ADCS Attack Paths in BloodHound — Part 3

posts.specterops.io/adcs-attack-paths-in-bloodhound-part-3-33efb00856ac

Jonas Bülow Knudsen September 11, 2024

Posts from SpecterOps team members on various topics relating information security

In Part 1 of this series, we explained how we incorporated Active Directory Certificate Services (ADCS) objects into BloodHound and demonstrated how to effectively use BloodHound to identify attack paths, including the ESC1 domain escalation technique. Part 2 covered the Golden Certificates and the ESC3 techniques.

In this blog post, we will continue to explore more of the new edges we have introduced with ADCS support in BloodHound. More specifically, we will cover how we have incorporated the ESC6, ESC9, and ESC10 domain escalation techniques.

Keyfactor Technical Team published a blog post in 2016, Hidden Dangers: Certificate Subject Alternative Names (SANs), which describes the dangerous configuration that enables the domain escalation technique Will Schroeder and Lee Chagolla-Christensen later named "ESC6" in their ADCS whitepaper Certified Pre-Owned. Oliver Lyak found and described ESC9 and ESC10 in the blog post Certipy 4.0: ESC9 & ESC10, BloodHound GUI, New Authentication and Request Methods — and more!. Much kudos to these people for sharing their research with the community.

The configuration of *implicit certificate mapping* is a common factor impacting the three techniques, and it is the first topic we will explore.

Certificate Mapping

When you use a password to authenticate in Active Directory (AD), you must specify both a username and a password. A domain controller (DC) will look up the AD account with a matching username and verify that the password you provided is correct for this account.

When you use a certificate instead of a password to authenticate, the DC performs "*certificate mapping*" to verify that the certificate "maps" to the AD account you specified.

Implicit Certificate Mapping

When AD user Alice enrolls a certificate from ADCS, the Certificate Authority (CA) includes the `userPrincipalName` (UPN) of her account in the issued certificate Subject Alternative Name (SAN). When Alice attempts to authenticate with her username and the certificate, the DC finds Alice's AD account with the username and verifies that the certificate maps to the account by confirming the certificate SAN UPN matches the AD account's UPN.

Not all users have a UPN. To account for that, the DC attempts to map the SAN UPN without the "@domain.name" part to the `sAMAccountName` of the AD account as a second try. As a last try, the DC repeats the second attempt but with a "$" added, which gives a match if Alice is a computer since the `sAMAccountName` of computers end with "$".

The DC will check if there are any other accounts with a matching attribute for the given SAN value before moving to the second and third try. If that is the case, then the authentication attempt will fail.

This concept is known as *UPN mapping*. There is a similar concept known as *DNS mapping*, where the `dnsHostName` (DNS) attribute is used instead. In that case, the DC will attempt mapping the hostname part of the SAN DNS value to the `sAMAccountName`.

Abusing Implicit Certificate Mapping

Oliver Lyak found that he could abuse weaknesses in the implicit mapping logic to enroll a certificate as one account and authenticate as another. You can read about Oliver's cool finding here: Certifried: Active Directory Domain Privilege Escalation (CVE-2022–26923).

TLDR: You can change an AD victim account's UPN or DNS attribute to match the `sAMAccountName` of any target account, such that when you enroll a certificate as the victim, the certificate will contain the manipulated attribute value which maps to the target account. You can then use this certificate to login as the target account.

The overall steps of the attack are:

1. Modify a victim account's UPN/DNS attribute to match a target's `sAMAccountName`
2. Enroll a certificate as the victim
3. Authenticate as the target account using the certificate

If you are modifying the DNS attribute in step 1, then you first remove any `servicePrincipalNames` (SPNs) that contains the victim account's DNS name. The DC will automatically attempt to update those SPNs if not deleted, which causes a conflict with the target's SPN and the DNS change failing.

If you are modifying the UPN attribute in step 1, then you have to change the victim's UPN once again after step 2 to avoid the certificate mapping to the victim.

Here is an animation example of how the abuse goes with Kerberos UPN mapping:

The Patch

Microsoft patched the vulnerability by making enterprise CAs add a new certificate extension (i.e., `szOID_NTDS_CA_SECURITY_EXT`) to new certificates containing the enrollee's SID. The extension is commonly referred to as the SID (or security) extension. The extension enables the DC to verify that the account that enrolled the certificate is also the account the certificate maps to and otherwise disallow the authentication attempt. Microsoft refers to the use of the SID extension as *strong* mapping.

Microsoft could not enforce strong mapping overnight as all existing certificates would not have this new SID extension. They therefore introduced a new registry value named `StrongCertificateBindingEnforcement` under `HKLM\System\CurrentControlSet\Services\Kdc` on DCs to roll out the strong mapping enforcement for Kerberos in a controlled manner. Strong certificate mapping for Kerberos has the following modes:

The two weak modes are only temporarily available. The disabled mode is already unsupported, meaning that if you configure a DC with this mode, it behaves as in compatibility mode. Full enforcement is currently scheduled for "*Feb 11, 2025, or later*". This date has been postponed three times, so it may change again.

Strong mapping works with features that enable legitimate impersonation like *Enrollee Supplies Subject* (ESC1) or *Enroll On Behalf Of* (ESC3). The CA allows the enrollee to specify the SID extension in case the certificate template has the Enrollee Supplies Subject flag and it will add the SID extension by itself when you send an Enroll On Behalf Of request.

Microsoft also introduced a new certificate template flag called `NO_SECURITY_EXTENSION`. If enabled, the enterprise CA avoids adding the SID extension to certificates of the given template. This enables admins to keep using applications/scripts that do not add or support the new SID extension. It is also interesting from an abuse perspective, as it enables us to obtain a new certificate that we can use for weak implicit mapping. The flag is disabled by default, though.

Schannel

The patch also updated how Schannel mapping works, disabling weak mapping by default:

Flag	Mapping	Enabled by default
0x01	Subject/Issuer (weak)	No
0x02	Issuer (weak)	No
0x04	UPN (weak)	No
0x08	S4U2Self	Yes
0x10	S4U2Self explicit	Yes

DCs store these flags in the registry value named `CertificateMappingMethods` under `HKLM\System\CurrentControlSet\Control\SecurityProviders\Schannel`.

Schannel tries the Kerberos Service-For-User-To-Self (S4U2Self) mappings first. The 0x08 S4U2Self mapping means the DC uses the Kerberos mapping logic. I do not know what the S4U2Self explicit mapping is for or what it does. You would think it is for Kerberos explicit mapping, but that seems to be enabled by 0x08 S4U2Self based on the tests I have done.

