

ADCS Attack Paths in BloodHound — Part 1

posts.specterops.io/adcs-attack-paths-in-bloodhound-part-1-799f3d3b03cf

Jonas Bülow Knudsen

January 24, 2024

Posts from SpecterOps team members on various topics relating information security

Since

[Will Schroeder](#)

[Certified Pre-Owned](#)

This blog post details the ESC1 domain escalation requirements and explains how BloodHound incorporates the relevant components. We will demonstrate how to effectively use BloodHound to identify attack paths that involve ESC1 abuse.

If you are new to ADCS attacking techniques or need a recap of how ADCS works, we recommend reading through the *Background* section of the Certified Pre-Owned whitepaper. For those who prefer watching over reading, check out the recording of our webinar: [AD CS Attack Paths in BloodHound](#).

Acknowledgments

The community has played a crucial role in our work by sharing research and tooling on ADCS. A big thank you to everyone contributing knowledge!

Special thanks to

[Oliver Lyak](#)

[Certipy](#)

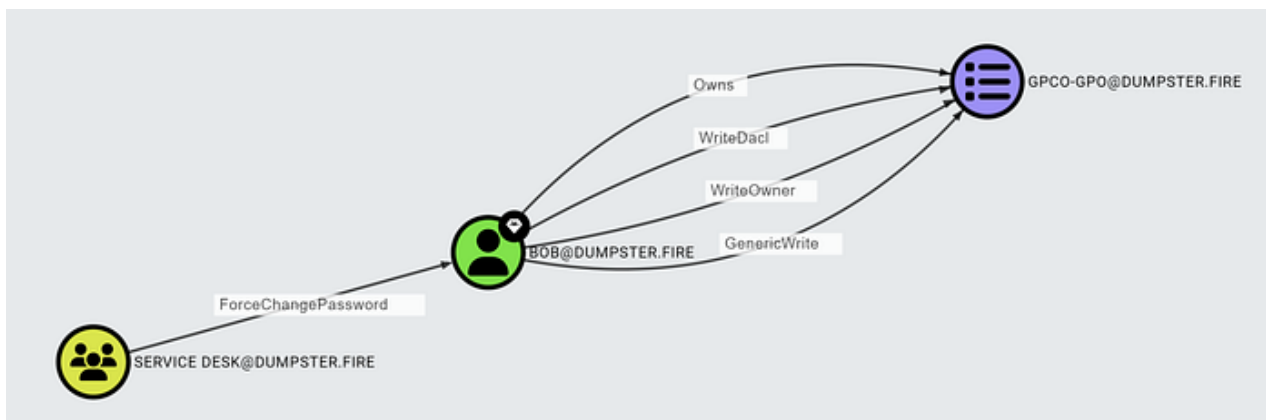
ADCS in BloodHound

With the implementation of ADCS attack paths in BloodHound, we introduce a multitude of *non-traversable* edges. Let's first understand the concept behind traversable and non-traversable edges before delving into the new features.

Traversable Edges

Most edges in BloodHound are traversable, representing a relationship between two nodes where the starting node can take control of the ending node to a degree that allows an attacker to abuse outgoing edges.

For example, consider the ForceChangePassword edge:

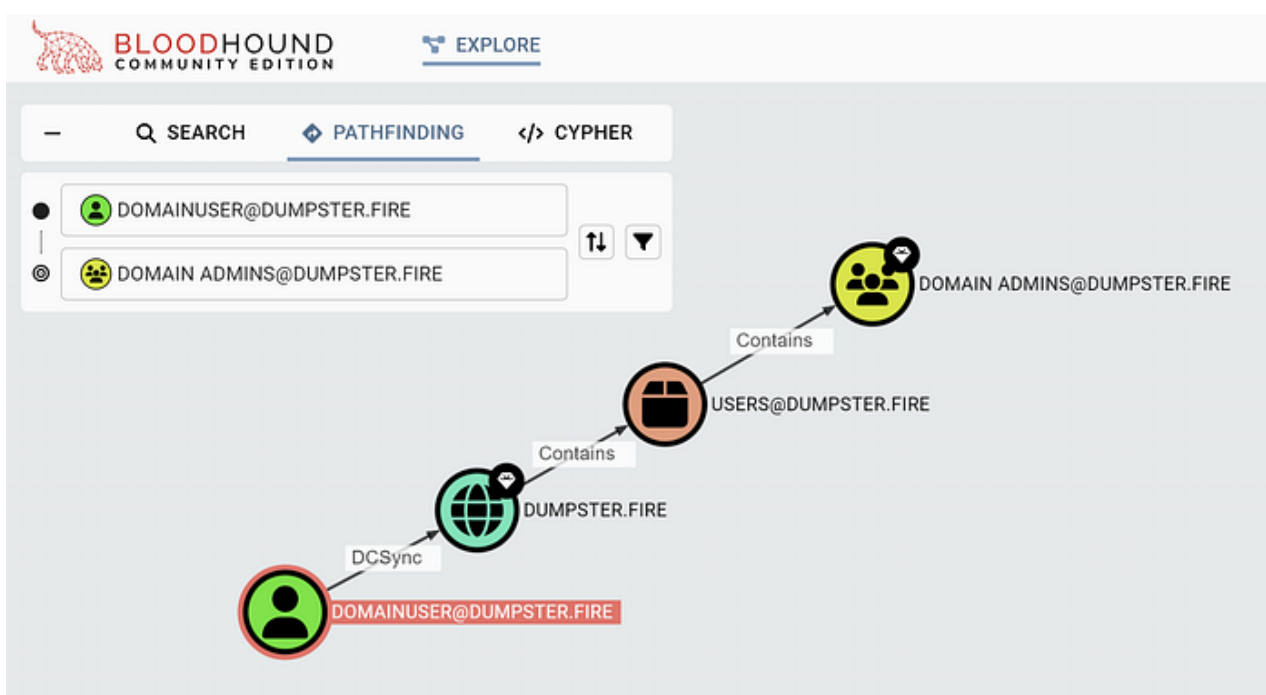


The Service Desk group has permission to force change the password of Bob without knowing Bob's current password. An attacker can abuse this to change the password, log in as Bob, and exploit Bob's privileges. Traversable edges like ForceChangePassword facilitate graph traversal and enable the pathfinding logic in BloodHound.

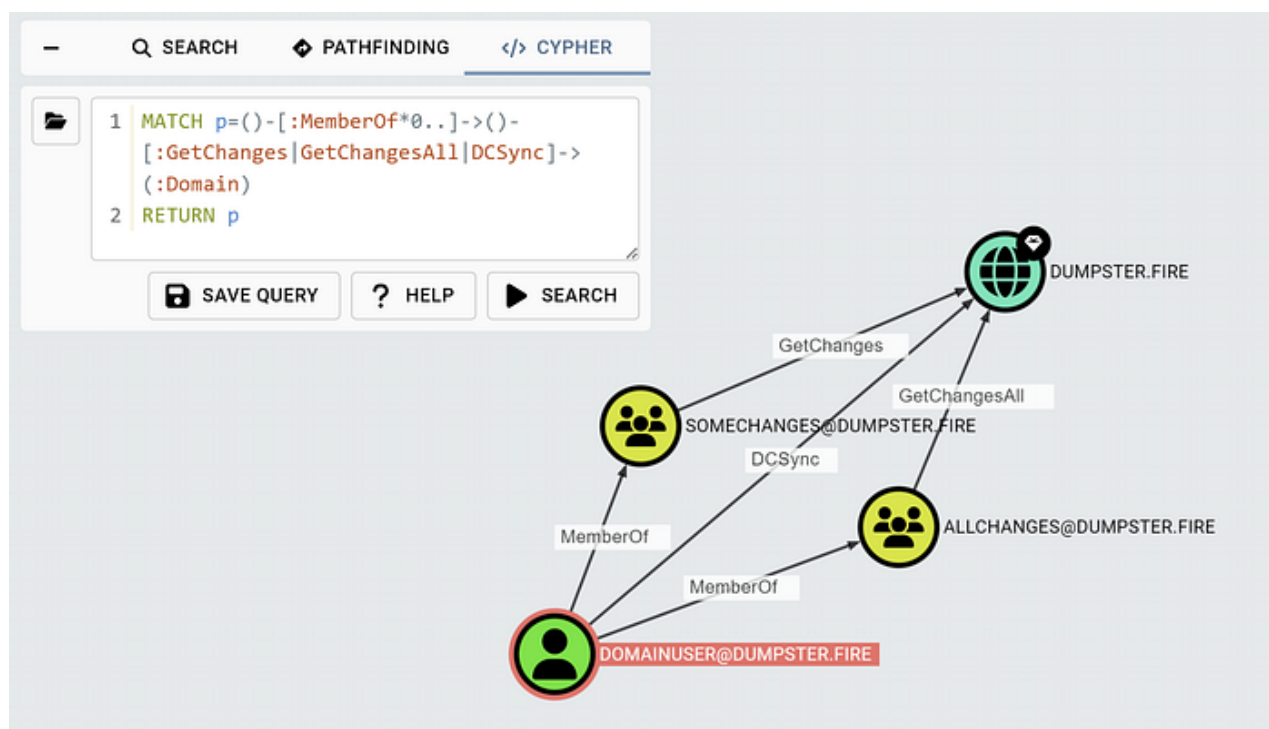
Non-Traversable Edges

If you cannot abuse a given relationship between two nodes to take control of the end node, then the relationship is non-traversable. However, some relationships are non-traversable but can form a traversable relationship when combined. An example is the DCSync attack narrative, discovered and implemented into mimikatz by [Vincent Le Toux](#). [GetChanges](#) and [GetChangesAll](#) permissions on the domain object combined enable you to perform the DCSync attack. GetChanges and GetChangesAll are non-traversable edges, and BloodHound uses them to produce the traversable DCSync edge in what we call the *post-processing* logic.

