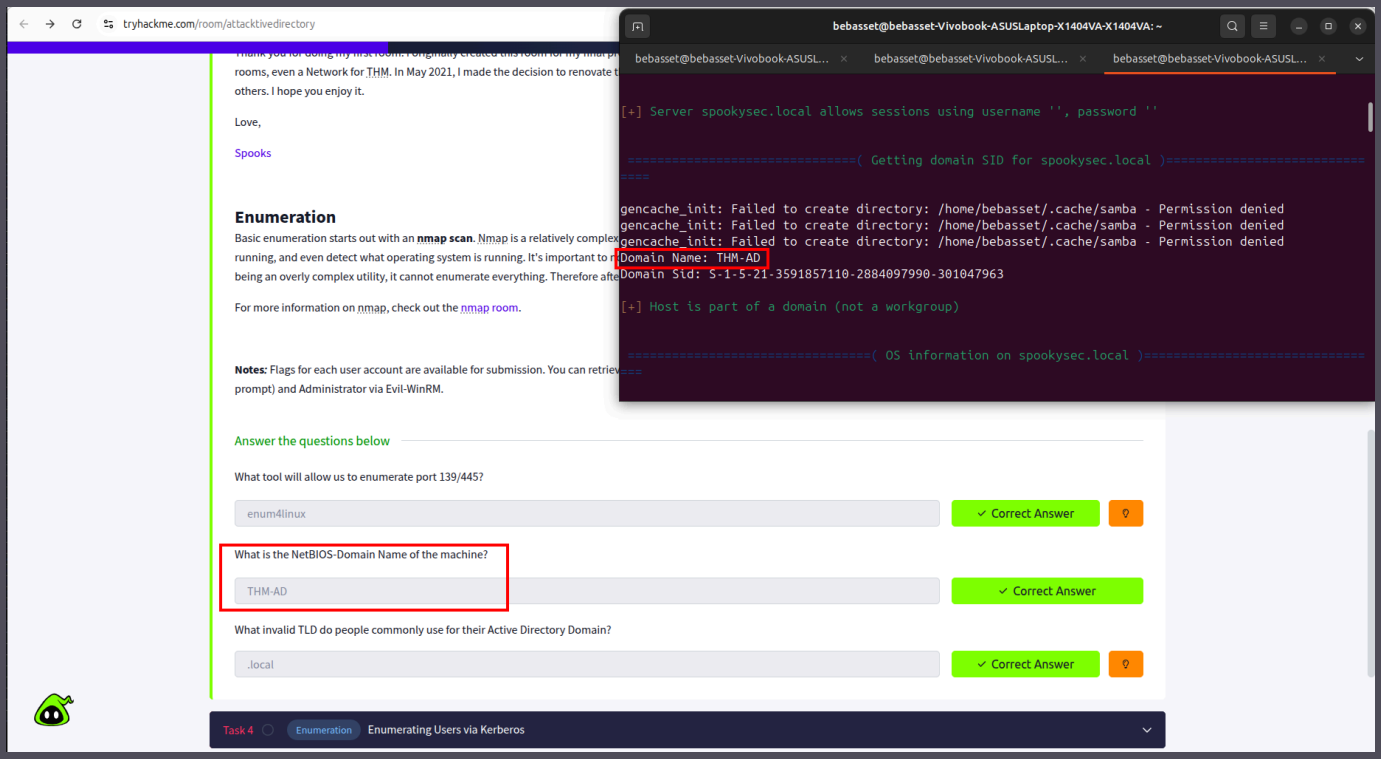
This screenshot captures the use of enum4linux against spookysec.local. This step was used to interrogate SMB and NetBIOS-related services in order to identify useful domain information, including naming conventions and potential authentication details.

Command Used:

```
enum4linux spookysec.local
```

Figure 8. Discovering the NetBIOS Domain Name:

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



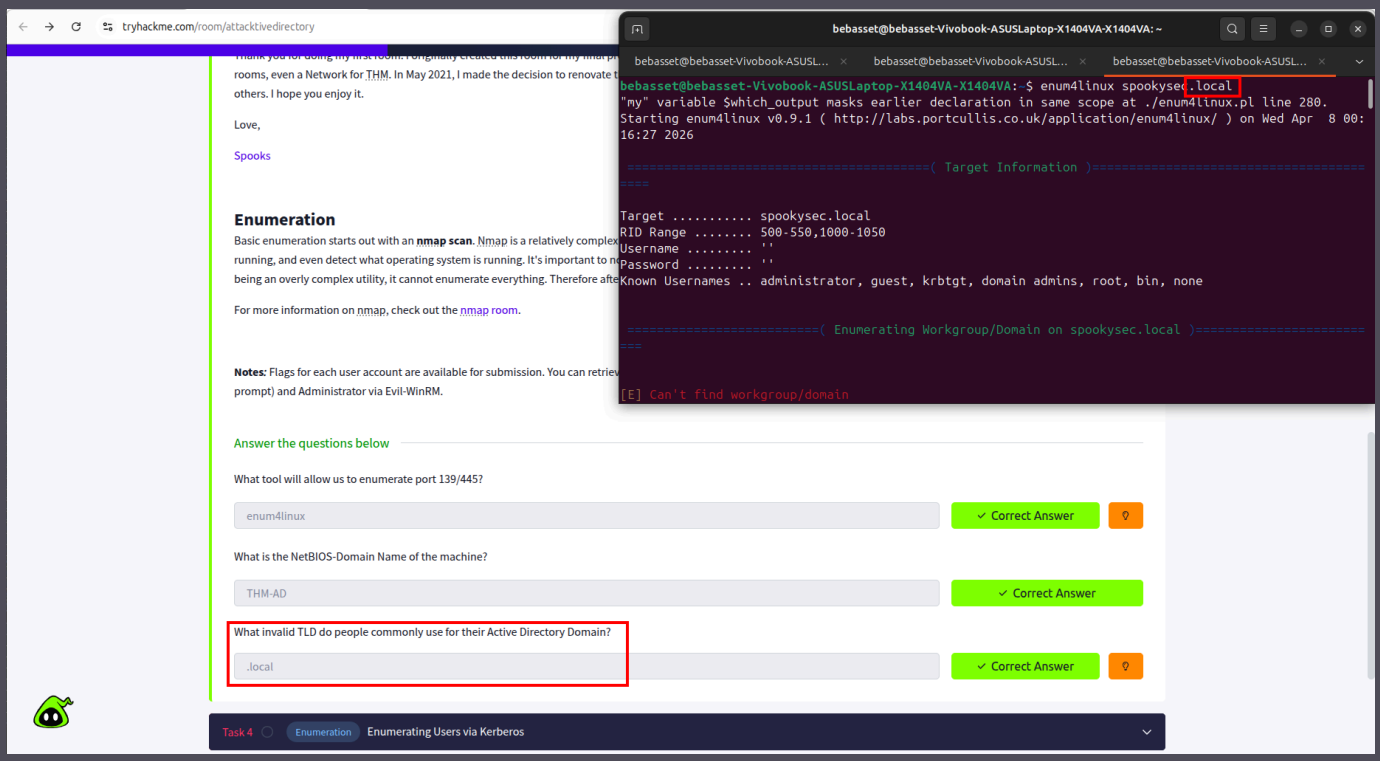
This screenshot highlights the discovery of the NetBIOS domain name: **THM-AD**. Identifying the NetBIOS name was important because it confirmed how the domain was represented internally and provided the correct answer for the room question.

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.



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:

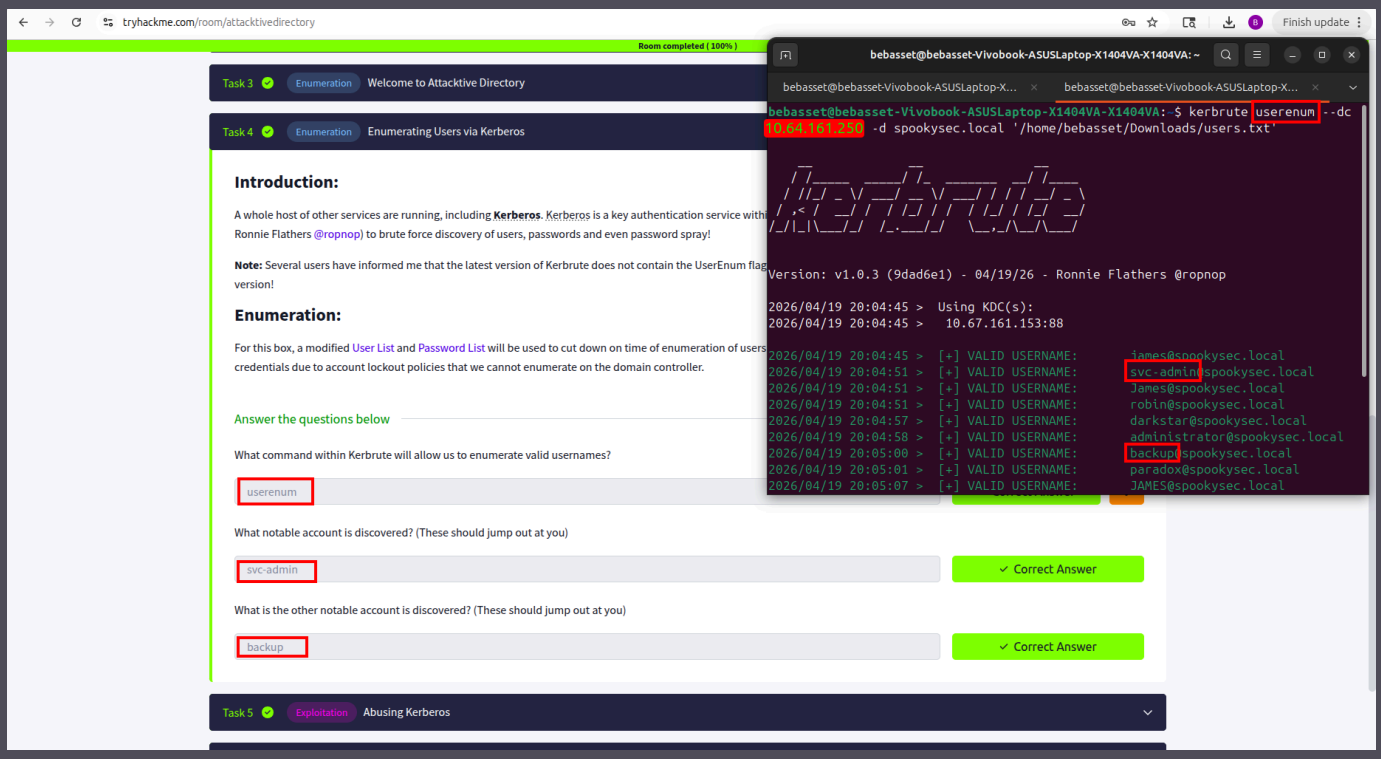
During the enumeration phase, I began by reviewing the room guidance and performing host discovery to identify active systems on the target subnet. After identifying live hosts, I conducted a full TCP SYN scan to enumerate exposed services and compare the attack surface of each system. Based on the results, 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.