The DC continues to 0x04 UPN mapping if it is enabled if S4U2Self fails. It is this flag that enables weak implicit mapping over Schannel — both UPN mapping and DNS mapping. This mapping does not support the SID extension and ignores it if present.

ESC9 and ESC10 — Abuse of Weak Certificate Mapping

Oliver found that the patch did not prevent his implicit certificate mapping abuse entirely, and it was still possible under two cases, which he named ESC9 and ESC10:

ESC9 requirements (with `NO_SECURITY_EXTENSION` template):

- Certificate template has the `NO_SECURITY_EXTENSION` flag enabled
- `StrongCertificateBindingEnforcement` set to 0/1 (Disabled/Compatibility) OR `CertificateMappingMethods` contains UPN flag

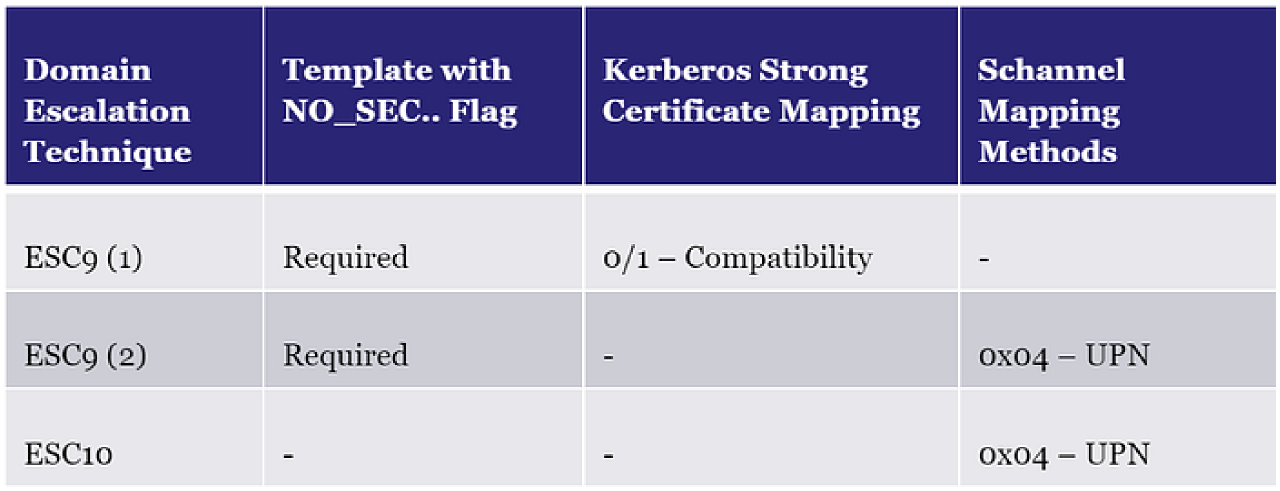
ESC10 requirements (without `NO_SECURITY_EXTENSION` template):

`StrongCertificateBindingEnforcement` set to 0 (Disabled) OR `CertificateMappingMethods` contains UPN flag

The *Abusing Implicit Certificate Mapping* section explains the concept and steps of these techniques. You can also check out Oliver's blog post [Certipy 4.0: ESC9 & ESC10, BloodHound GUI, New Authentication and Request Methods — and more!]() for examples of the techniques executed in a lab environment.

ESC9 and ESC10 Requirements in BloodHound

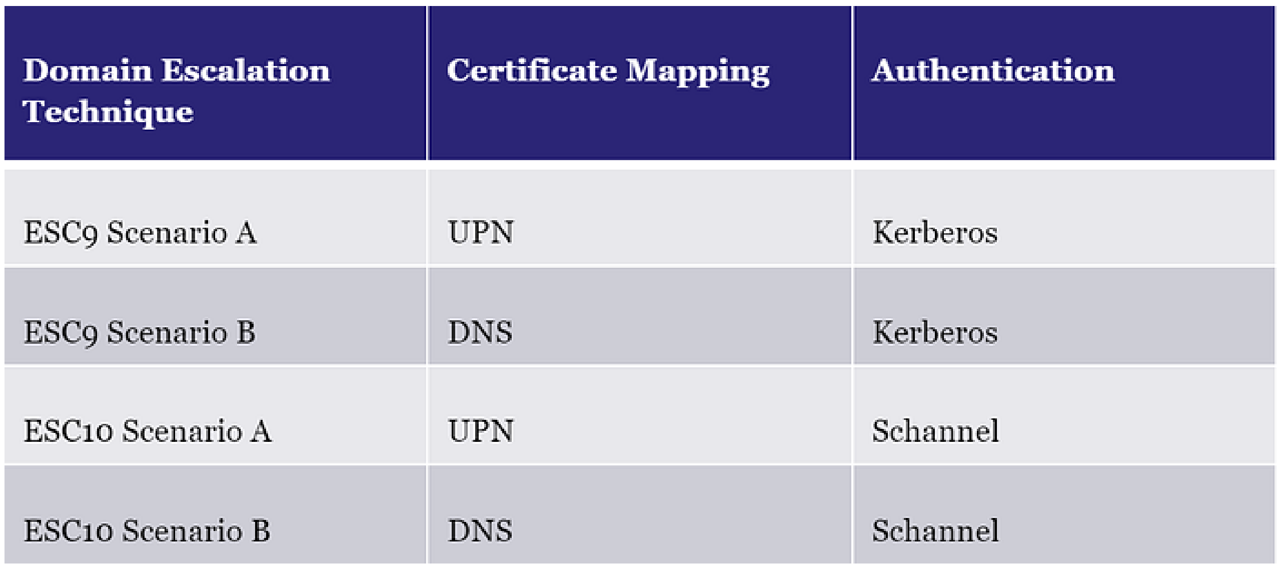
It has been a couple of years since Oliver published ESC9 and ESC10 and the Kerberos strong mapping disabled mode is now unsupported. We can, therefore, eliminate that as an enabling factor for the two techniques. That leaves us with the following cases:



Domain Escalation Technique	Template with NO_SEC.. Flag	Kerberos Strong Certificate Mapping	Schannel Mapping Methods
ESC9 (1)	Required	0/1 – Compatibility	-
ESC9 (2)	Required	-	0x04 – UPN
ESC10	-	-	0x04 – UPN

The SID extension does not affect Schannel UPN mapping, so we can remove the second ESC9 case. As a consequence, ESC9 in BloodHound is for Kerberos authentication and ESC10 for Schannel.