Pathfinding includes only traversable edges. As a result, you might get a DCSync edge in a path like this:



But you will not see any GetChanges or GetChangesAll edge. However, you can use Cypher to reveal the GetChanges and GetChangeAll edges that the DCSync edge relies on:



ESC1 in BloodHound

The Certified Pre-Owned whitepaper outlines the ESC1 requirements:

- 1) An overly permissive certificate template security descriptor grants certificate enrollment rights to low-privileged users.
- 2) The certificate template allows requesters to specify a subjectAltName in the CSR.
- 3) The certificate template defines EKUs that enable domain authentication.
- 4) Manager approval is disabled.
- 5) No authorized signatures are required.
- 6) The Enterprise CA grants low-privileged users enrollment rights.

Additionally, the Enterprise CA must meet the following requirements for a successful ESC1 attack:

- 7) The Enterprise CA is trusted for NT authentication.
- 8) The Enterprise CA's certificate chain is trusted.


Last but not least, there is an implicit requirement:

- 9) The Enterprise CA has the certificate template published.

We will now go through the requirements one by one to explore how we have implemented them in BloodHound and how we can find the principals that can perform an ESC1 attack.

Certificate Template Requirements

We have introduced a new node type called *CertTemplate* to represent the AD objects of the *pKICertificateTemplate* LDAP class:



EXCHANGEUSER@DUMPSTER.FIRE	
Display Name:	Exchange User
Object ID:	0A7DCC85-5582-4A9C-9ACC-35277F52BCCB
ACL Inheritance Denied:	TRUE
Authentication Enabled:	FALSE
Authorized Signatures Required:	0
Certificate Name Flags:	ENROLLEE_SUPPLIES_SUBJECT
Created:	2023-05-13 15:56 GMT+2 (GMT+0200)
Distinguished Name:	CN=EXCHANGEUSER,CN=CERTIFICATE TEMPLATES,CN=PUBLIC KEY SERVICES,CN=SERVICES,CN=CONFIGURATION,DC=DUMP...
Domain FQDN:	DUMPSTER.FIRE
Domain SID:	S-1-5-21-2697957641-2271029196-387917394
Effective EKUs:	1.3.6.1.5.5.7.3.4
Enhanced Key Usage:	1.3.6.1.5.5.7.3.4
Enrollee Supplies Subject:	TRUE
Enrollment Flags:	INCLUDE_SYMMETRIC_ALGORITHMS
Last Collected by BloodHound:	2024-01-22 13:03 GMT+1 (GMT+0100)
No Security Extension:	FALSE
OID:	1.3.6.1.4.1.311.21.8.4571196.1884641.3293620.1068628...
Renewal Period:	6 weeks
Requires Manager Approval:	FALSE
Schema Version:	1

Let's delve into the *CertTemplate* node properties and relevant edges through the ESC1 requirements.

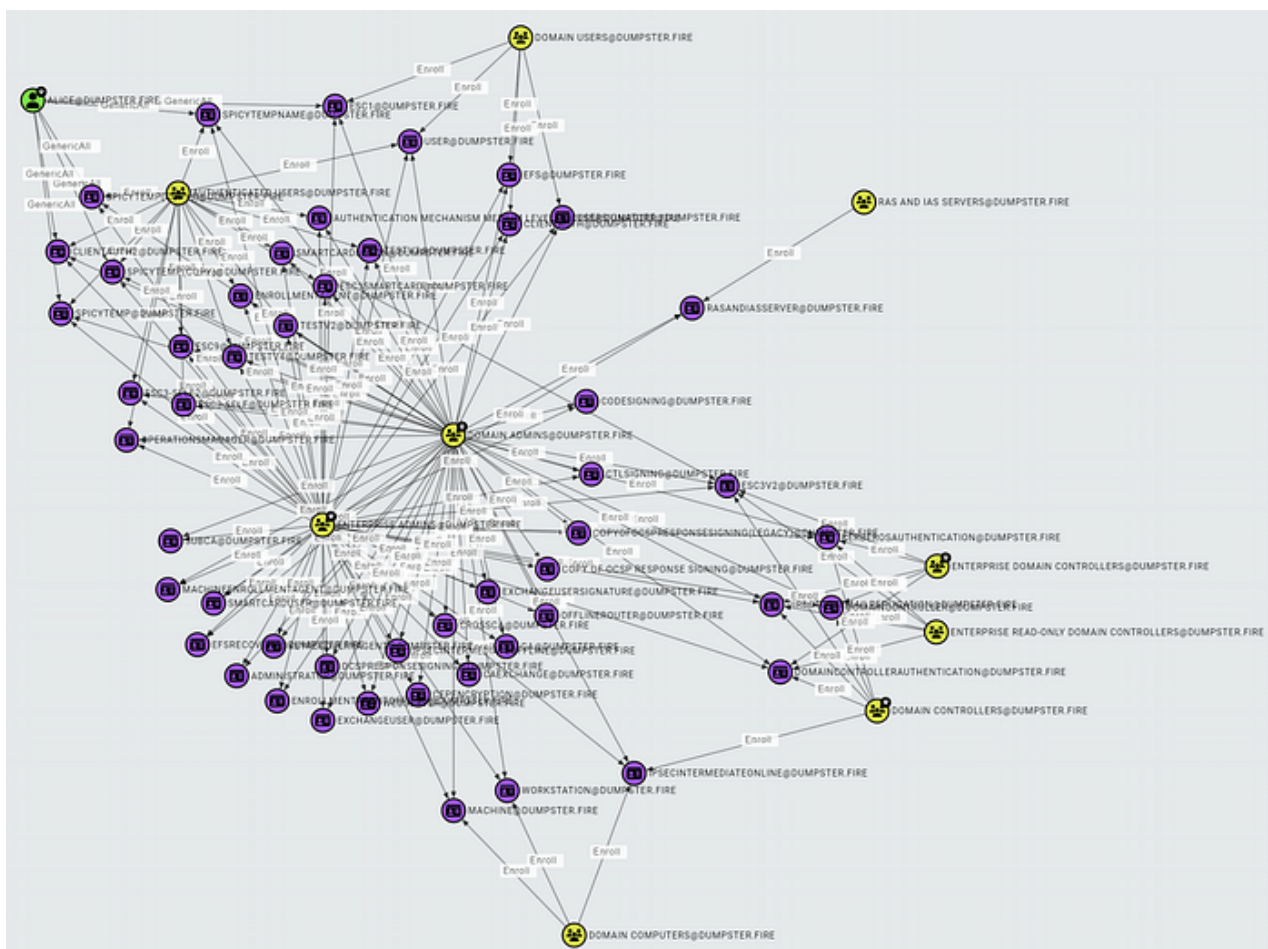
ESC1 Requirement 1: An overly permissive certificate template security descriptor grants certificate enrollment rights to low-privileged users.

The enrollee principal must have at least one of the following permissions/edges on the certificate template to enroll a certificate:

- : Enroll edge (new!)
- : AllExtendedRights edge
- : GenericAll edge

To find all principals with certificate enrollment rights, use this Cypher query:

```
p = ()-[:|||]->(:) p
```



ESC1 Requirement 2: The certificate template allows requesters to specify a subjectAltName in the CSR.

The certificate template allows the enrollee to specify the **subjectAltName** (SAN) in the issued certificate if the template has the **CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT** flag present in its **msPKI-Certificate-Name-Flag** attribute. You can check if the flag is present in BloodHound under “*Enrollee Supplies Subject*” in the CertTemplate entity panel or by checking the CertTemplate node’s **enrolleesuppliessubject** property.