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:

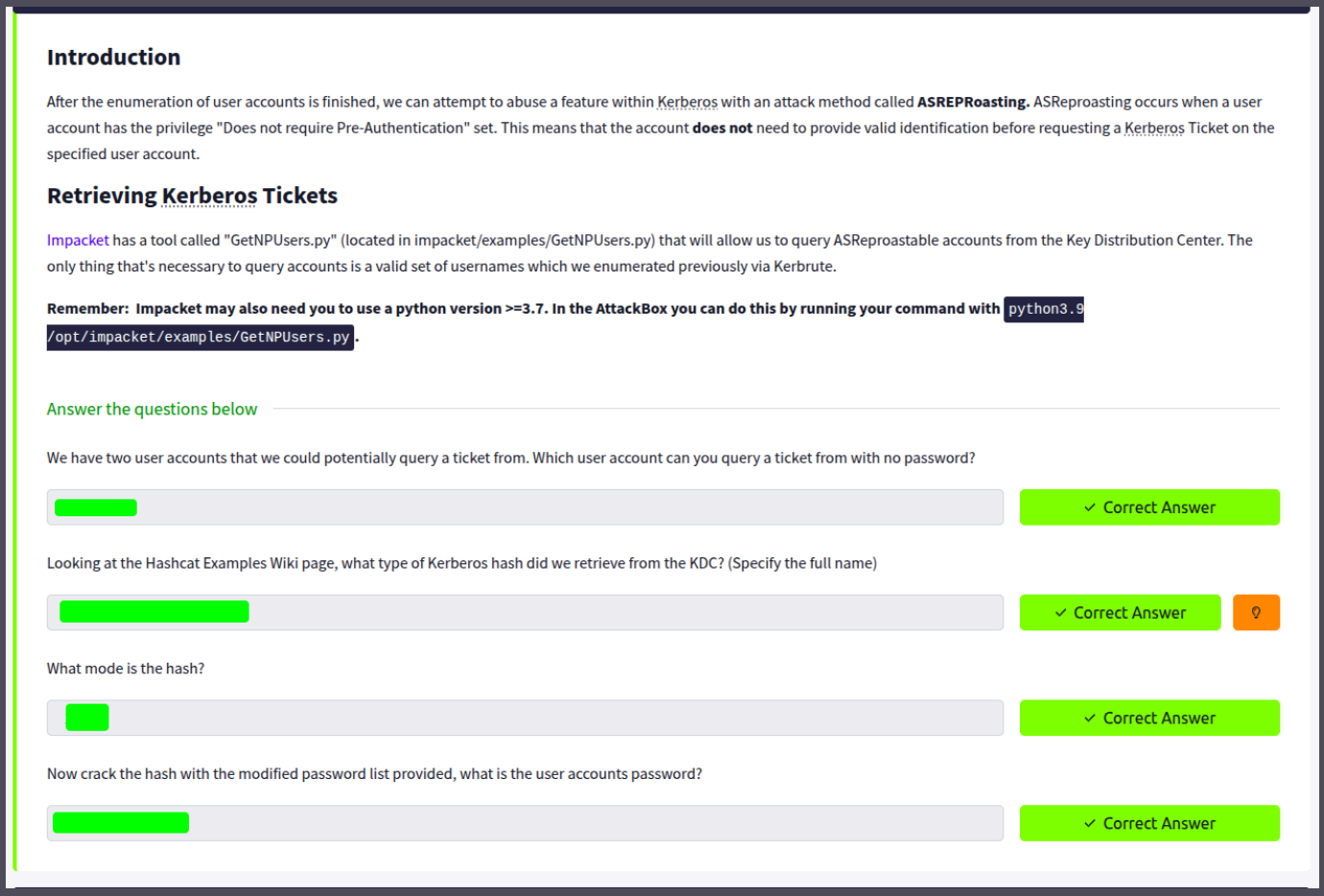
In the second phase of enumeration, I shifted focus to the Kerberos service running on the domain controller. Using Kerbrute in userenum mode, I tested a prepared username list against the spookysec.local domain to identify valid accounts. This approach allowed me to confirm legitimate usernames without 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

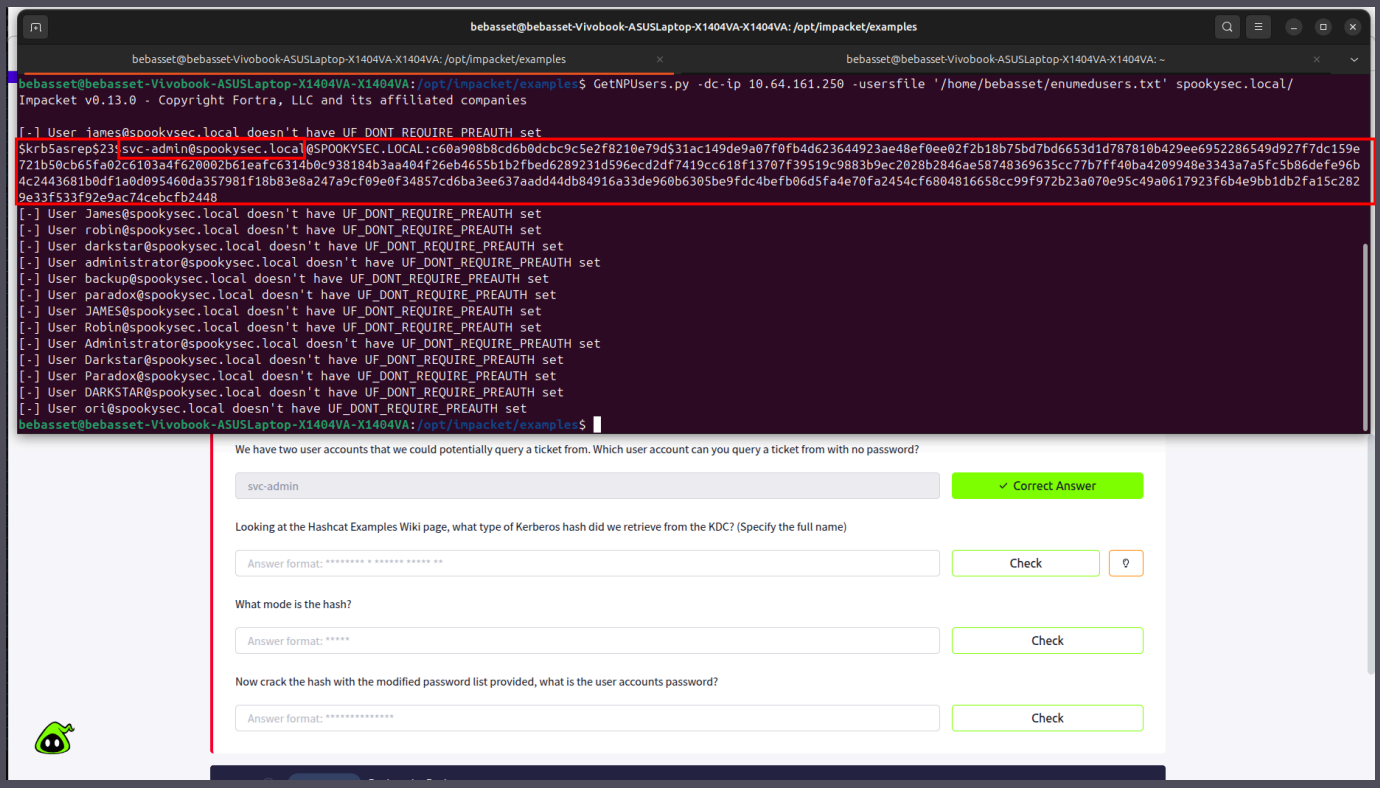
Now crack the hash with the modified password list provided, what is the user accounts password?

✓ 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.



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.