We have split the techniques up into an A and B scenario in BloodHound based on what type of implicit mapping they abuse:



Domain Escalation Technique	Certificate Mapping	Authentication
ESC9 Scenario A	UPN	Kerberos
ESC9 Scenario B	DNS	Kerberos
ESC10 Scenario A	UPN	Schannel
ESC10 Scenario B	DNS	Schannel

This split allows us to provide more precise information and instructions in the edge entity panels, as there are specific details you need to be aware of for both UPN and DNS mapping abuse.

All four technique scenarios have the following requirements:

1) The principal (i.e., attacker) has control over a victim account
2) The victim has enrollment rights on a certificate template
3) The certificate template has manager approval disabled
4) The certificate template does not require any authorized signatures
5) The certificate template is published to an enterprise CA
6) The victim has enrollment rights on the enterprise CA
7) The enterprise CA is trusted for NT authentication
8) The enterprise CA's certificate chain is trusted

ESC9 has the additional requirements for Kerberos authentication to pass:

9) The certificate template defines EKUs that enable Kerberos authentication
10) The certificate template has the `NO_SECURITY_EXTENSION` flag
11) A DC in the forest supports weak Kerberos certificate mapping

ESC10 has similarly these additional requirement for Schannel authentication:

12) The certificate template defines EKUs that enable Schannel authentication
13) A DC in the forest has Schannel UPN mapping enabled

Additionally, for the A scenarios, UPN mapping must work, which requires:

14) The certificate template requires the enrollee's UPN in SAN
15) If the victim is a user, then the certificate template must NOT require the enrollee's DNS in SAN

For the B scenarios, DNS mapping must work, which requires:

16) The certificate template requires the enrollee's DNS in SAN
17) The victim account is a computer

The Part 1 blog post describes requirements 2–9 in detail and how we have implemented the checks for those in BloodHound. We will go through how we have implemented the remaining requirements.

1) The principal (i.e., attacker) has control over a victim account

As the attacker, we need to have sufficient control over the victim account in order to:

1. Modify the UPN/DNS attribute of the victim
2. Impersonate the victim (to request a certificate)

There are many combinations of permissions that provide sufficient control and you can do the impersonation many different ways. You can, for example, impersonate a victim by abusing a ForceChangePassword edge to perform a password reset attack; alternatively, you can abuse an AdminTo edge to request a certificate as a computer victim from the host. However, those edges will not provide sufficient permissions to change the UPN/DNS attributes. We therefore check for the edges in BloodHound that give the permissions to do both the attribute manipulation and impersonation:

- GenericAll
- GenericWrite
- WriteDacl
- WriteOwner
- Owns

10) The certificate template has the `NO_SECURITY_EXTENSION` flag

The certificate template attribute `msPKI-Enrollment-Flag` holds the (`CT_FLAG_`)`NO_SECURITY_EXTENSION` flag. In BloodHound, we check if templates have the flag enabled and store it as a boolean property on the CertTemplate node named *No Security Extension* (`nosecurityextension`).

11) A DC in the forest supports weak Kerberos certificate mapping

The DC supports weak Kerberos certificate mapping if it is in compatibility mode (i.e., the registry key value `StrongCertificateBindingEnforcement` is 0 or 1). It is in compatibility mode by default for now, but that might change next year as described earlier.

The computer node in BloodHound has been updated with the following properties that reveal the Kerberos strong certificate mapping configuration:

- (`strongcertificatebindingenforcementraw`): The raw integer value of the registry setting
- (`strongcertificatebindingenforcement`): A string explaining the meaning of the raw value: (0), (1), or (2)

It only requires a single DC in the forest with compatibility mode, as the attacker can target an account of that given domain and authenticate against this specific DC. BloodHound will verify that at least one such DC exists.

12) The certificate template defines EKUs that enable Schannel authentication

Not all EKUs that enable Kerberos authentication works for Schannel authentication:

EKU	Enables Kerberos Authentication	Enables Schannel Authentication
Client Authentication (1.3.6.1.5.5.7.3.2)	True	True
Any Purpose (2.5.29.37.0)	True	True
SubCA (*no EKU*)	True	True
PKINIT Client Authentication (1.3.6.1.5.2.3.4)	True	False
Smart Card Logon (1.3.6.1.4.1.311.20.2.2)	True	False

To make life simple for BloodHound users, we have added a certificate template property named *Schannel Authentication Enabled* (`schannelauthenticationenabled`) which reveals if you can use certificates of a given template for Schannel authentication. The logic for creating the property besides the EKUs part is the same as for the *Authentication Enabled* (Kerberos) property described in Part 1.

13) A DC in the forest has Schannel UPN mapping enabled

The DC registry key value `CertificateMappingMethods` must contain the 0x04 UPN flag to enable Schannel UPN/DNS mapping. This flag is not set by default.

The computer node in BloodHound has been updated with the following properties that reveal the Schannel certificate mapping configuration:

- (`certificatemappingmethodsraw`): The raw integer value of the registry setting
- (`certificatemappingmethods`): A list of the supported mappings based on the raw value:-

BloodHound will verify that at least one DC in the forest has the 0x04 flag in this setting.

14) The certificate template requires the enrollee's UPN in SAN

The enterprise CA will put the enrollee's UPN into the SAN of the issued certificate if the certificate template has one of the following flags in its `msPKI-Certificate-Name-Flag` attribute:

- `CT_FLAG_SUBJECT_ALT_REQUIRE_UPN`

- CT_FLAG_SUBJECT_ALT_REQUIRE_SPN

Yes — the SPN flag will make the UPN included. Microsoft has documented it here.

We have implemented the two flags in BloodHound as boolean CertTemplate properties:

- (subjectaltrequireupn)
- (subjectaltrequirespn)

BloodHound verifies that the found CertTemplate has any of these properties set to true.

15) If the victim is a user, then the certificate template must NOT require the enrollee's DNS in SAN

There are two certificate template msPKI-Certificate-Name-Flag flags that require the enrollee has their DNS attribute set:

- CT_FLAG_SUBJECT_ALT_REQUIRE_DNS
- CT_FLAG_SUBJECT_ALT_REQUIRE_DOMAIN_DNS

The first one will include the DNS attribute in the SAN and the second one is just for the domain name of the attribute. If the enrollee does not have their DNS attribute set, enrollment will fail.