We can use the **enrolleesuppliessubject** node property to extend our query to filter out certificate templates without the **CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT** flag:

```
p = ()-[:|||]->(:) ct. = p
```

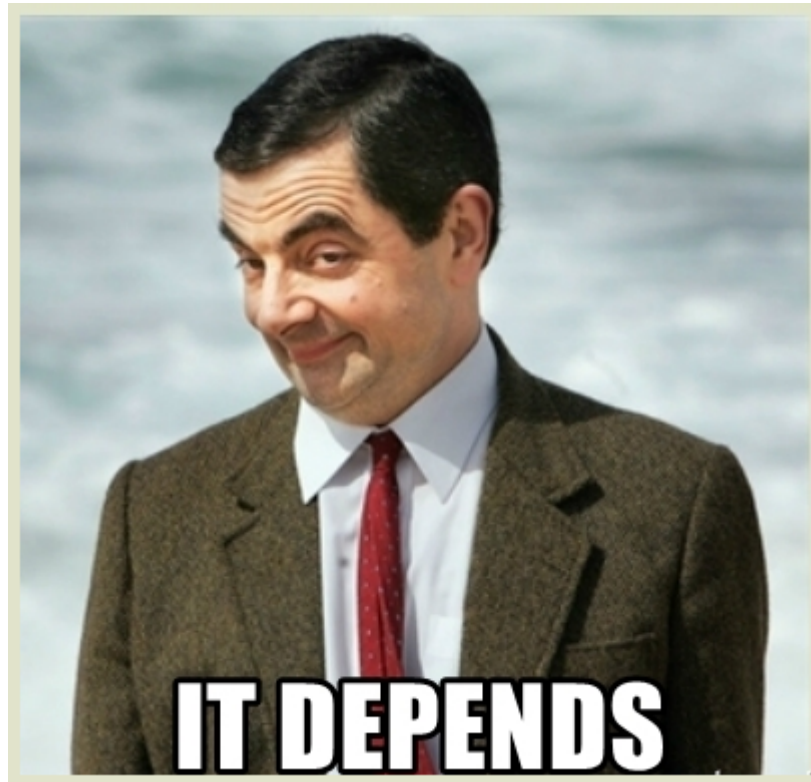
ESC1 Requirement 3: The certificate template defines EKUs that enable domain authentication.

The Certified Pre-Owned whitepaper explained how the certificate template must have one of the following EKUs (Extended Key Usages) in its **pkiExtendedKeyUsage** attribute:

- 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)

However, the **pKIExtendedKeyUsage** attribute is not the only attribute made for EKUs. There is another attribute named **msPKI-Certificate-Application-Policy**. So does that attribute enable authentication as well?



It is the EKUs that end up in the *Enhanced Key Usage* (2.5.29.37) property of the issued certificate that are the effective EKUs. Our testing showed that the schema version of the certificate template and whether the certificate has attributes **pKIExtendedKeyUsage** and **msPKI-Certificate-Application-Policy** configured with EKUs or not set (Null) affect the content of the Enhanced Key Usage (2.5.29.37) property:

msPKI-Template-Schema-Version	pKIExtendedKeyUsage	msPKI-Certificate-Application-Policy	Enhanced Key Usage (2.5.29.37)
1	EKU(s)	EKU(s)	pKIExtendedKeyUsage EKU(s)
1	EKU(s)	Null	pKIExtendedKeyUsage EKU(s)
1	Null	EKU(s)	msPKI-Certificate-Application-Policy EKU(s)
1	Null	Null	Null
2/3/4	EKU(s)	EKU(s)	msPKI-Certificate-Application-Policy EKU(s)
2/3/4	EKU(s)	Null	Null
2/3/4	Null	EKU(s)	msPKI-Certificate-Application-Policy EKU(s)
2/3/4	Null	Null	Null

msPKI-Certificate-Application-Policy takes precedence over **pKIExtendedKeyUsage** unless the certificate template is of schema version 1 and **pKIExtendedKeyUsage** contains EKUs.

EDIT (24 Oct 2024): The above described logic was confirmed with Client Authenticated ECU and Kerberos authentication. [Justin Bollinger](#) from TrustedSec proved the logic wrong for Schannel authentication with his [ESC15](#) discovery. I will update the logic in the article here and in BloodHound when Microsoft has revealed their patch for ECS15. Thanks to [JimmySu](#) for making me aware of this issue.

We do not expect anyone to remember this rule. To make life easier, we added a couple of extra properties to the CertTemplate node:

- (**effectiveekus**): The effective EKUs based on the logic above.
- (**authenticationenabled**): Whether the effective EKUs enable authentication.

We have the LDAP attributes represented with properties as well:

- (**schemaversion**): Reflects **msPKI-Template-Schema-Version** attribute.
- (**ekus**): Reflects **pKIExtendedKeyUsage** attribute.

- (**certificateapplicationpolicy**): Reflects **msPKI-Certificate-Application-Policy** attribute.

TESTV3@DUMPSTER.FIRE

Authentication Enabled:	TRUE
Authorized Signatures Required:	0
Certificate Application Policies:	1.3.6.1.5.5.7.3.2
Certificate Name Flags: ENROLLEE_SUPPLIES_SUBJECT, ENROLLEE_SUPPLIES_SUBJECT_ALT_NAME	
Created:	0
Distinguished Name: CN=TESTV3,CN=CERTIFICATE TEMPLATES,CN=PUBLIC KEY SERVICES,CN=SERVICES,CN=CONFIGURATION,DC=DUMPST...	
Domain FQDN:	DUMPSTER.FIRE
Domain SID:	S-1-5-21-2697957641-2271029196-387917394
Effective EKUs:	1.3.6.1.5.5.7.3.2
Enhanced Key Usage:	1.3.6.1.5.5.7.3.4
Enrollee Supplies Subject:	TRUE
Enrollment Flags:	PUBLISH_TO_DS
Last Collected by BloodHound: 2024-01-05 17:13 GMT+1 (GMT+0100)	
No Security Extension:	FALSE
OID: 1.3.6.1.4.1.311.21.8.4571196.1884641.3293620.10686285.1...	
Renewal Period:	6 weeks
Requires Manager Approval:	FALSE
Schema Version:	3

If you assessed certificate templates for ESCx opportunities based on **pKIExtendedKeyUsage** EKUs only, you may have encountered false positives or false negatives, but likely not. The **msPKI-Certificate-Application-Policy** attribute is Null by default for schema version 1 templates, and in version 2 and above, the **msPKI-Certificate-Application-Policy** and **pKIExtendedKeyUsage** attributes are identical by default. If you edit the EKUs through the Windows built-in Certificate Template Manager

GUI, it will set both `msPKI-Certificate-Application-Policy` and `pKIExtendedKeyUsage` to the chosen set of EKUs. That means `pKIExtendedKeyUsage` does hold the effective set of EKUs unless you tamper with the attributes in some other way.

To conclude the requirement, we add `authenticationenabled = True` to our Cypher query:

```
p = ()-[:|||]->(:) ct. = ct. = p
```

ESC1 Requirement 4: Manager approval is disabled.

If a certificate template's `msPKI-Enrollment-Flag` attribute contains the `CT_FLAG_PEND_ALL_REQUESTS` flag, then a CA administrator must manually approve certificate enrollment requests before the CA issues certificates. You can check if the certificate template requires manager approval for enrollment in BloodHound by the CertTemplate property `requiresmanagerapproval`, or under “*Requires Manager Approval*” in the entity panel.

Our Cypher query gets another line:

```
p = ()-[:|||]->(:) ct. = ct. = ct. = p
```