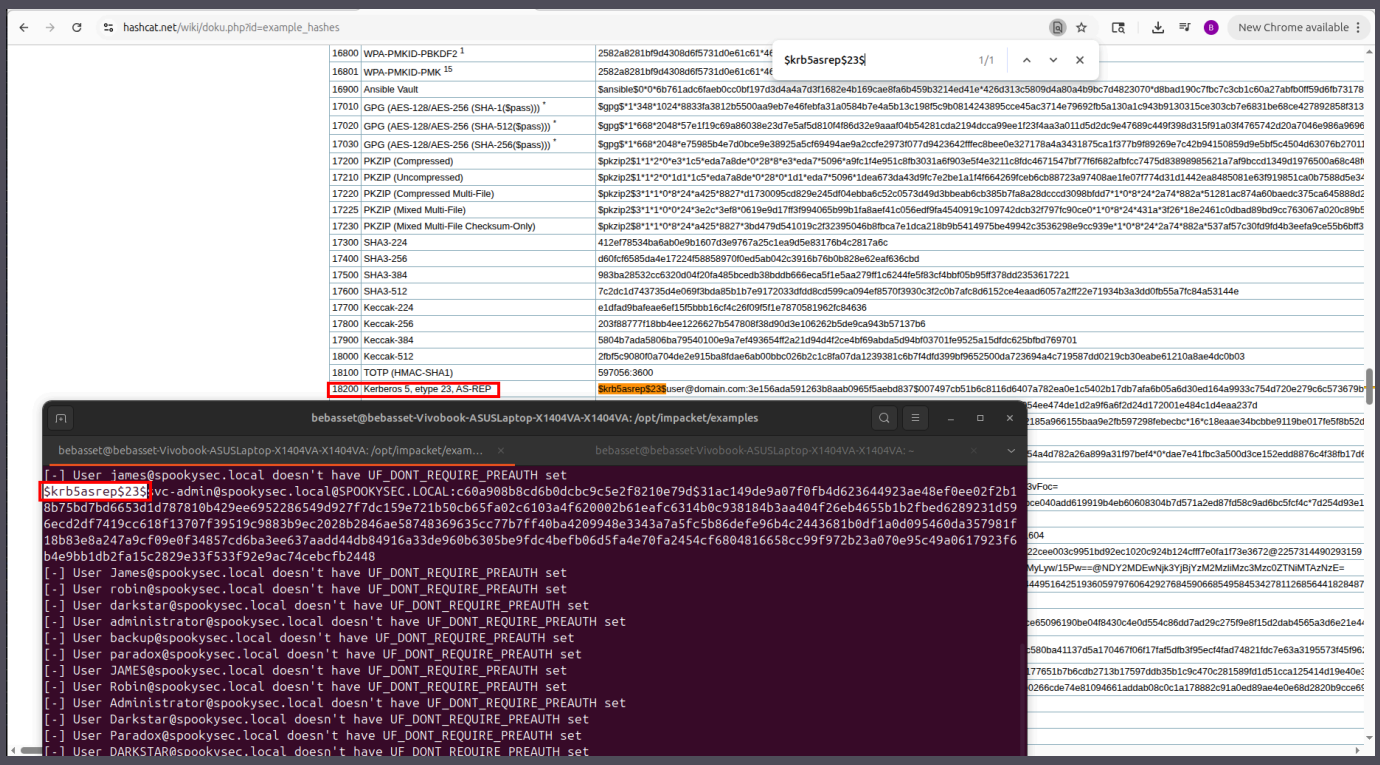
The highlighted output confirms that svc-admin was the account vulnerable to AS-REP Roasting.

Command Used:

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

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

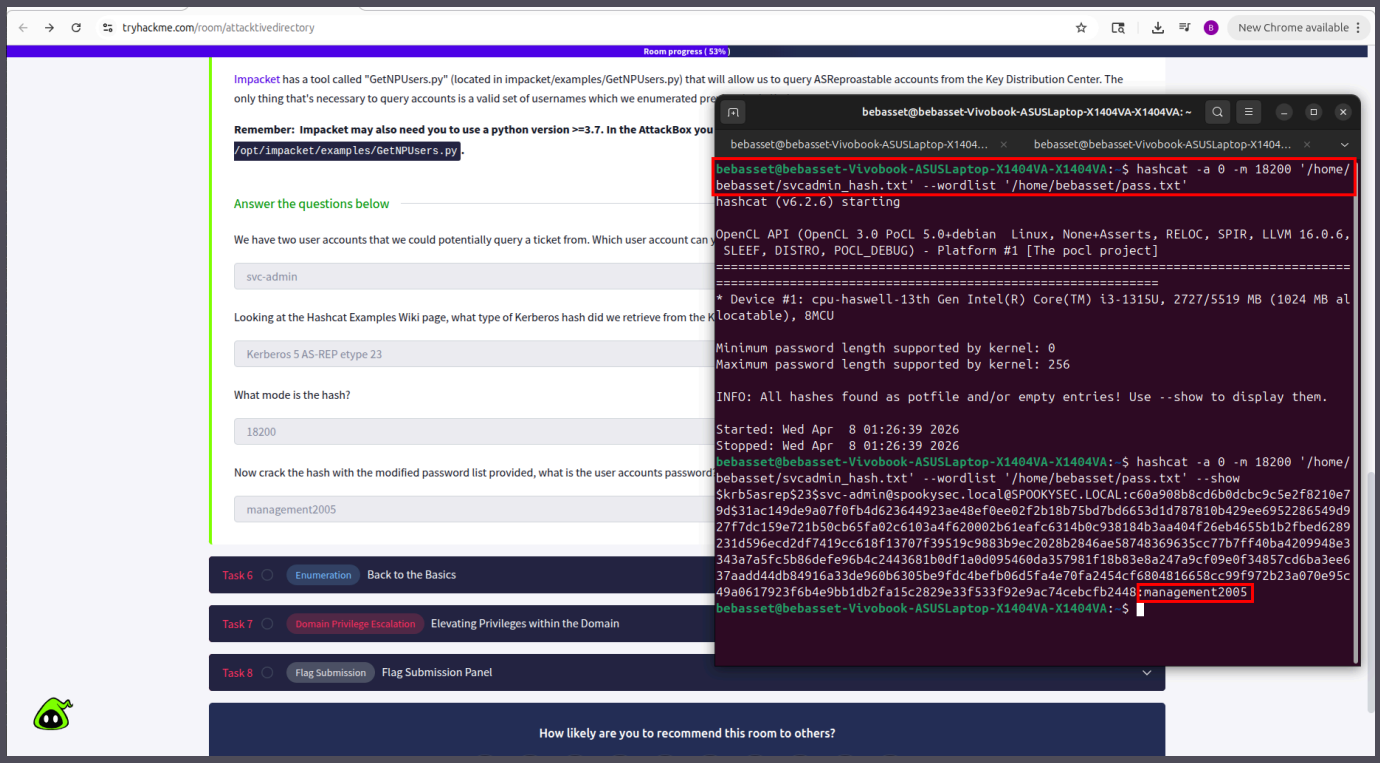
The purpose here is to match the recovered hash format to the correct Hashcat cracking mode before attempting password recovery.



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.



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.

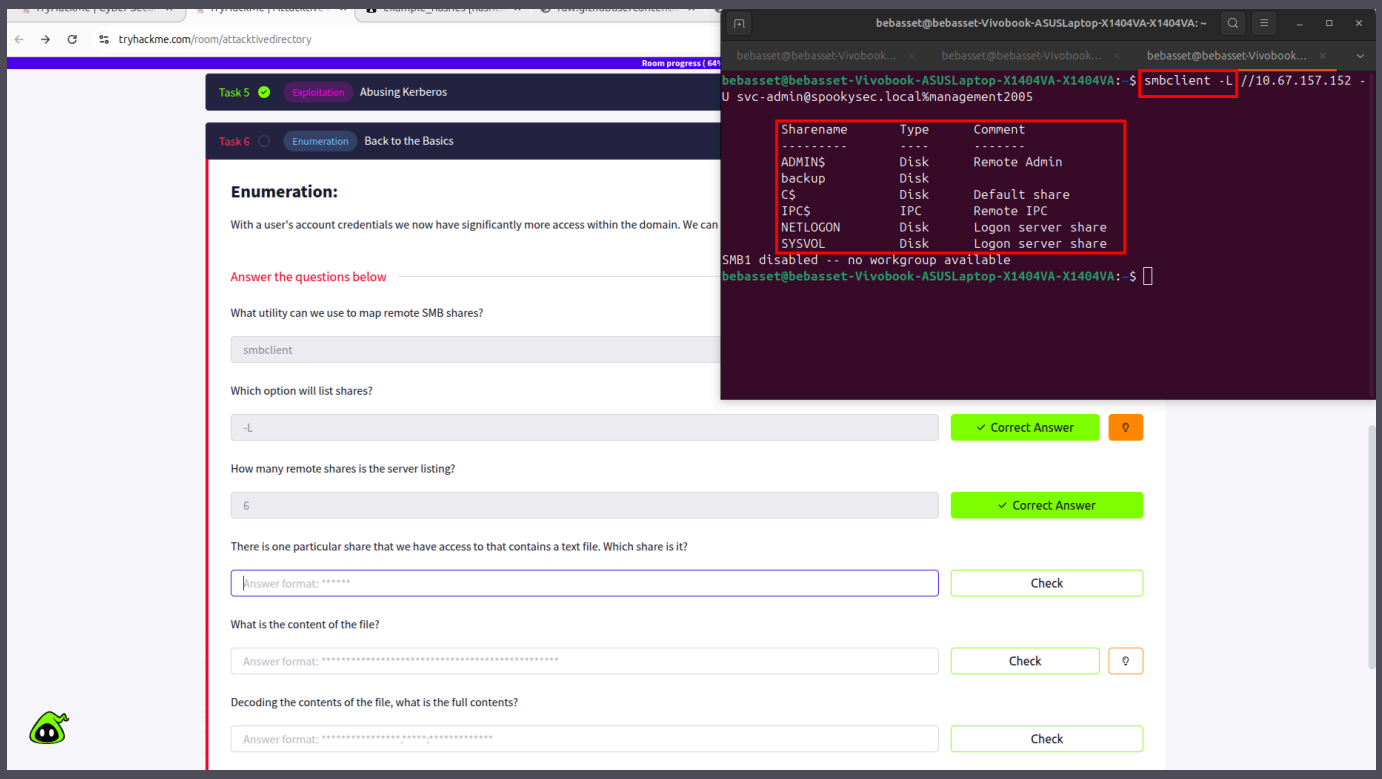
Result: Displayed the password management2005.