The corresponding boolean CertTemplate properties are:

- (subjectaltrequiredns)
- (subjectaltrequiredomaindns)

You cannot set the DNS attribute on a victim user account as the AD user class does not include the DNS attribute. BloodHound therefore verifies that both these flags are not set on the CertTemplate if the victim account is a user.

16) The certificate template requires the enrollee’s DNS in SAN

BloodHound verifies that the CertTemplate has the Subject Alternative Name Require DNS property described above set to true.

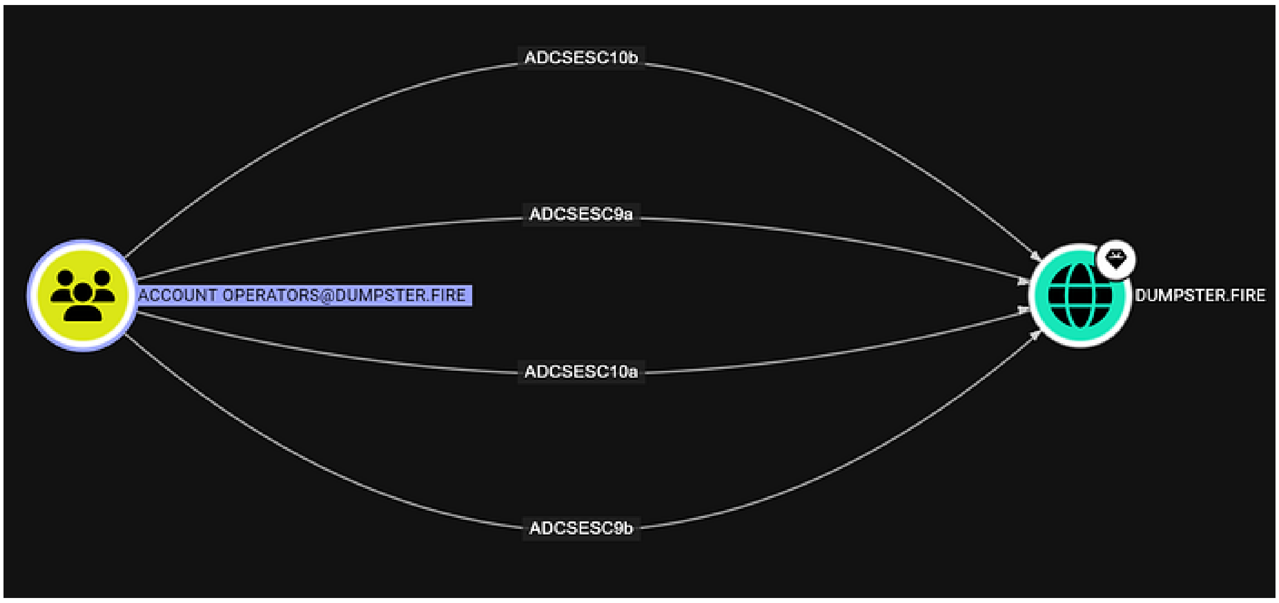
17) The victim account is a computer

As we discussed above, users cannot enroll in certificate templates with the DNS flag. So when that is the case, the victim must be a computer node (or a gMSA or sMSA which is derived from the AD computer class).

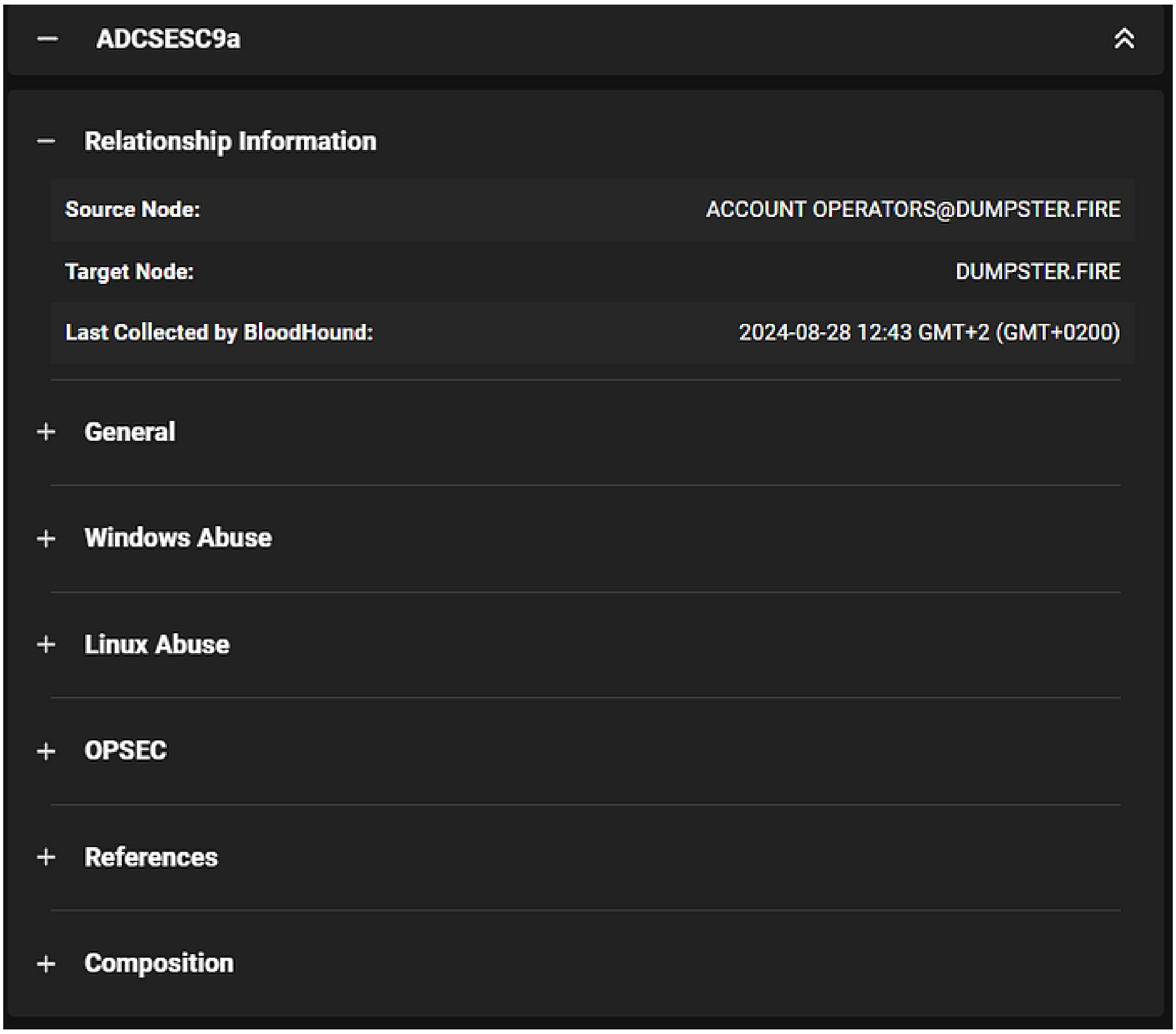
ESC9 and ESC10 — New BloodHound Edges

If all the above requirements are met, then BloodHound creates the following new edges going from the attacking principal towards the root domain node:

- ESC9 Scenario A: ADCSESC9a
- ESC9 Scenario B: ADCSESC9b
- ESC10 Scenario A: ADCSESC10a
- ESC10 Scenario B: ADCSESC10b



BloodHound shows the edge entity panel if you clicking on any of the edges, which contain information about the edge:



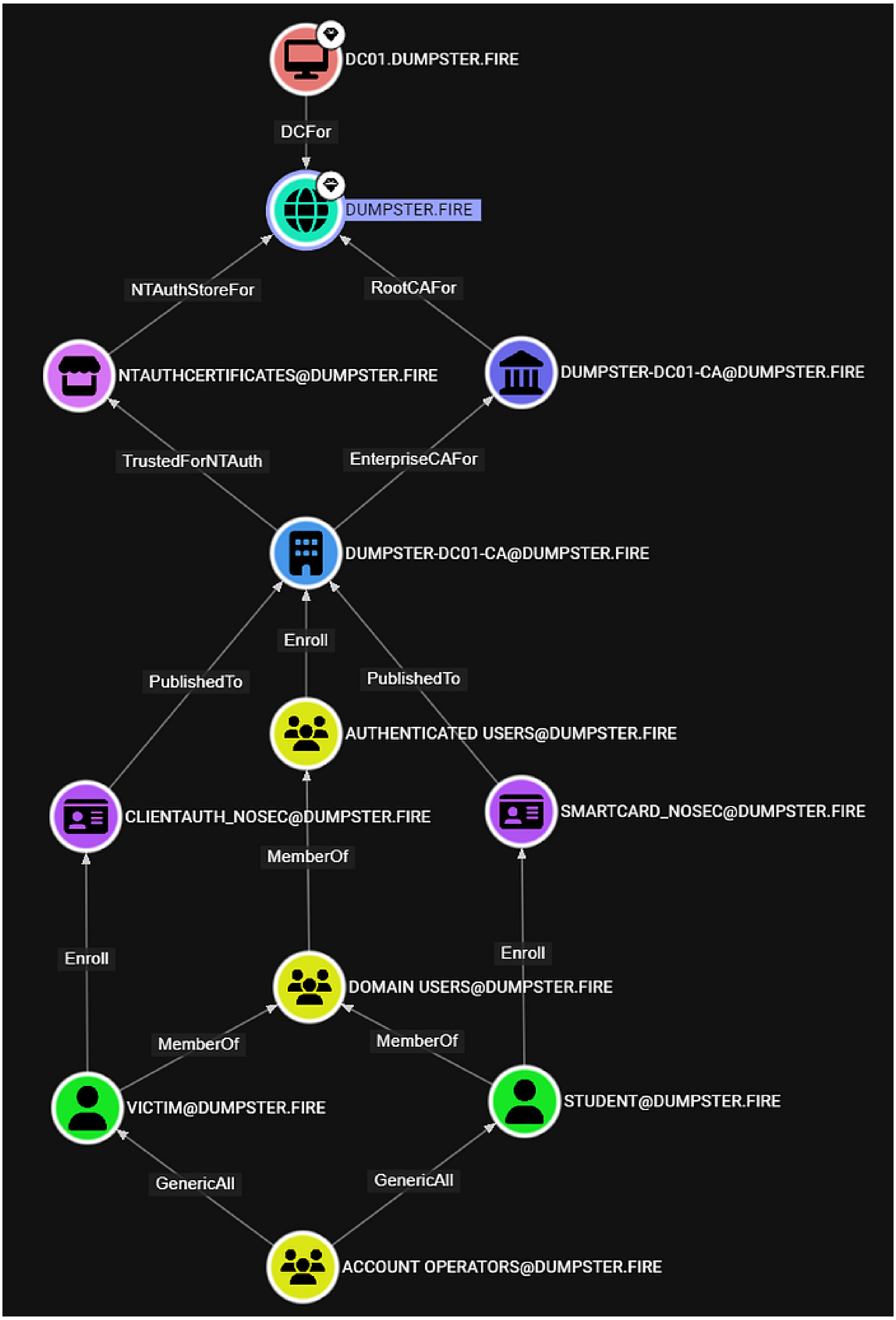
ADCSESC9a

Relationship Information

Source Node:	ACCOUNT OPERATORS@DUMPSTER.FIRE
Target Node:	DUMPSTER.FIRE
Last Collected by BloodHound:	2024-08-28 12:43 GMT+2 (GMT+0200)

You can also find the edge information in the BloodHound support docs.

Like the other ADCSESCx edges, the edges for ESC9 and ESC10 have a composition graph that BloodHound will show if you click on the *Composition* section in the entity panel, revealing the nodes that meet the requirements for the ADCSESCx edge. Here is the composition graph for the ADCSESC9a edge from Account Operators:


DC01.DUMPSTER.FIRE
DCFor
DUMPSTER.FIRE
NTAuthStoreFor
RootCAFor
NTAUTHCERTIFICATES@DUMPSTER.FIRE
DUMPSTER-DC01-CA@DUMPSTER.FIRE
TrustedForNTAuth
EnterpriseCAFor
DUMPSTER-DC01-CA@DUMPSTER.FIRE
Enroll
PublishedTo
PublishedTo
AUTHENTICATED USERS@DUMPSTER.FIRE
CLIENTAUTH_NOSEC@DUMPSTER.FIRE
SMARTCARD_NOSEC@DUMPSTER.FIRE
MemberOf
Enroll
Enroll
DOMAIN USERS@DUMPSTER.FIRE
MemberOf
MemberOf
VICTIM@DUMPSTER.FIRE
STUDENT@DUMPSTER.FIRE
GenericAll
GenericAll
ACCOUNT OPERATORS@DUMPSTER.FIRE

Account Operators have control over two victim user accounts — that means we can abuse any of the two in an attack. The victims have enrollment rights on two certificate templates that meet the requirements for the attack. An enterprise CA has both templates published and grants enrollment rights to the victims through group membership. The enterprise CA chains up to a root CA for the domain which trusts the enterprise CA for NT authentication. At last, we have the computer node in the graph, which is the DC that supports weak Kerberos certificate mapping.