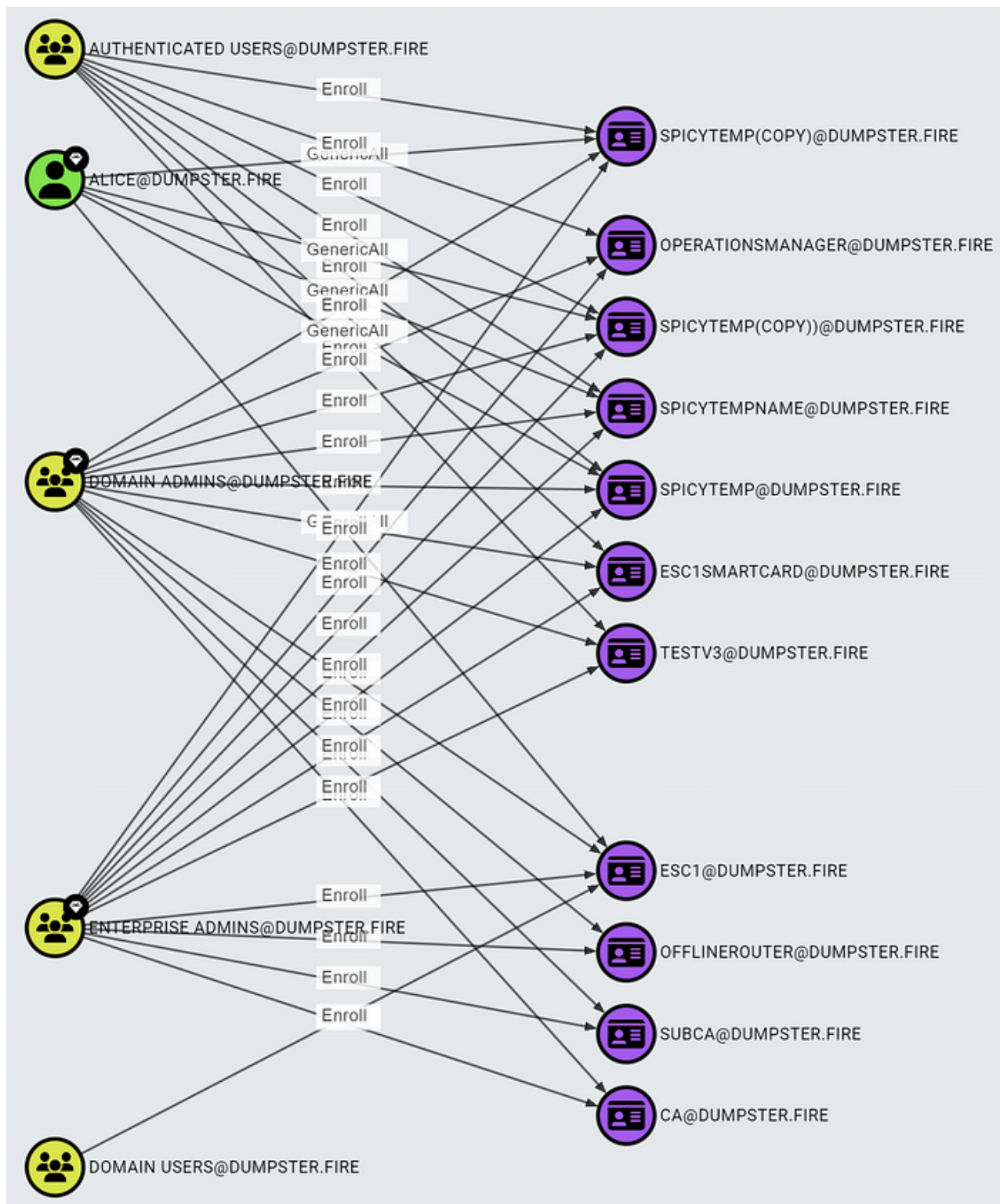
ESC1 Requirement 5: No authorized signatures are required.

The certificate template's `msPKI-RA-Signature` attribute holds the number of authorized signatures required for enrollment. CertTemplate property `authorizedsignatures` reflects the attribute, and you can view the value under “*Authorized Signatures Required*” in the entity panel.

Certificate templates of schema version 1 do not support the authorized signatures requirement. We therefore need to check that as well in our Cypher query:

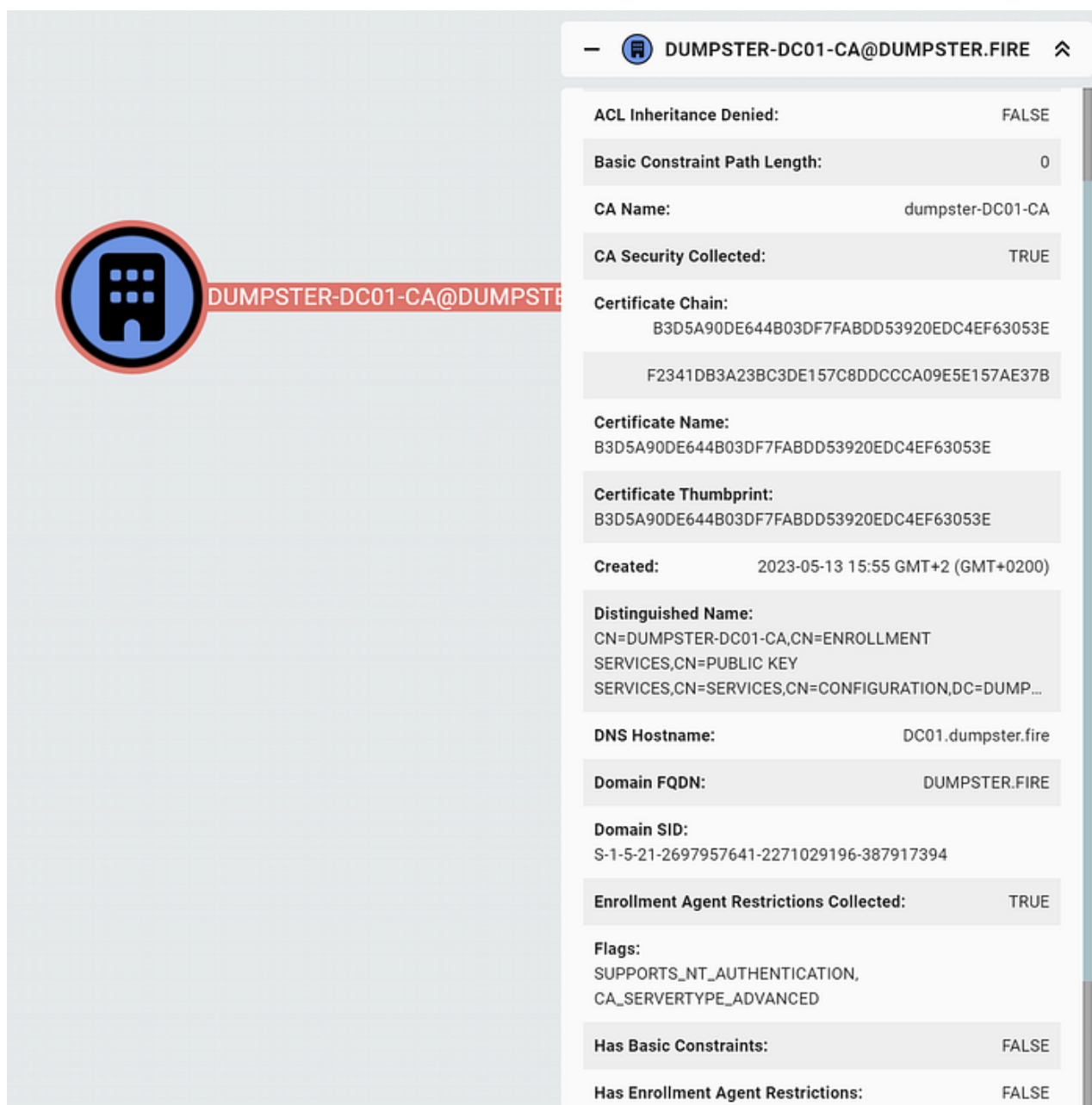
```
p = ()-[:|||]->(:) ct. = ct. = ct. = (ct. = ct. = ) p
```

Our query now finds all the certificate templates that meet the ESC1 requirements and the principals that have enrollment rights on them:



Enterprise CA Requirements

BloodHound now has an *EnterpriseCA* node type, representing the `pKIEnrollmentService`



The screenshot displays the SharpHound interface for an Enterprise Certificate Authority (CA). On the left, there is a circular icon with a building-like symbol. The main panel on the right lists various attributes of the CA object:

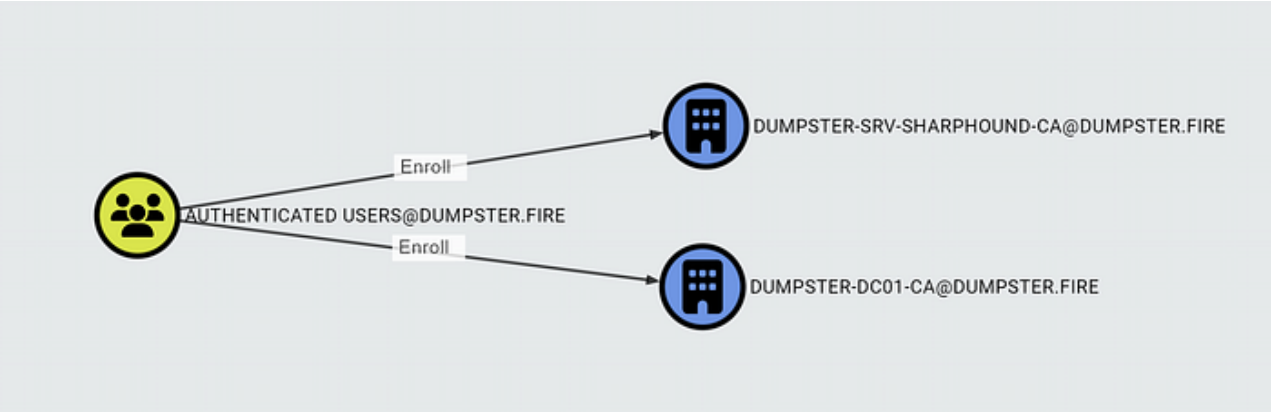
- ACL Inheritance Denied:** FALSE
- Basic Constraint Path Length:** 0
- CA Name:** dumpster-DC01-CA
- CA Security Collected:** TRUE
- Certificate Chain:**
 - B3D5A90DE644B03DF7FABDD53920EDC4EF63053E
 - F2341DB3A23BC3DE157C8DDCCCA09E5E157AE37B
- Certificate Name:** B3D5A90DE644B03DF7FABDD53920EDC4EF63053E
- Certificate Thumbprint:** B3D5A90DE644B03DF7FABDD53920EDC4EF63053E
- Created:** 2023-05-13 15:55 GMT+2 (GMT+0200)
- Distinguished Name:** CN=DUMPSTER-DC01-CA,CN=ENROLLMENT SERVICES,CN=PUBLIC KEY SERVICES,CN=SERVICES,CN=CONFIGURATION,DC=DUMP...
- DNS Hostname:** DC01.dumpster.fire
- Domain FQDN:** DUMPSTER.FIRE
- Domain SID:** S-1-5-21-2697957641-2271029196-387917394
- Enrollment Agent Restrictions Collected:** TRUE
- Flags:** SUPPORTS_NT_AUTHENTICATION, CA_SERVERTYPE_ADVANCED
- Has Basic Constraints:** FALSE
- Has Enrollment Agent Restrictions:** FALSE

ESC1 Requirement 6: The Enterprise CA grants low-privileged users enrollment rights.

A principal must have the *Enroll* permission on the Enterprise CA to enroll for any published certificate template of the Enterprise CA. The Enterprise CA stores the permissions in registry on the computer hosting the Enterprise CA service, but you can also find the permissions in the DACL of the Enterprise CA AD object. However, the DC will not send changes made in the AD DACL to the Enterprise CA host, and the effective enrollment rights remain unchanged. SharpHound will attempt to collect both the AD DACL and the registry version and let the registry version overrule if SharpHound collects both successfully.

The non-traversable Enroll edge, already introduced under the CertTemplate node, implements the Enroll permission. We can find all principals with enrollment rights on Enterprise CAs with the Cypher query:

p = ()-[[:]->(:) p




ESC1 Requirement 7: The Enterprise CA is trusted for NT authentication.

When authenticating in AD using a certificate, the DC will check if it trusts the issuer of the certificate, the Enterprise CA, for NT authentication. It does that by checking if the issuer certificate is present in its local *enterprise NTAAuth store*, cached in registry under `HKLM\SOFTWARE\Microsoft\EnterpriseCertificates\NTAuth\Certificates`.

ADCS has a central NTAAuth store. It is the object named *NTAuthCertificates* in the ADCS PKI container. It holds the certificates trusted for NT authentication in its `caCertificate` attribute. DCs of the AD forest replicate the values of this attribute to their local enterprise NTAAuth stores using group policy.

To represent the NTAAuth store in ADCS, we introduce the new *NTAuthStore* node:

NTAUTHCERTIFICATES@DUMPSTER.FIRE

NTAUTHCERTIFICATES@DUMPSTER.F...

Object Information

Object ID: 2F9F3630-F46A-49BF-B186-6629994EBCF9

ACL Inheritance Denied: FALSE

Certificate Thumbprints:
B3D5A90DE644B03DF7FABDD53920EDC4EF63053E
F2341DB3A23BC3DE157C8DDCCCA09E5E157AE37B

Created: 2023-05-13 15:55 GMT+2 (GMT+0200)

Distinguished Name:
CN=NTAUTHCERTIFICATES,CN=PUBLIC KEY
SERVICES,CN=SERVICES,CN=CONFIGURATION,DC=DUMP...

Domain FQDN: DUMPSTER.FIRE

Domain SID:
S-1-5-21-2697957641-2271029196-387917394

Last Collected by BloodHound:
2024-01-22 13:03 GMT+1 (GMT+0100)

+ Inbound Object Control

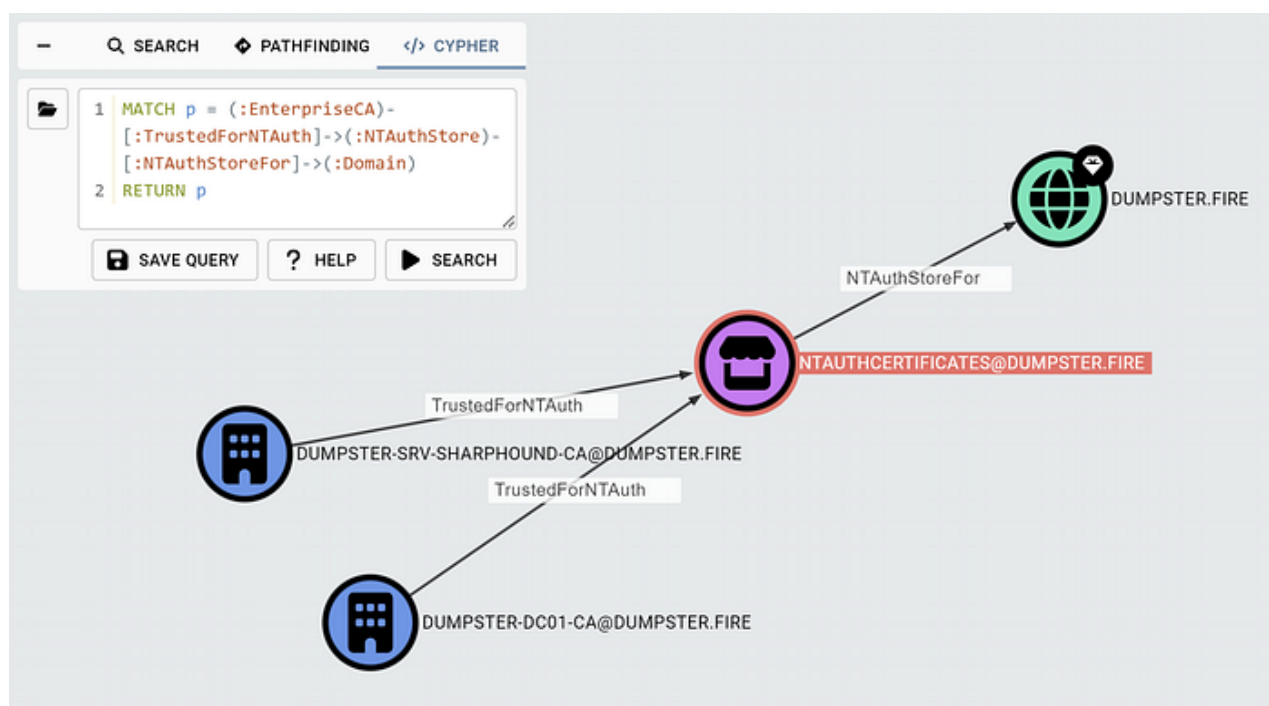
24

The “*Certificate Thumbprints*” entry in the entity panel holds the value of the node property `certthumbprints`, which represents the thumbprints of the certificates in the `caCertificate` attribute of the NTAuth store. Likewise, the EnterpriseCA node has a property, `certthumbprint`, which is the thumbprint of the certificate stored in the `caCertificate` attribute of the Enterprise CA AD object.

To make it easy to check the Enterprise CAs trusted for NT authentication, we have added a non-traversable edge called *TrustedForNTAuth*, going from EnterpriseCA nodes to NTAuthStore nodes. BloodHound creates the edge if the EnterpriseCA’s `certthumbprint` property value is present in the NTAuthStore’s `certthumbprints` property. As you might have multiple forests in your BloodHound graph, we have introduced another non-traversable edge, *NTAuthStoreFor*, going from the NTAuthStore node to the forest root domain the NTAuth store belongs to.

You can check the Enterprise CAs trusted for NT authentication with the following Cypher query:

```
p = (:) -[:]->(:) -[:]->(:) p
```




ESC1 Requirement 8: The Enterprise CA’s certificate chain is trusted.

It is not enough for the DC, that it trusts the issuer Enterprise CA for NT authentication. It will also check that it trusts all the certificates in the *certificate chain*. The certificate chain includes the certificate of the issuer CA, followed by the certificate of the CA which issued and signed the issuer CA’s certificate, and so on until it ends at a self-signed Root CA certificate.

The DC checks the certificates of the chain against its local *intermediate CA store*. Group policy automatically populates the store with the certificates of the AD NTAuth store, the certificates of the CA objects under the *A/A* container, and the certificates of the

Enterprise CAs (located under the *Enrollment Services* container). Additionally, the DC verifies that it has the Root CA certificate in its local *trusted Root CA store*, which has the certificates of the Root CA AD objects of AD CS located in the *Certification Authorities* container.