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:

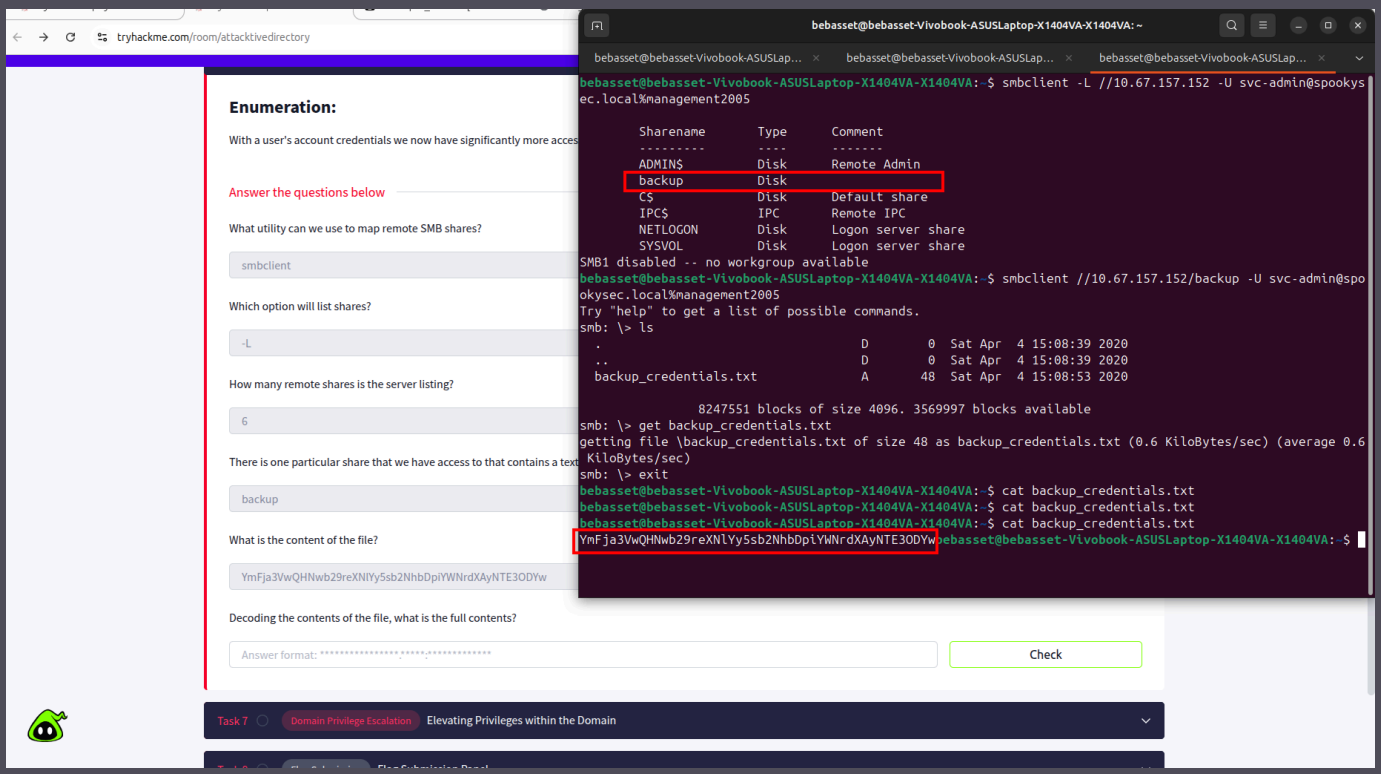
Our purpose here is to list the SMB shares available on the target using the previously recovered svc-admin credentials.



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:

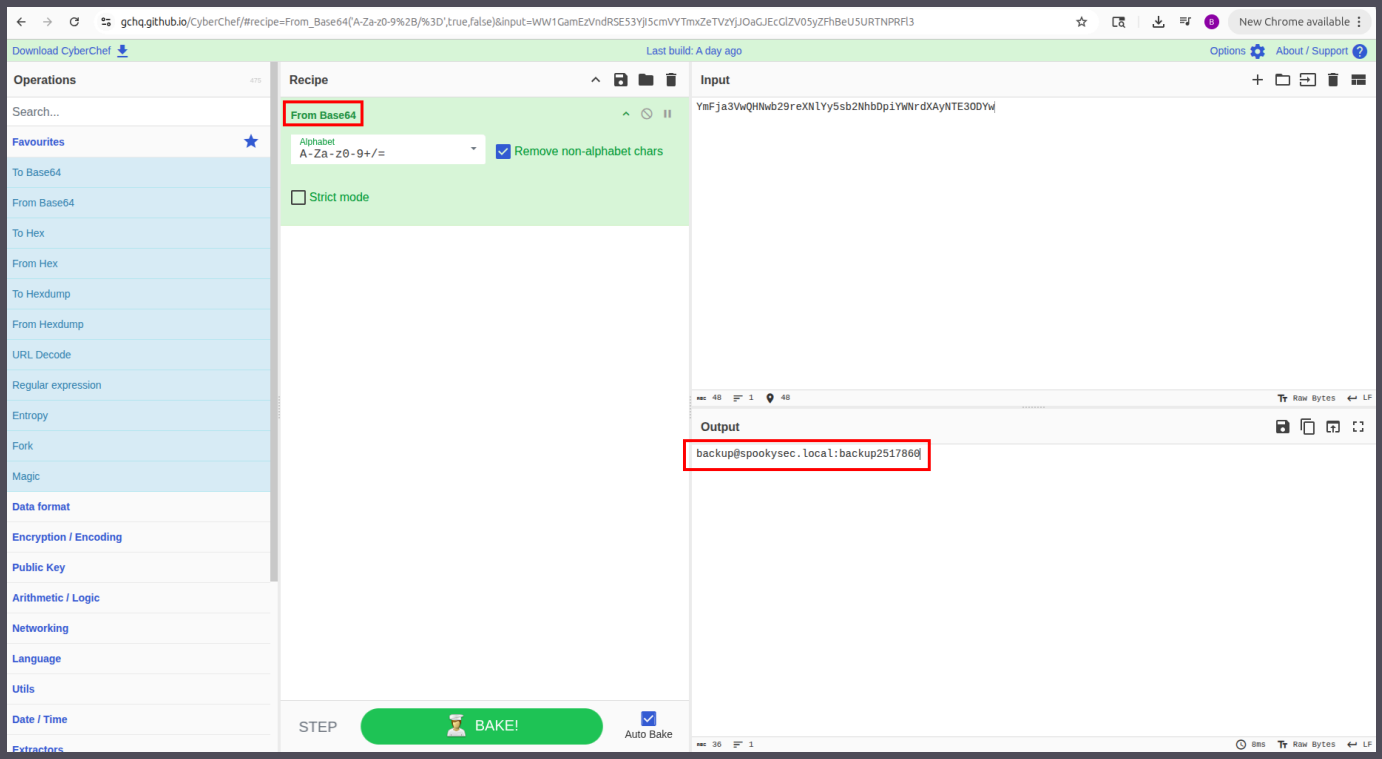
Access the backup share, identify files within it, download the available text file, and review its contents locally.



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 YmFja3VwQHNwb29reXNlYy5sb2NhbDpiYWNrdXAyNTE3ODYw, which appeared to be Base64-encoded data rather than plaintext credentials.

Figure 3. Decoding the Retrieved Base64 String in CyberChef:

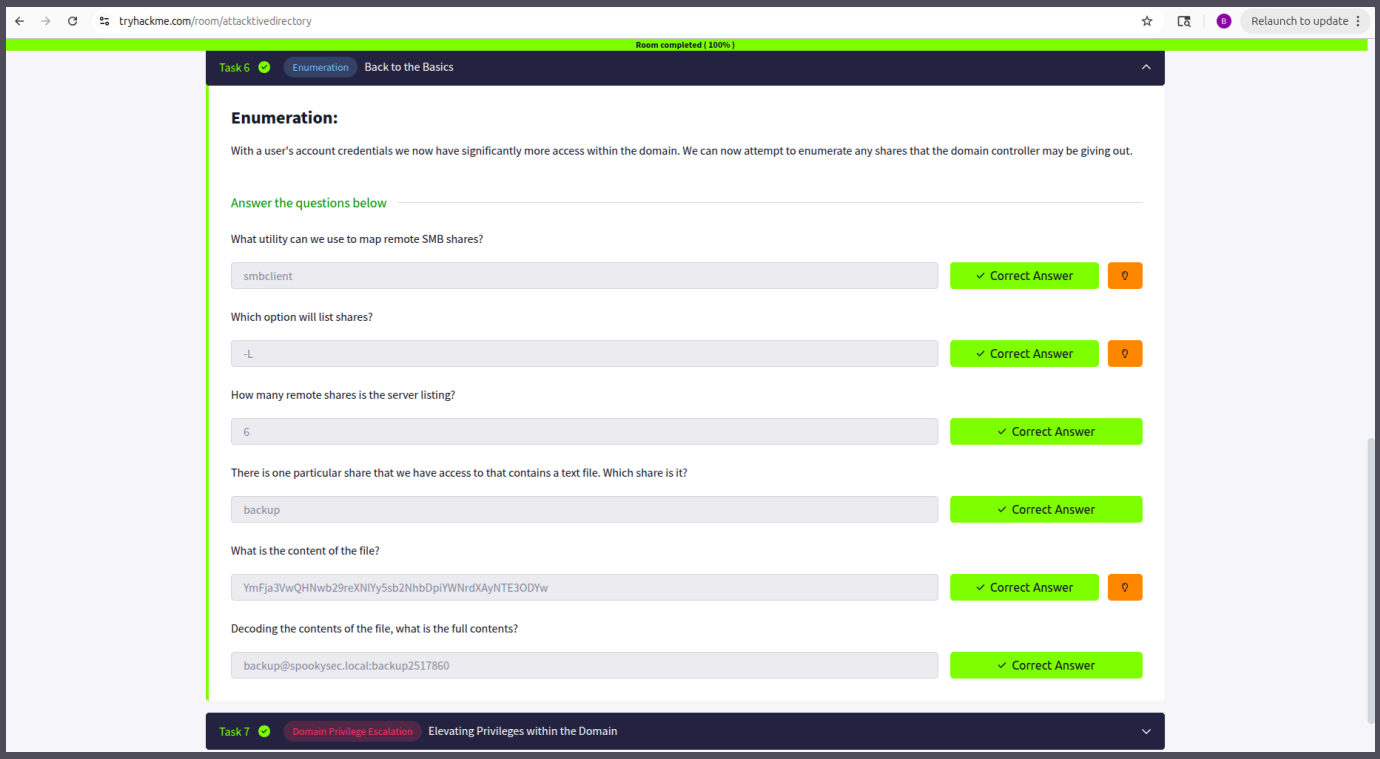
Here I decode the contents of backup_credentials.txt to reveal the underlying credential material.