The composition graph provides us with information about the components we need to execute the attack or remediate it. You can find the abuse steps for Windows and Linux in the abuse sections of the edge entity panel.

ESC6 — Abuse of `EDITF_ATTRIBUTESUBJECTALTNAME2`

The key requirement for ESC6 is that an enterprise CA has the dangerous `EDITF_ATTRIBUTESUBJECTALTNAME2` flag. When this flag is set, the enrollee may specify the SAN of the certificate in the certificate request using any certificate template. Therefore, an attacker can use any published certificate template they have enrollment rights on to obtain a certificate as any user or computer in the AD forest. Crazy stuff.

Strong Certificate Mapping and ESC6

The MS patch discussed earlier makes the `EDITF_ATTRIBUTESUBJECTALTNAME2` functionality harder to use. The enterprise CA still allows you to specify the SAN, but it does not allow you to specify the SID extension.

However, some time after the patch, Microsoft released an update with an interesting feature called "SAN URI". This feature enables you to create a certificate valid for strong certificate mapping by including the principal SID in an URL component of the SAN instead of in the SID extension. The DC will first check the SID extension of the certificate and authentication will fail if there is a mismatch. But, if the certificate does not contain the SID extension, then the DC will check if the SAN URL contains a matching SID.

That means we can abuse the SAN URI feature together with a `NO_SECURITY_EXTENSION` flag certificate template to make the ESC6 attack work over Kerberos — even if the DC enforces strong certificate mapping.

You can specify the SAN URL component when you enroll a certificate using Certify with the recently added `/url` parameter:

```
Certify.exe request /ca:SERVER\ca-name /template:ESC6-Template
/altname:target /url:tag:microsoft.com,2022-09-14:sid:S-1-5-21-2697957641-
2271029196-387917394-2136
```

The value of the URL component must follow this pattern of the above example with the microsoft.com tag and the specific date.

For Schannel, ESC6 relies on the same UPN flag as ESC10.

ESC6 Requirements in BloodHound

The base requirement we check for when creating ESC6 edges in BloodHound are:

1) Principal has enrollment rights on a certificate template
2) The certificate template has manager approval disabled
3) The certificate template does not require any authorized signatures
4) The certificate template is published to an enterprise CA
5) Principal has enrollment rights on the enterprise CA
6) The enterprise CA is trusted for NT authentication
7) The enterprise CA's certificate chain is trusted
8) The enterprise CA's `EDITF_ATTRIBUTESUBJECTALTNAME2` flag is enabled

Additionally, the requirements for either Kerberos or Schannel authentication must be met.

Kerberos authentication:

9) The certificate template defines EKUs that enable Kerberos authentication
10) The certificate template has the `NO_SECURITY_EXTENSION` flag

Schannel authentication:

11) The certificate template defines EKUs that enable Schannel authentication
12) A DC in the forest has Schannel UPN mapping enabled

The Part 1 blog post describes requirement 1–7 in detail and how we have implemented the checks for those in BloodHound, and we went through requirement 9–12 in the section for ESC9 and ESC10.

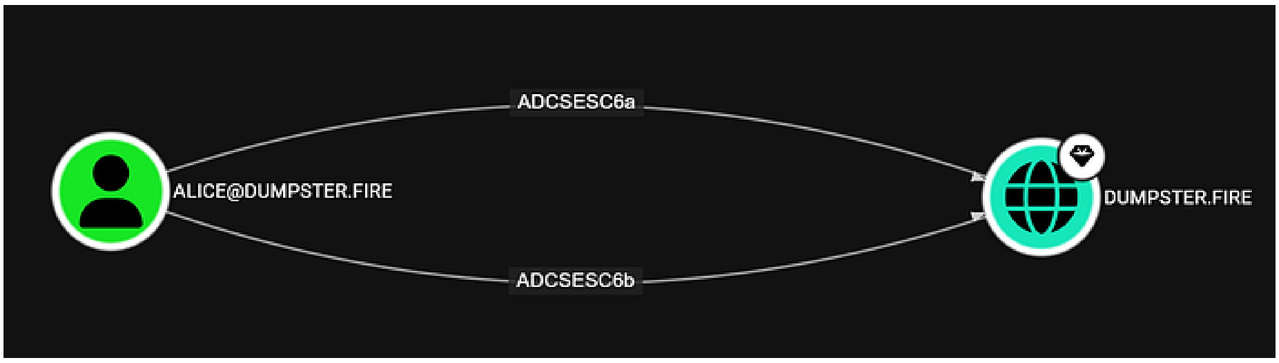
Requirement 8 is the only unique requirement for ESC6. For that, we have added a boolean property on EnterpriseCA nodes named *Is User Specifies San Enabled* (`isuserspecifiessanenabled`) which reveals whether the CA has `EDITF_ATTRIBUTESUBJECTALTNAME2` enabled.

ESC6 — New BloodHound Edges

Similarly to the implementation of ESC9 and ESC10, we have divided ESC6 up into two scenarios, this time based on what type of authentication is possible:

- ESC6 Scenario A (ADCSESC6a): ESC6 requirements met with Kerberos
- ESC6 Scenario B (ADCSESC6b): ESC6 requirements met with Schannel

We have implemented the ADCSESC6 edges in the same way as the other ADCSESCx edges going from the principal that has the privileges to perform the attack towards the domain node of the forest root:



The ESC6 edges have a composition graph that reveals the components involved in the same manner as the other ADCSESCx edges.

DC Registry Collection

We have added support in SharpHound for collecting the `StrongCertificateBindingEnforcement` and `CertificateMappingMethods` registry values from DCs with the *DCRegistry* collection method. BloodHound will not produce any ESC6 Scenario B, ESC9, or ESC10 edges without the registry values collected.

However, the account running SharpHound must have admin rights on the DCs to collect this information. As a defender, you might be able to run SharpHound with those privileges, but it is likely not an option if you are a red teamer or penetration tester.