To support certificate chain verification in BloodHound, we have introduced the *AIACA* and the *RootCA* node types:



MYFAKECA@DUMPSTER.FIRE

MYFAKECA@DUMPSTER.FIRE

Object Information

Object ID: FE6E70A4-315A-437F-AC45-C441EAB59142

ACL Inheritance Denied: FALSE

Basic Constraint Path Length: 0

Certificate Chain: C8E45A32D12EE4307F899F89373139E4FB434E7D

Certificate Name: C8E45A32D12EE4307F899F89373139E4FB434E7D

Certificate Thumbprint: C8E45A32D12EE4307F899F89373139E4FB434E7D

Created: 2023-09-29 00:50 GMT+2 (GMT+0200)

Distinguished Name: CN=MYFAKECA,CN=AIA,CN=PUBLIC KEY SERVICES,CN=SERVICES,CN=CONFIGURATION,DC=DUMP...

Domain FQDN: DUMPSTER.FIRE


Domain SID: S-1-5-21-2697957641-2271029196-387917394

Has Basic Constraints: FALSE

Has Cross Certificate Pair: FALSE

Last Collected by BloodHound: 2024-01-22 13:03 GMT+1 (GMT+0100)

AIACA node



DUMPSTER-DC01-CA@DUMPSTER.FIRE

DUMPSTER-DC01-CA@DUMPSTER.FIRE

Object Information

Object ID: FA4FB111-ADDB-4320-951C-39F909ED2B1D

ACL Inheritance Denied: FALSE

Basic Constraint Path Length: 0

Certificate Chain: F2341DB3A23BC3DE157C8DDCCCA09E5E157AE37B

Certificate Name: F2341DB3A23BC3DE157C8DDCCCA09E5E157AE37B

Certificate Thumbprint: F2341DB3A23BC3DE157C8DDCCCA09E5E157AE37B

Created: 2023-05-13 15:55 GMT+2 (GMT+0200)

Distinguished Name: CN=DUMPSTER-DC01-CA,CN=CERTIFICATION AUTHORITIES,CN=PUBLIC KEY SERVICES,CN=SERVICES,CN=CONFIGURATION,DC=DUMP...

Domain FQDN: DUMPSTER.FIRE

Domain SID: S-1-5-21-2697957641-2271029196-387917394

Has Basic Constraints: FALSE

Last Collected by BloodHound: 2024-01-22 13:03 GMT+1 (GMT+0100)

RootCA node

The AIACA, RootCA, and EnterpriseCA nodes all have a `certthumbprint` property containing the certificate thumbprint and a `certchain` property that holds a list of thumbprints of certificates in the certificate chain. To make it easy to check the certificate chain and show the PKI hierarchy, we have introduced the new non-traversable edge *IssuedSignedBy*, which goes from EnterpriseCA nodes to the next EnterpriseCA/RootCA in the certificate chain.

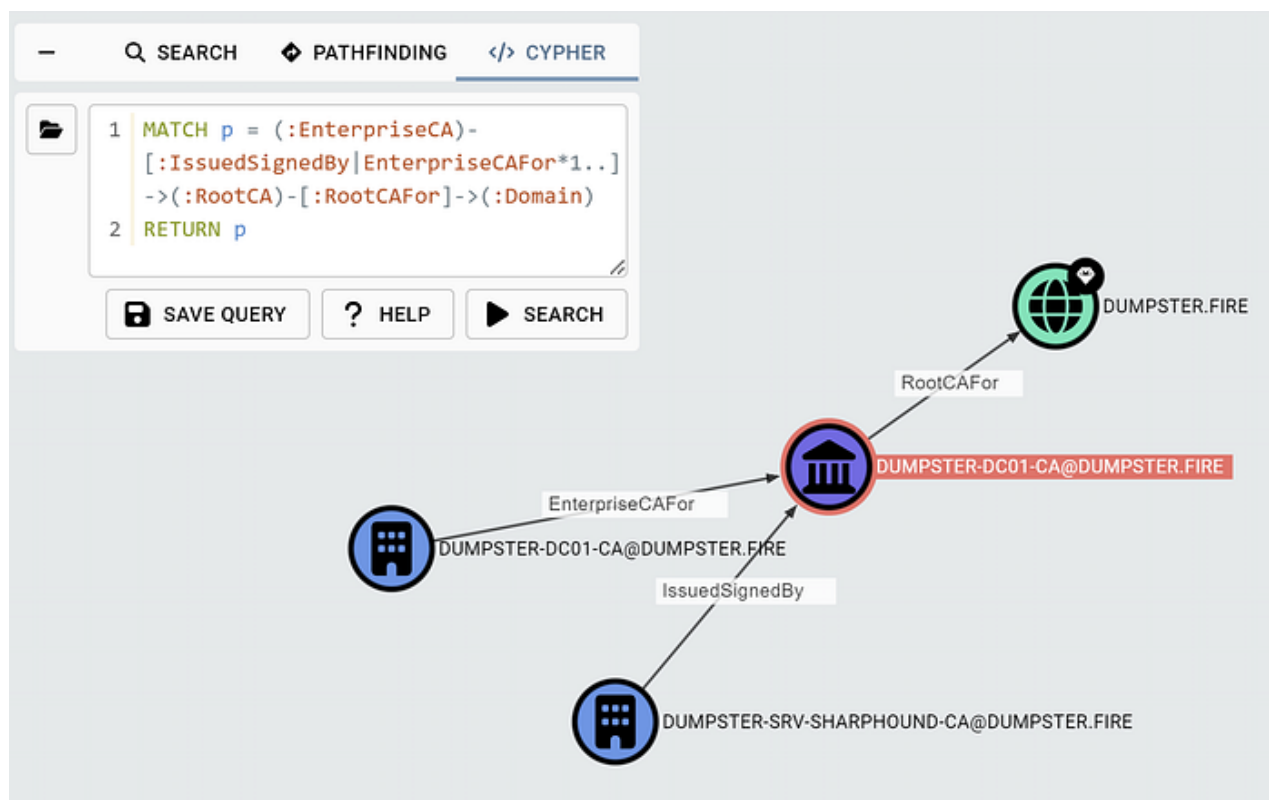
When you create an Enterprise or Root CA in ADCS, you will automatically also get and CA object in the AIA container representing the same CA. So AIACA nodes most commonly represent a CA which is also represented by an EnterpriseCA node or a RootCA node, to our understanding. For that reason, we do not create IssuedSignedBy edges to/from AIACA nodes, at least for now.

It is possible to install the CA enrollment service on a Root CA, such that the Root CA is also an Enterprise CA. This configuration exists if the organization has a single-tier CA hierarchy for example. In that case, we link the EnterpriseCA node to the corresponding RootCA node with an *EnterpriseCAFor* edge.

Additionally, we create a non-traversable edge, *RootCAFor*, going from the RootCA node to the forest root domain of the Root CA.

To check that DCs of the forest trust an Enterprise CA's certificate chain, you can run the Cypher query:

```
p = (:) -[:|*.] -> (:) -[:] -> (:) p
```