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.



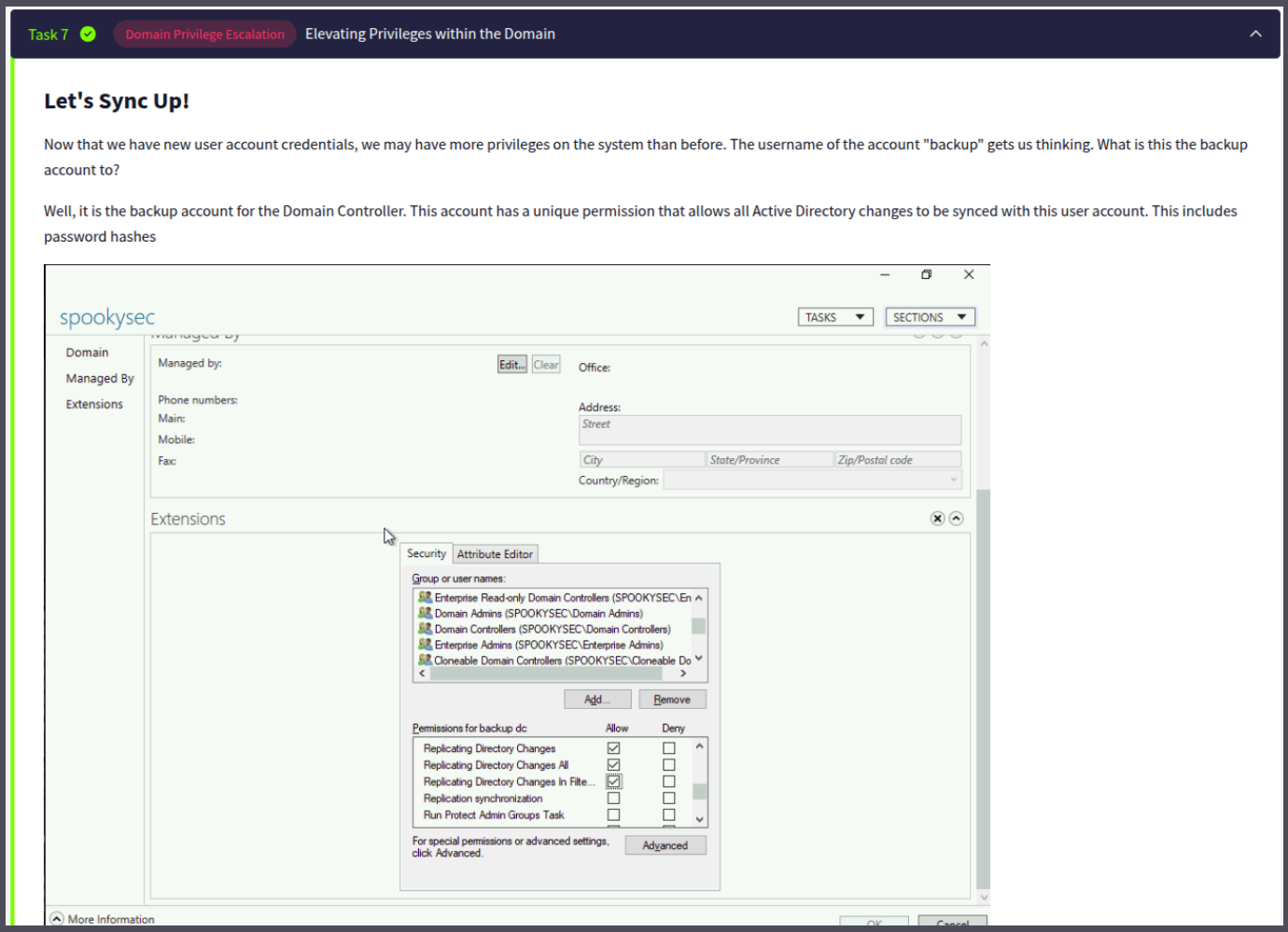
This screenshot shows the completed room answers for the "Back to the Basics" section. The findings confirmed that smbclient was the utility used to map remote SMB shares, the -L option listed shares, the server exposed six shares, the accessible share was backup, the retrieved file contained a Base64-encoded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

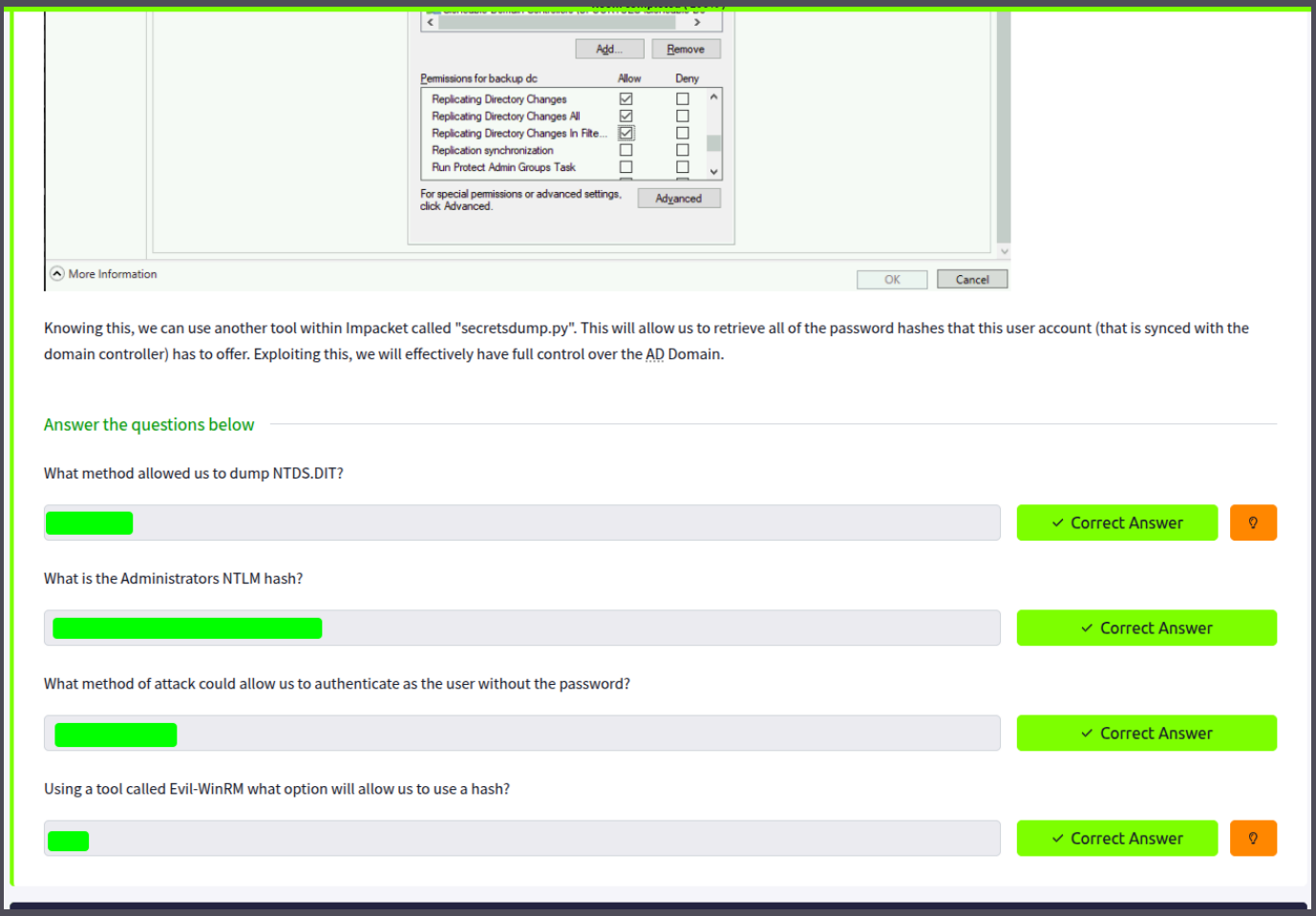
The purpose of this screenshot is to review why the backup account is high value and how its permissions can be abused for domain compromise.



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:

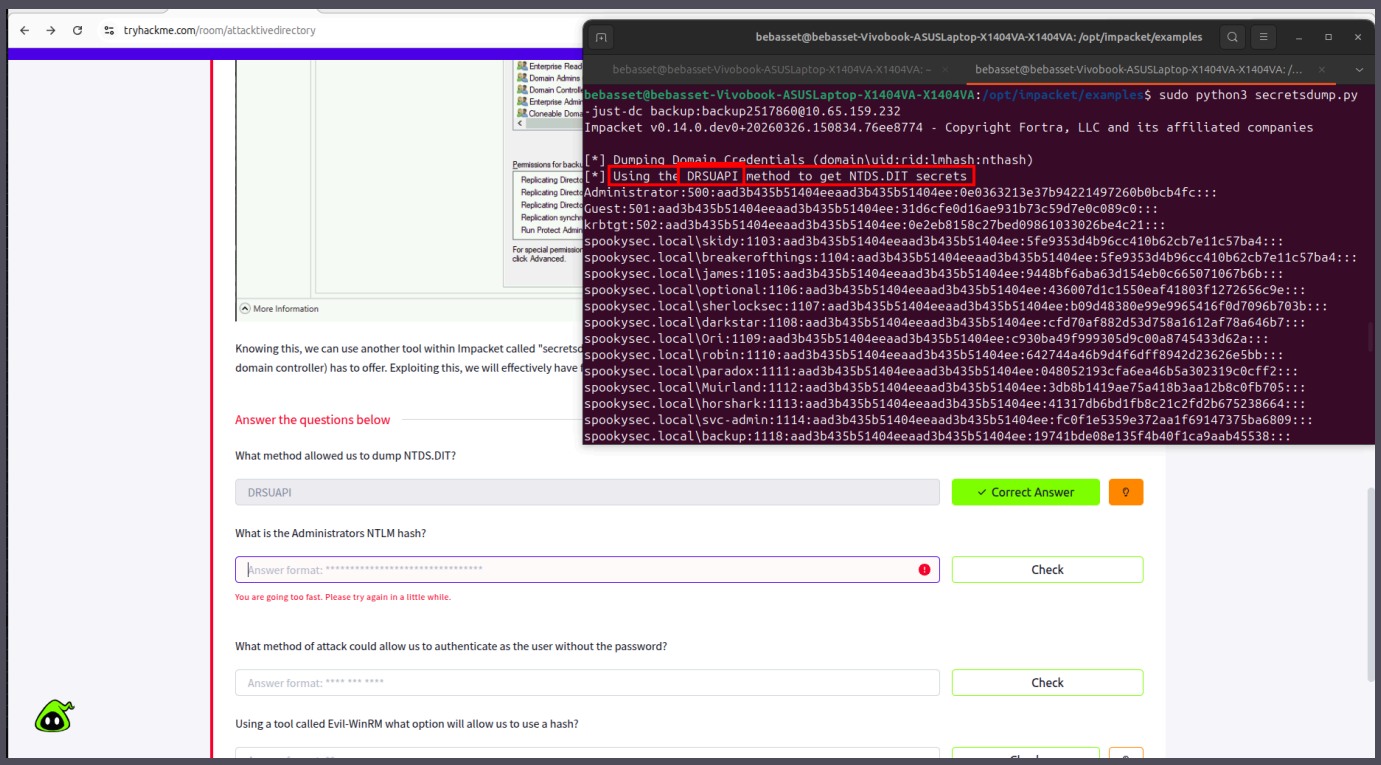
The purpose of this screenshot is to capture the main goals of the privilege escalation phase before executing the attack path.