As an alternative, you can manually replace the `DCRegistryData` JSON object in the SharpHound computers.json output file under a DC node to assume it meets the requirements for weak certificate mapping for both Kerberos and Schannel:

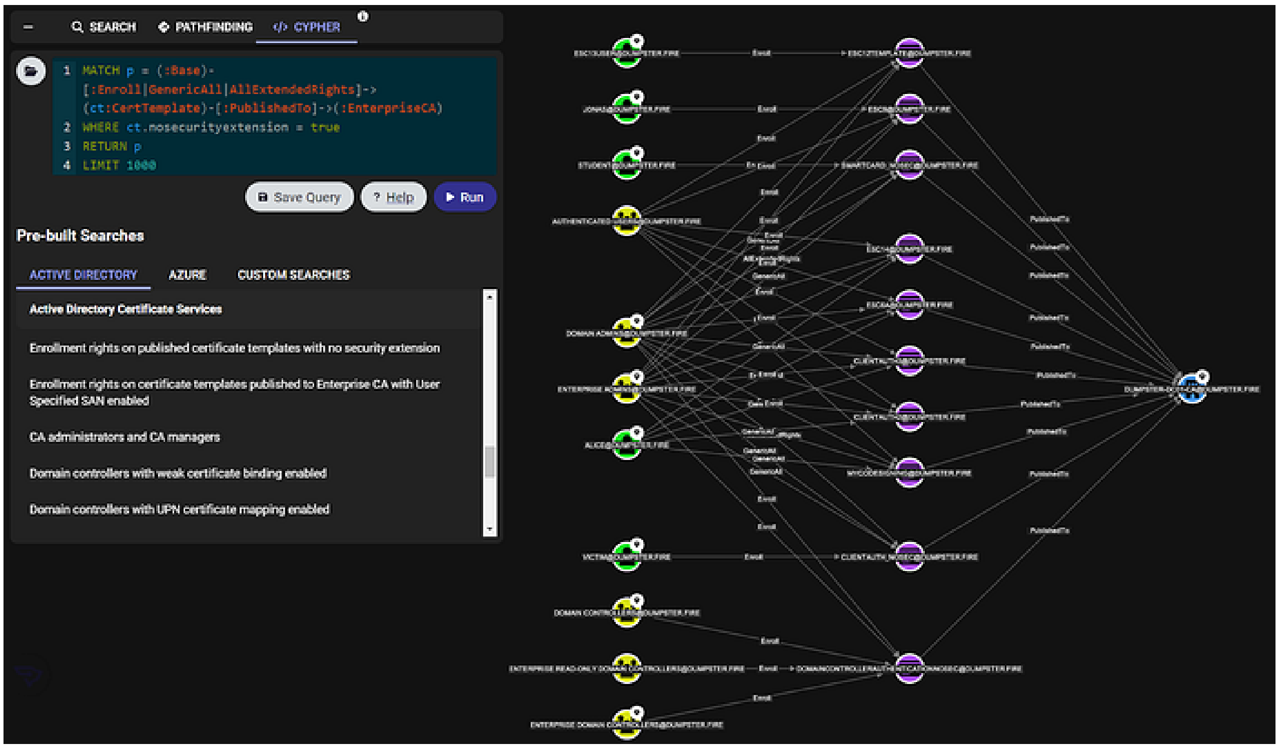
You can also assume default configuration by changing the 31 integer to 24.

Alternatively, you can use some of the predefined queries covered in the next section to find ESC6/9/10 opportunities without ADCSESCx edges.

Predefined Cypher Queries

We have a handful of new predefined cypher queries that can help you find interesting things related to ESC6/9/10:

- Enrollment rights on published certificate templates with no security extension
- Enrollment rights on certificate templates published to Enterprise CA with User Specified SAN enabled
- Domain controllers with weak certificate binding enabled
- Domain controllers with UPN certificate mapping enabled


SEARCH
PATHFINDING
CYPHER
Save Query
Help
Run
ACTIVE DIRECTORY
AZURE
CUSTOM SEARCHES
JONAS@DUMPSTER.FIRE
STUDENT@DUMPSTER.FIRE
DOMAIN ADMINS@DUMPSTER.FIRE
ENTERPRISE ADMINS@DUMPSTER.FIRE
ALICE@DUMPSTER.FIRE
VICTIM@DUMPSTER.FIRE
DOMAIN CONTROLLERS@DUMPSTER.FIRE
ENTERPRISE READ-ONLY DOMAIN CONTROLLERS@DUMPSTER.FIRE
ENTERPRISE DOMAIN CONTROLLERS@DUMPSTER.FIRE
Enroll
PublishedTo

```
MATCH p = (:Base)-[:Enroll|GenericAll|AllExtendedRights]->(ct:CertTemplate)-[:PublishedTo]->(:EnterpriseCA)
WHERE ct.nosecurityextension = true
RETURN p
LIMIT 1000
```

Pre-built Searches

Active Directory Certificate Services

Enrollment rights on published certificate templates with no security extension

Enrollment rights on certificate templates published to Enterprise CA with User Specified SAN enabled

CA administrators and CA managers

Domain controllers with weak certificate binding enabled

Domain controllers with UPN certificate mapping enabled

Remediation

You are a defender and you have discovered (or been made aware) that your AD environment has ADCS ESC6/9/10 attack paths. Great...

ESC6 Remediation

You should get rid of the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag on your enterprise CAs. But first, try to get a clear understanding of the current usage of the functionality enabled by the flag so you do not break the environment.

1) Identify `EDITF_ATTRIBUTESUBJECTALTNAME2` Usage

When a principal creates a certificate request that is supposed to use the `EDITF_ATTRIBUTESUBJECTALTNAME2` feature, then the certificate request incorporates the SAN name with X509 name/value pairs. This is different from Enrollee Supplies Subject (ESC1) requests where an X509 extension is used, and it therefore allows us to distinguish between the two types.

You can use PSPKIAudit's `Get-CertRequest` function as a CA Administrator to identify certificate requests utilizing the `EDITF_ATTRIBUTESUBJECTALTNAME2` feature:

```
CA : DC01.dumpster.fire\dumpster-DC01-CARequestID : 180RequesterName :
DUMPSTER\jdRequesterMachineName : DC01.dumpster.fireRequesterProcessName :
Certify.exeSubjectAltNamesExtension : RoshiSubjectAltNamesAttrib :
RoshiSerialNumber : 7b000000b4822ab66661d867a10000000000b4CertificateTemplate :
ClientAuth2
(1.3.6.1.4.1.311.21.8.4571196.1884641.3293620.10686285.12068043.134.3767878.1854842
 : 12/13/2023 7:23:13 PMStartDate : 12/13/2023 7:13:13 PMEndDate : 12/12/2024
7:13:13 PM
```

These requests can help you determine which users (and from what computers) are utilizing the feature. You can ignore certificate requests with an expired end date, and also requests where the requester name refers to the same account as the `SubjectAltNamesAttrib` property.

2) Implement a More Secure Solution

The most secure feature allowing the creation of certificates as other accounts is *enrollment agents* (ESC3). You can use enrollment agent restrictions to configure which accounts a given enrollment agent can enroll as, preventing unintended domain escalation scenarios.

Alternatively, you can configure a certificate template with the Enrollee Supplies Subject flag (ESC1). This allows the enrollees to enroll as anyone, which is why you must restrict enrollment rights to Tier Zero accounts on such a template.

3) Remove `EDITF_ATTRIBUTESUBJECTALTNAME2` flag

Log in on the enterprise CA server as CA Administrator (or member of the Administrators on the host) and open CMD as Administrator. Check whether the CA has EDITF_ATTRIBUTESUBJECALTNAME2 enabled:

```
certutil -getreg policy\EditFlags
```

The CA has If `EDITF_ATTRIBUTESUBJECTALTNAME2` enabled if the output includes the flag. To disable the setting, run the following command:

```
certutil -setreg policy\EditFlags -EDITF_ATTRIBUTESUBJECTALTNAME2
```

Then restart the CA service:

```
net stop certsvc && net start certsvc
```

If you need to enable the flag again, then run this command followed by a restart of the CA service:

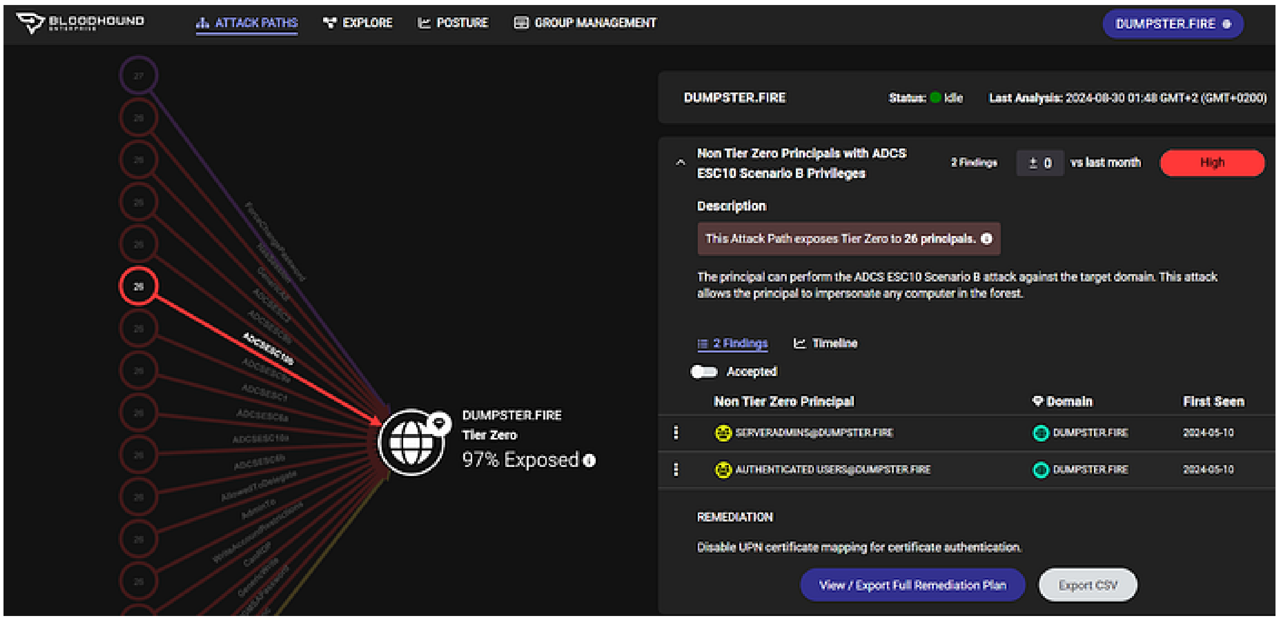
```
certutil -setreg policy\EditFlags +EDITF_ATTRIBUTESUBJECTALTNAME2
```

ESC9 and ESC10 Remediation

Check out the [KB5014754: Certificate-based authentication changes on Windows domain controllers]() article from Microsoft. It provides helpful information about strong certificate mapping enforcement and how you can use event IDs to track down accounts and systems that are not compliant.

ESC6/9/10 in BloodHound Enterprise

The six new edges are all traversable. That means we generate a finding for every non-Tier Zero principal that has any of the edges in BloodHound Enterprise so you can become easily aware of the risk:



DUMPSTER.FIRE Status: Idle Last Analysis: 2024-08-30 01:48 GMT+2 (GMT+0200)

Non Tier Zero Principals with ADCS ESC10 Scenario B Privileges 2 Findings ± 0 vs last month High

Description

This Attack Path exposes Tier Zero to 26 principals.

The principal can perform the ADCS ESC10 Scenario B attack against the target domain. This attack allows the principal to impersonate any computer in the forest.

2 Findings Timeline

Accepted

Non Tier Zero Principal	Domain	First Seen
SERVERADMINS@DUMPSTER.FIRE	DUMPSTER.FIRE	2024-05-10
AUTHENTICATED USERS@DUMPSTER.FIRE	DUMPSTER.FIRE	2024-05-10

REMEDIATION

Disable UPN certificate mapping for certificate authentication.

View / Export Full Remediation Plan Export CSV

The findings also include a full remediation plan.

Everything else covered in this blog post is available in both BloodHound Community Edition (CE) and Enterprise.

If you are interested in learning more about the features of BloodHound Enterprise then check out the [comparison of CE and Enterprise]() or schedule an informational [BloodHound Enterprise demo]() with us.

What is Next

We have now covered ESC1, Golden Certificate, ESC3, ESC6, ESC9, and ESC10 with this blog post series. Stay tuned for future posts, as we will dive further into more ADCS escalations and how you can identify them using BloodHound.

We are eager to hear your feedback. Please join us in the BloodHound Slack or report any issues on the BloodHound GitHub repo.