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.



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.

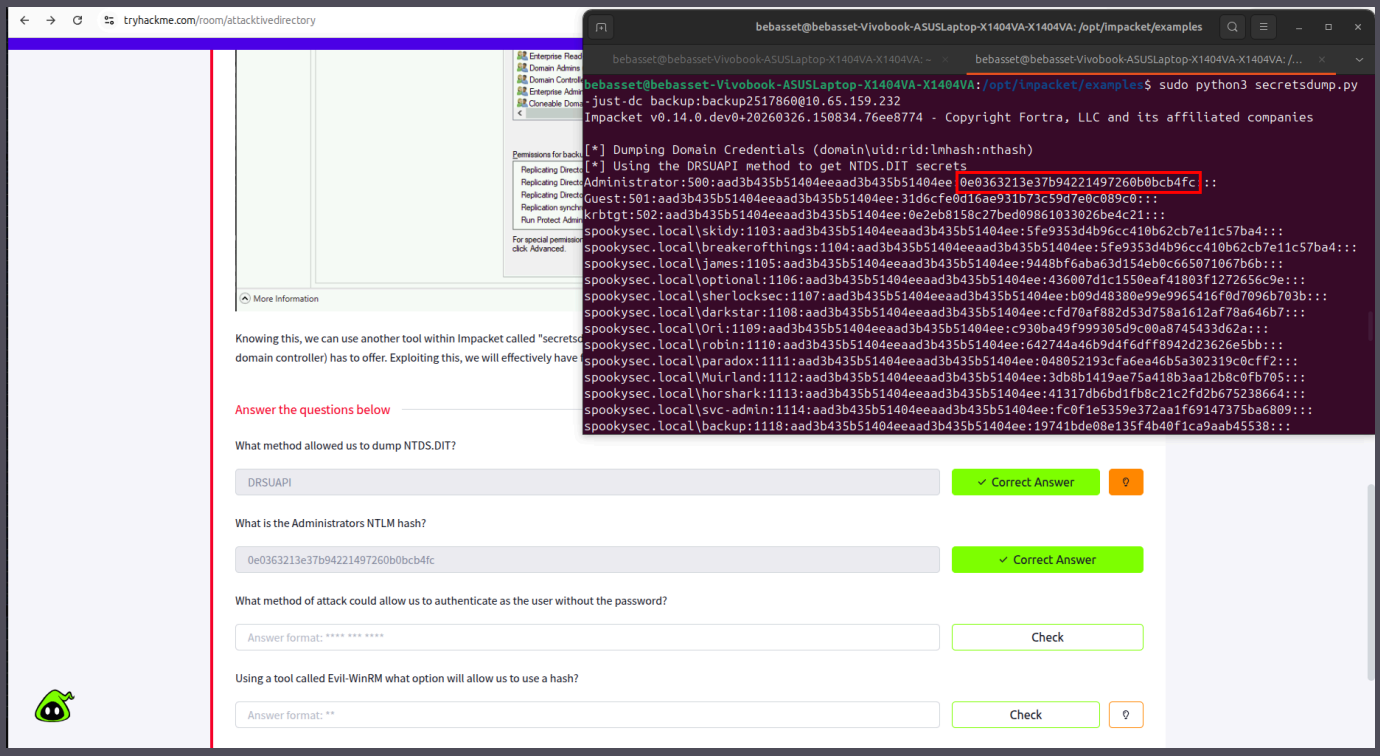
The highlighted line confirms the method used was DRSUAPI.

Command Used:

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

Figure 4. Extracting the Administrator NTLM Hash:

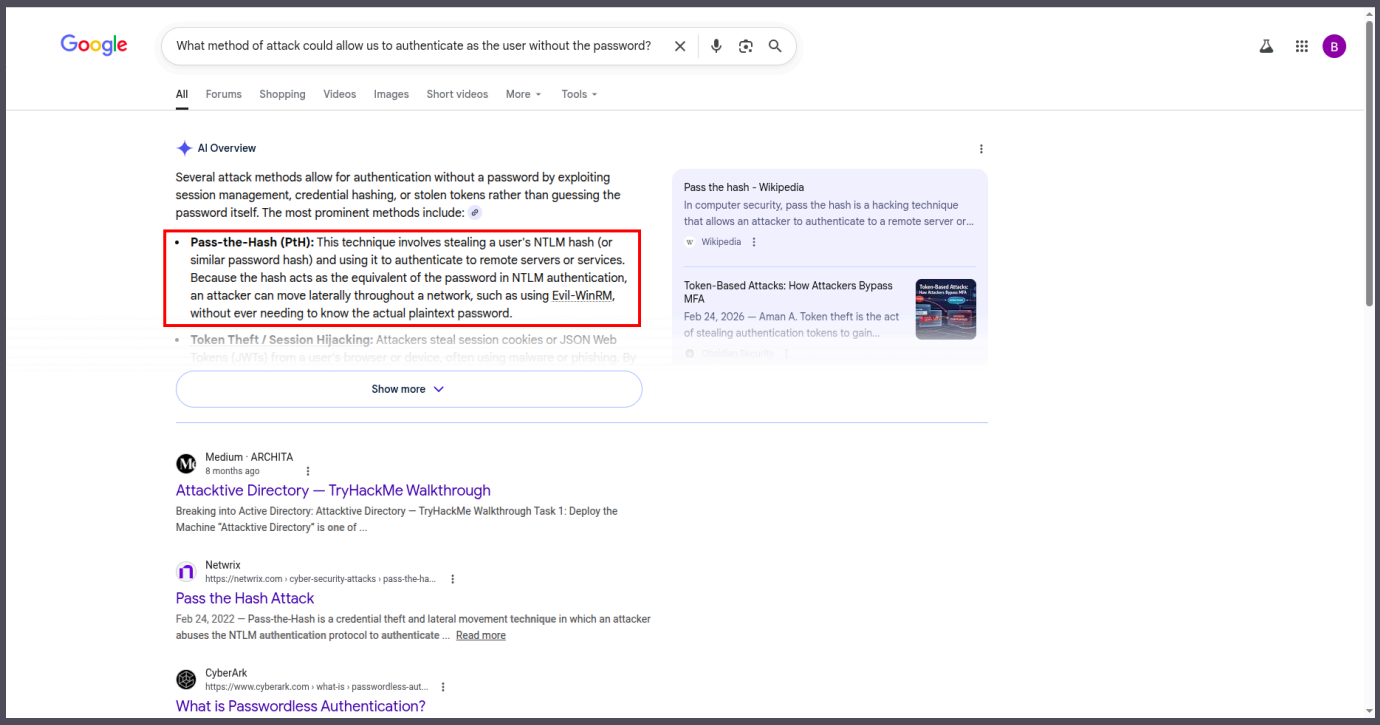
Here I Identify the Administrator account's NTLM hash from the dumped domain credential material.



This screenshot highlights the Administrator entry returned by secretsdump.py. The extracted NTLM hash shown for the Administrator account is: **0e0363213e37b94221497260b0bcb4fc**

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

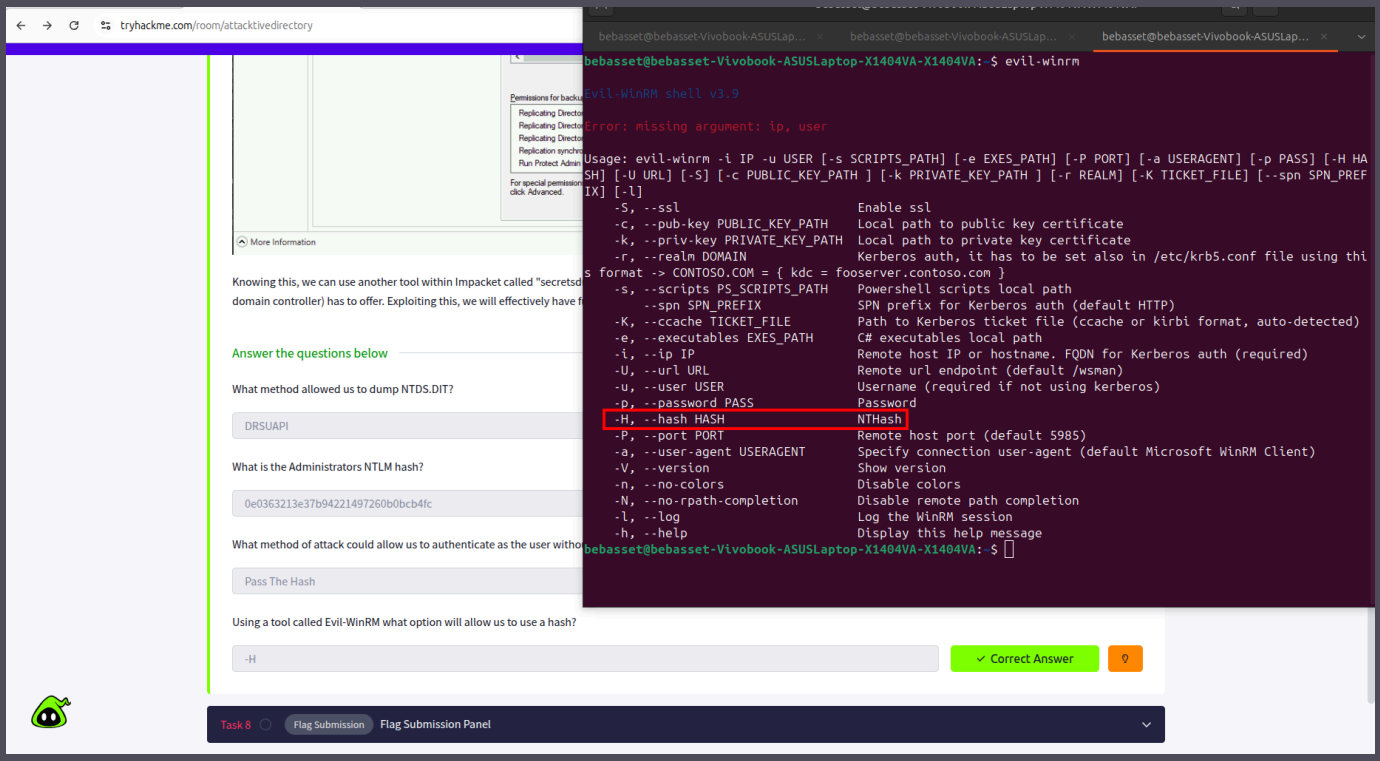
The purpose of this screenshot was to determine the method that allows authentication using an NTLM hash instead of the plaintext password.



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.



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:

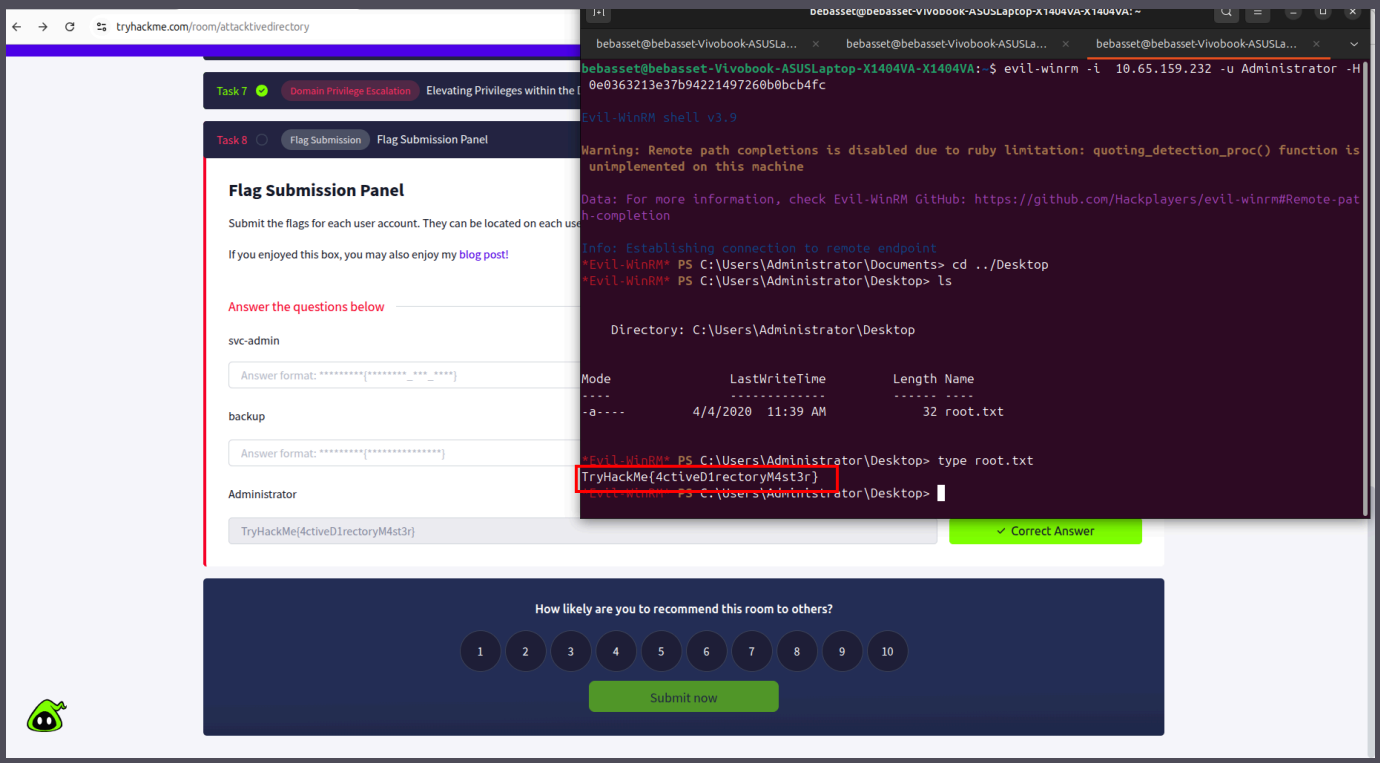
After obtaining the backup account credentials, I used them to abuse the account's replication privileges against the domain controller. With Impacket's secretsdump.py and the -just-dc option, I performed a DCSync-style credential dump and confirmed that the DRSUAPI method was used to retrieve NTDS.DIT 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.



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:

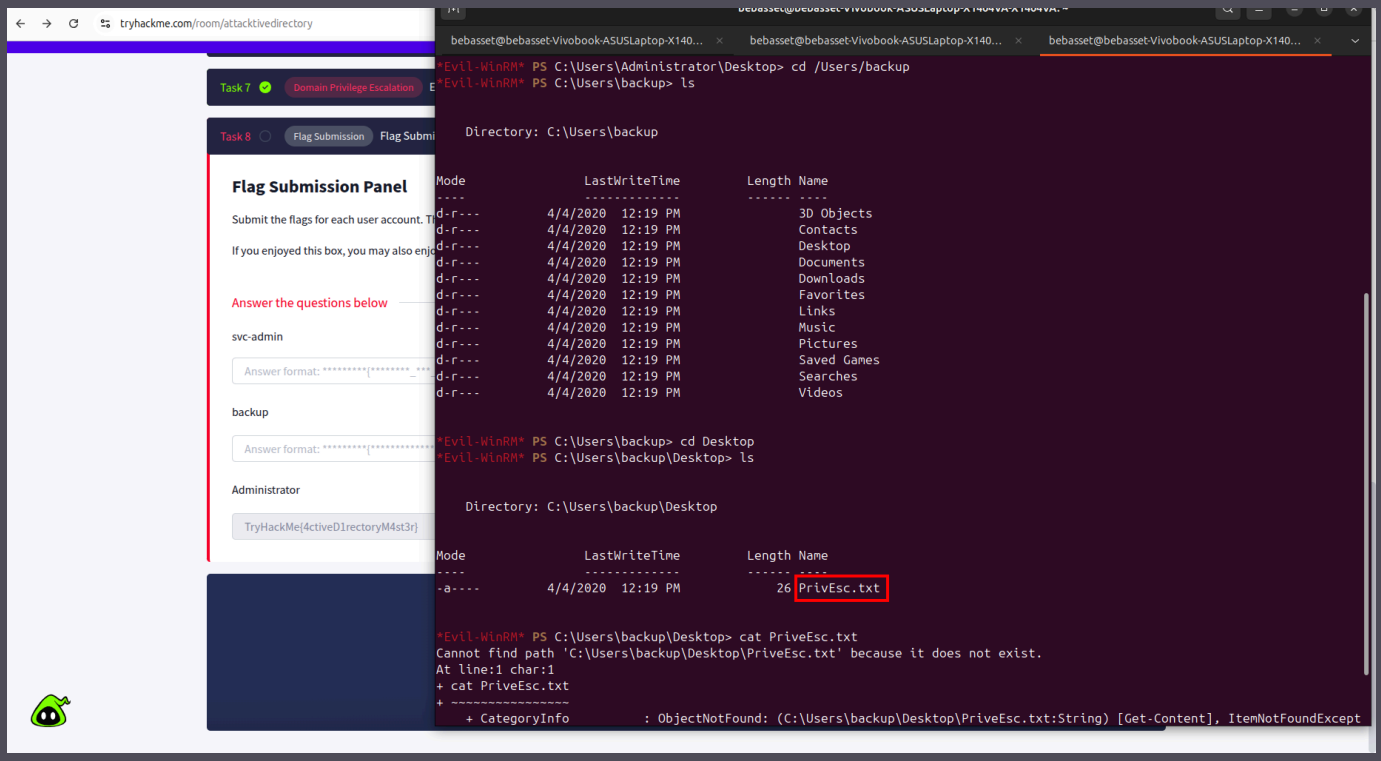
```
evil-winrm -i 10.64.161.250 -u Administrator -H 0e0363213e37b94221497260b0bcb4fc
```

Follow-up Commands:

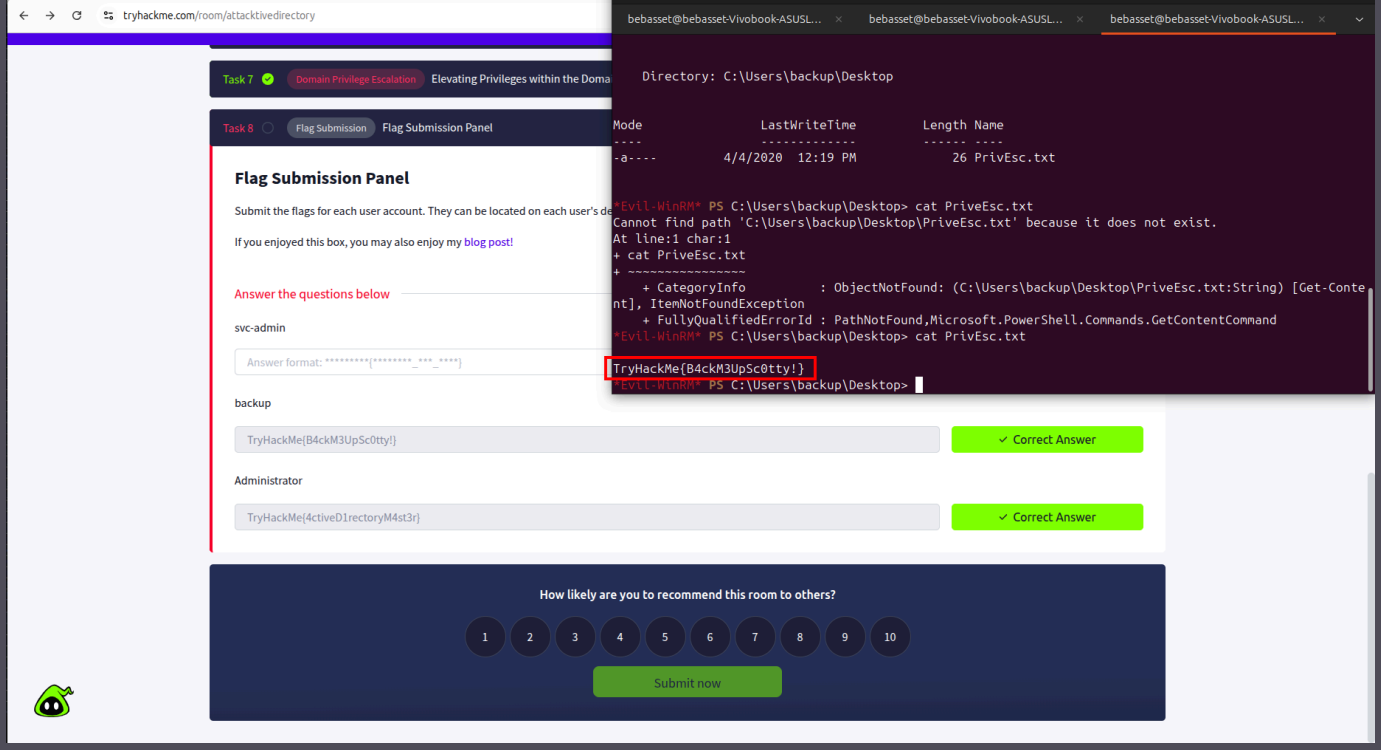
```
cd ../Desktop
ls
type root.txt
```

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

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



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.



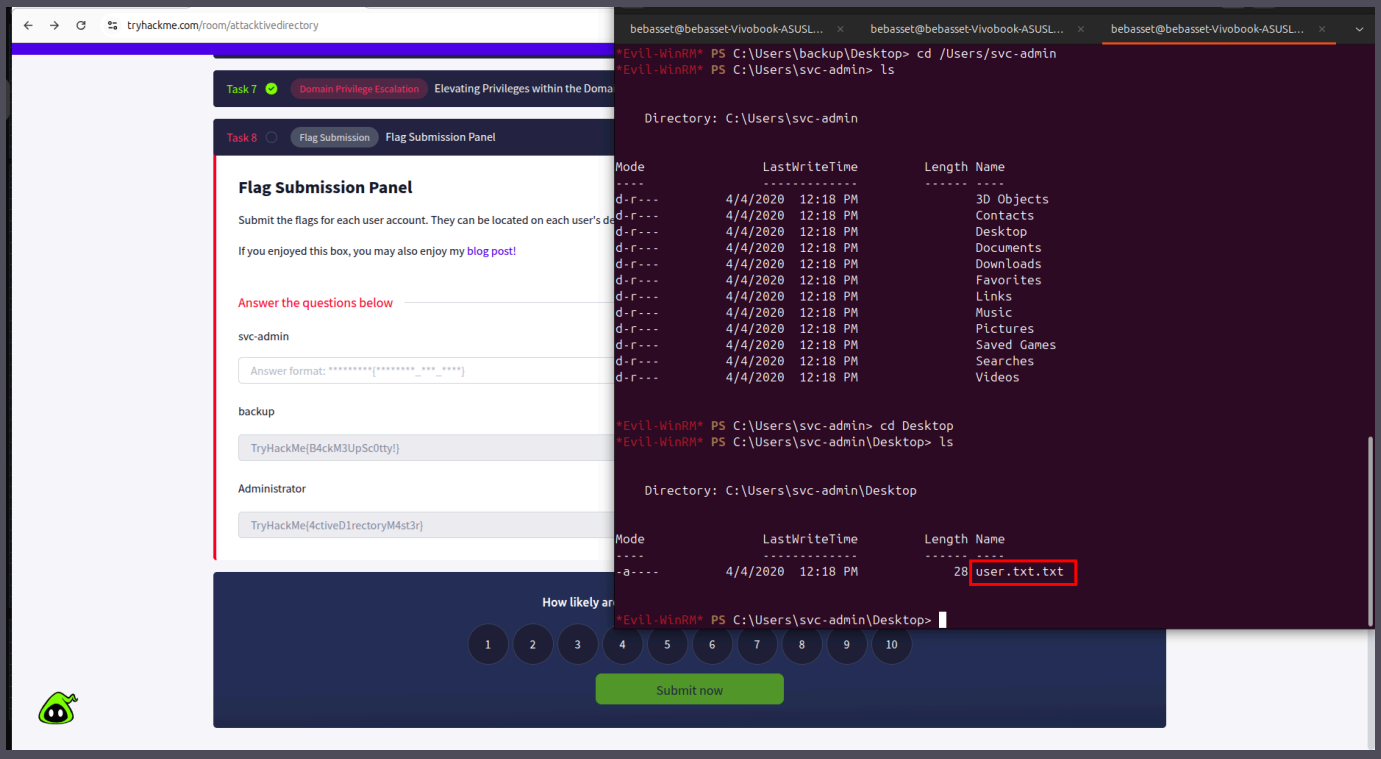
This screenshot shows the successful retrieval of the backup user flag from PrivEsc.txt and the flag value displayed. This completed the flag collection for the backup account.

Command Used:

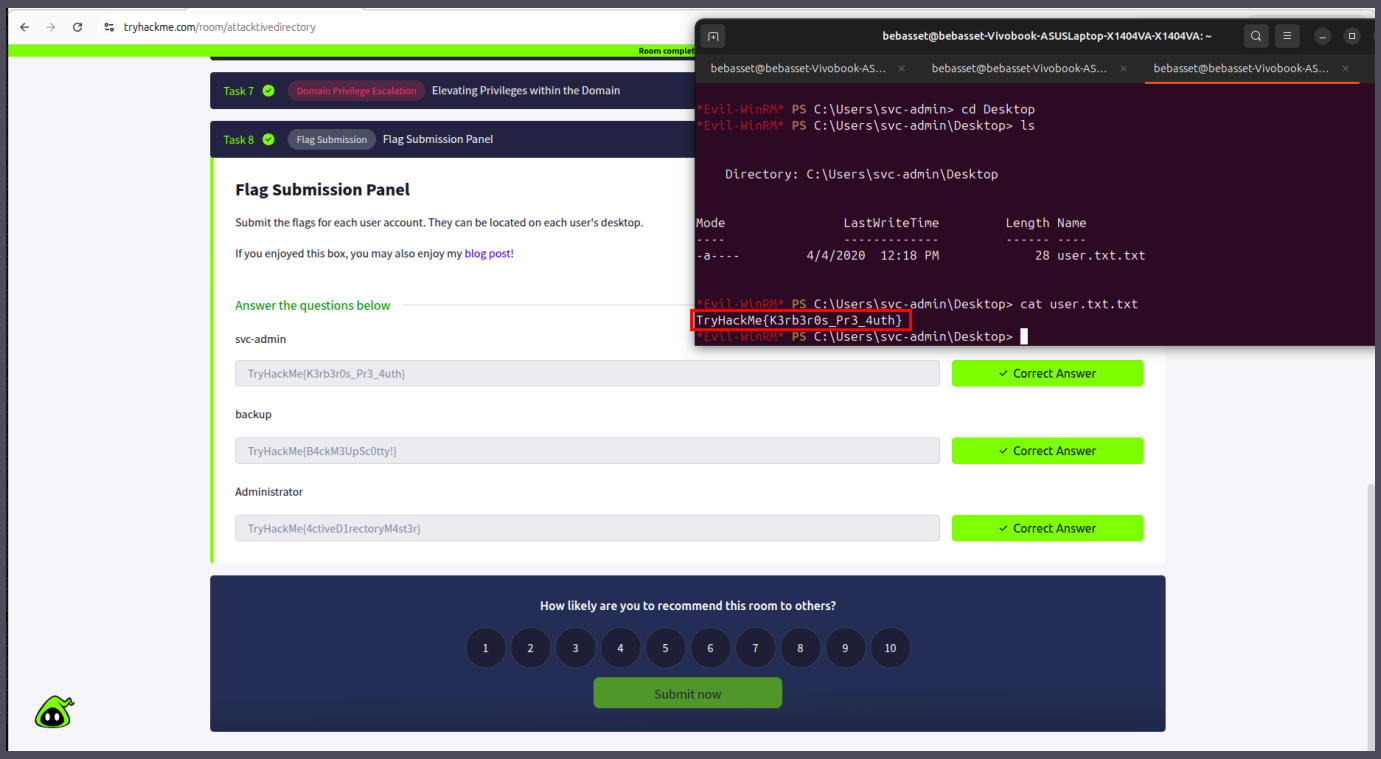
```
cd /Users/backup
cd Desktop
ls
cat PrivEsc.txt
```

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 Attacktive 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.