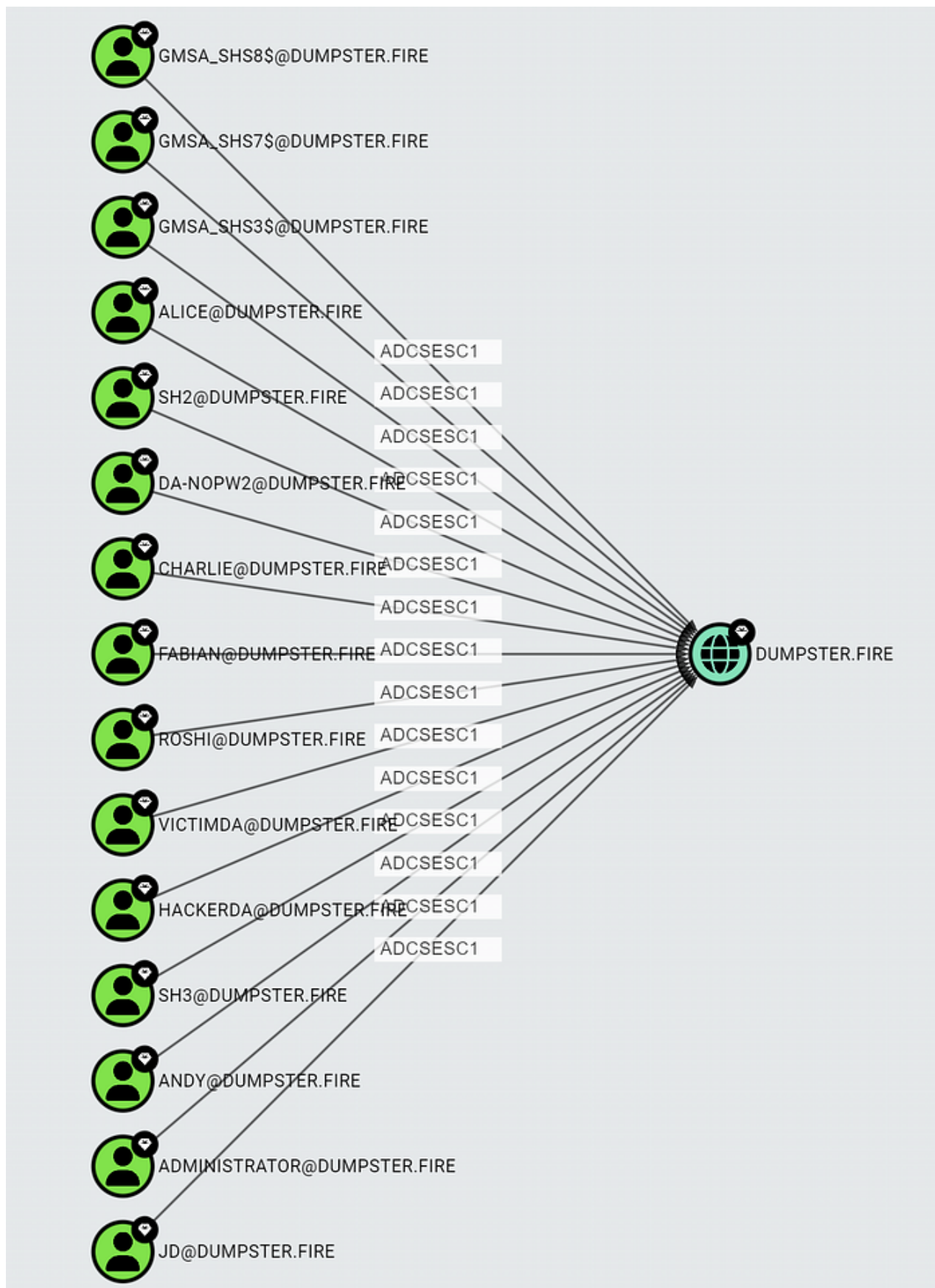


ESC1 Requirement 9: The Enterprise CA has the certificate template published.

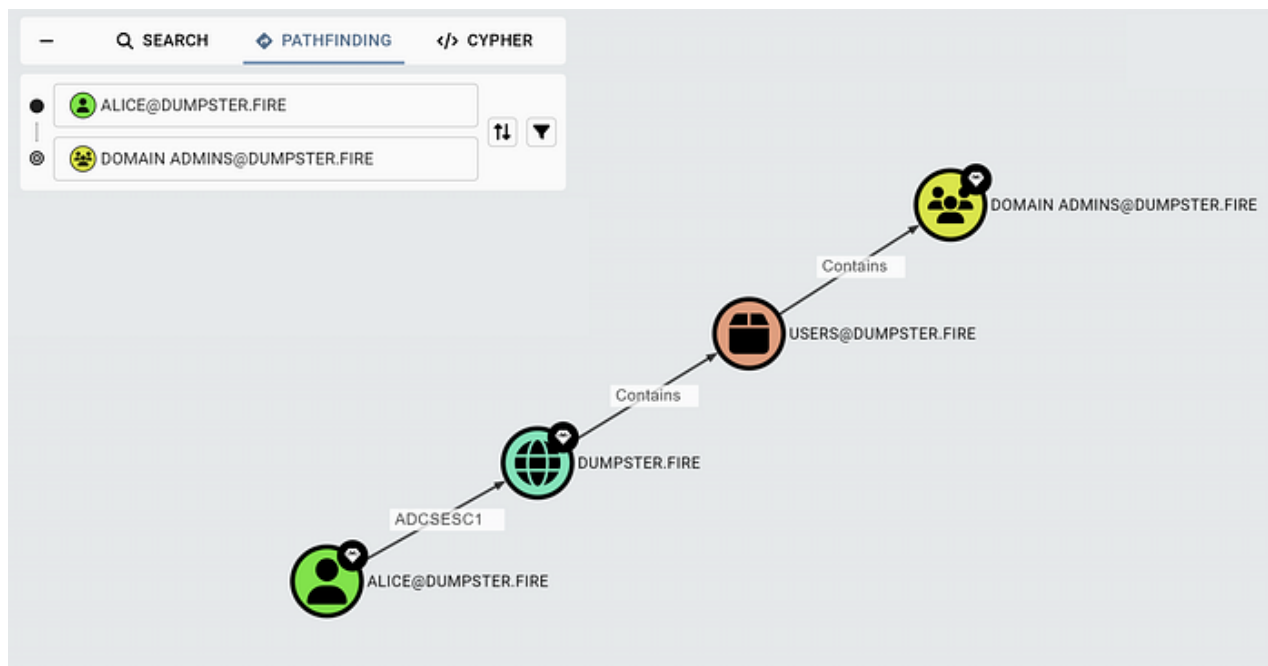
The Enterprise CA AD object has an attribute named `certificateTemplates`, which is the name of the certificate templates the Enterprise CA has published. We use that in BloodHound to generate non-traversable *PublishedTo* edges from CertTemplate nodes to EnterpriseCA nodes.

You can check what CertTemplates are published to which EnterpriseCAs using the following Cypher query:

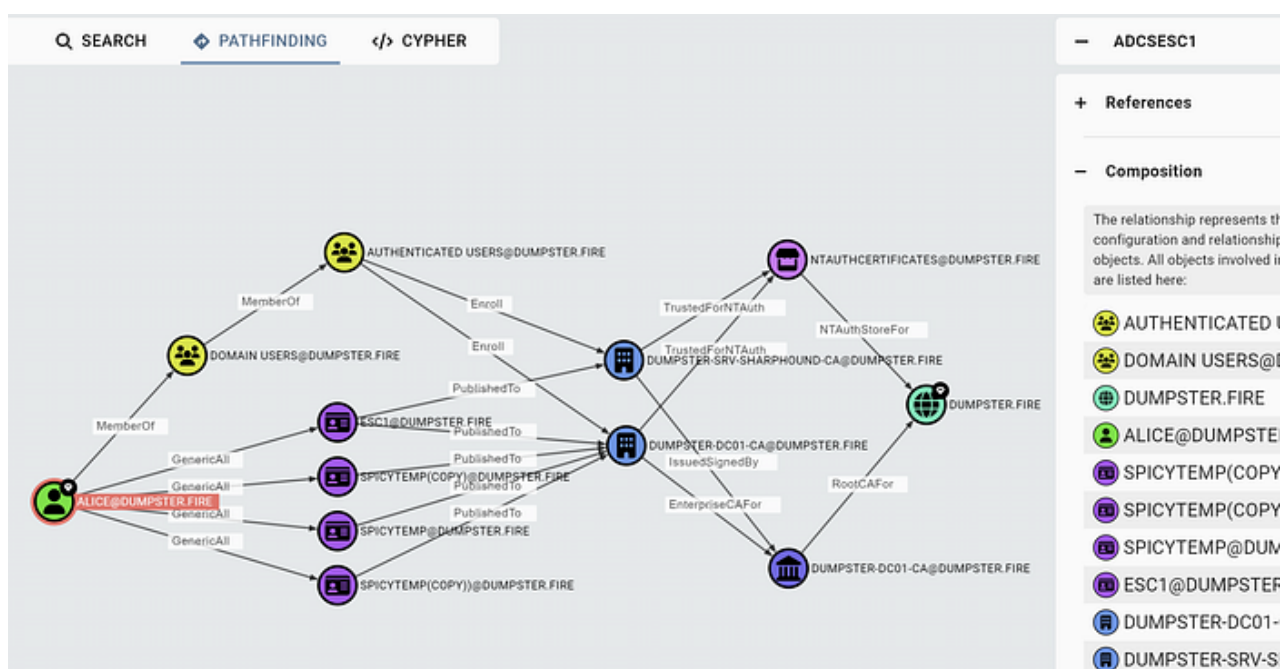
```
p = (:) -[:] -> (:) p
```

The edge ends at the forest root domain, as ESC1 enables the principal to authenticate as any principal in the forest. The edge is traversable, so ADCSESC1 will appear in pathfinding search if it is part of the shortest path to the target:



To know that ESC1 is possible is great, but you also need to know the details of how and why it is possible, both as a red teamer and a blue teamer. To get that info, click on the edge, and click on *Composition*:



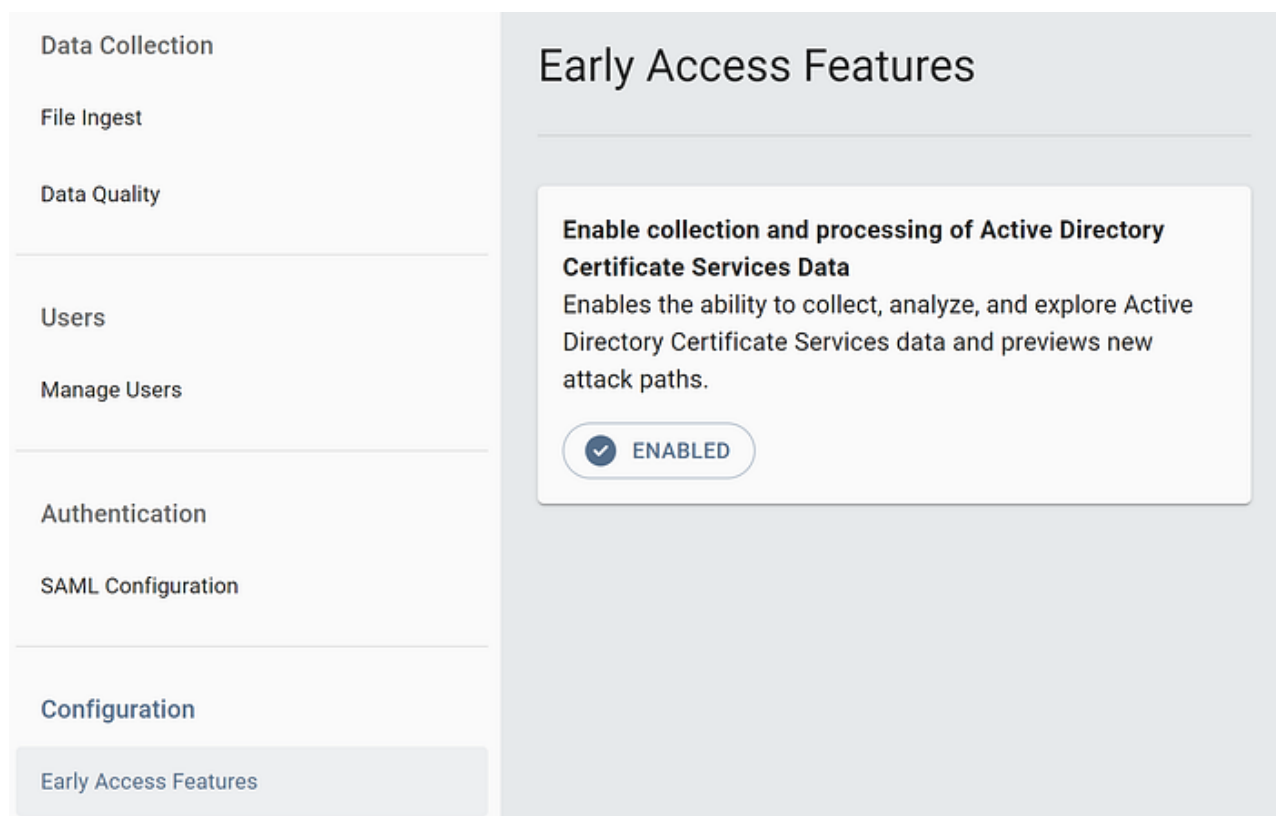
The composition will show you all the nodes and edges involved in meeting the requirements for the principal to have the ADCSESC1 edge. It might show you multiple options, like in the screenshot above, where Alice has four certificate templates and two Enterprise CAs to choose from if she wants to perform an ESC1 attack. The abuse sections in the entity panel provide info about how the attacker can execute the attack, as usual.

New Data Collection

The AD CS AD objects exist in the *Configuration* NC (Naming Context). This partition is forest-wide and replicated to all writeable DCs in the AD forest. SharpHound has traditionally only collected data from the *Default* NC, but we have extended SharpHound to collect the AD CS containers and objects in the Configuration NC when using the collection method **CertServices**, included in **Default**, **DCOnly**, and **All**. There are no special requirements for the Configuration NC collection, you simply need to collect from a single DC in the forest. SharpHound will collect the AD CS objects over LDAP, just like the regular AD objects from the Default NC.

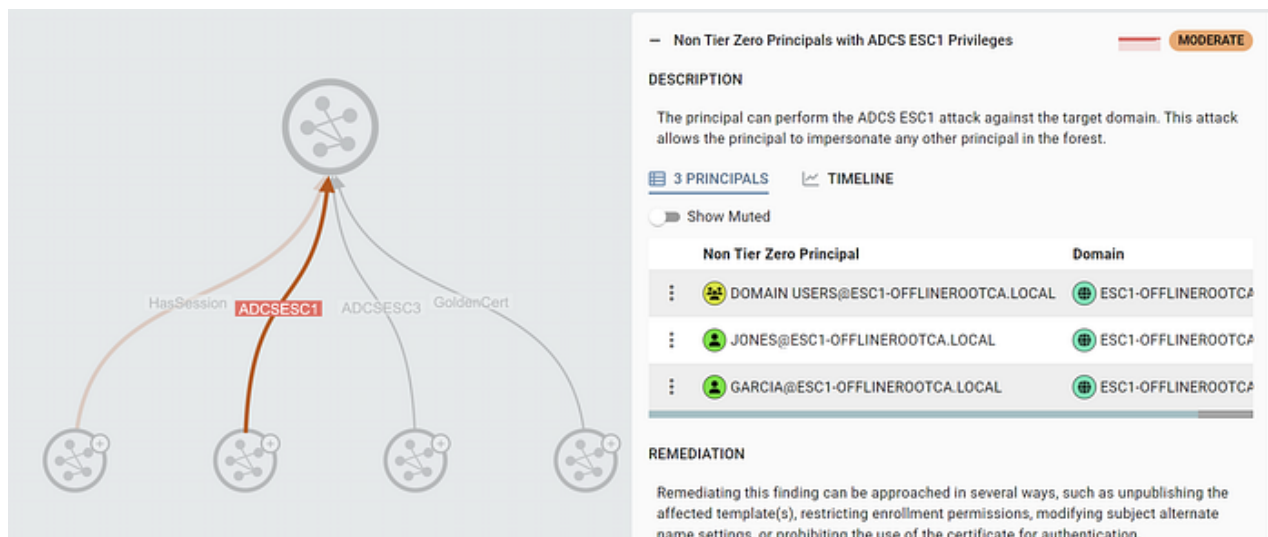
SharpHound v2.3 (or later) supports the AD CS collection. We highly recommend using the latest version, which you can download in BloodHound CE or from the [SharpHound GitHub repo](#). Make sure you are also using the latest BloodHound CE version. You can upgrade on Linux using this cmdlet: `docker compose pull && docker compose up -d` or on Windows using this PowerShell cmdlet: `docker compose pull; docker compose up -d`. If you have not installed BloodHound CE yet, follow the instructions here: [Install BloodHound Community Edition with Docker Compose](#).

You can enable the AD CS early access feature in BloodHound in *Administration* under *Early Access Features*:



AD CS in BloodHound Enterprise

BloodHound Enterprise customers will automatically see findings in the Attack Paths tab if principals not marked as Tier Zero have the ADCSESC1 edge:



The exposure percentage of the finding determines the finding severity, as for other graph-based findings.

Since ESC1 has multiple requirements that need to be met, there are also multiple ways to remediate the finding. Our recommended remediation will therefore suggest multiple options, all described in detail, and guide towards the best remediation based on the current usage of certificate templates involved:

Non Tier Zero Principals with ADCS ESC1 Privileges

Recommended Remediation

We advise gaining a clear understanding of the intended use of the certificate templates to determine the most suitable remediation approach. This can be achieved through an evaluation of existing certificates and authentication logs, as outlined in the [Certified Pre-Owned ADCS whitepaper](#) sections:

- Monitor User/Machine Certificate Enrollments - DETECT1
- Monitor Certificate Authentication Events - DETECT2

Collaborate with the individual responsible for ADCS within the organization to address the following questions pertaining to the identified certificate templates. This process will help in considering the appropriate checks and remedial actions described below:

1. Is the certificate template in use?

Check: Latest issued certificates and expiration dates.

Remediation: *Unpublish (disable) certificate template*

2. Which principals are enrolling in this template?

Check: Requester principals of issued certificates.

Remediation: *Remove Enroll permission (restrict to Tier Zero)*

3. Is the Subject Alternative Name (SAN) flag required?

Check: If the requester name and the SAN refer to the same principal in issued certificates.

Remediation: *Remove SAN flag*

4. Could the current setup be replaced with an enrollment agent setup?

Check: If it is feasible that a service account or group of employees in the IT department (potentially non-Tier Zero principals) performs the enrollment on behalf of the users that need the certificate.

Remediation: *Implement enrollment agent*

5. Does the certificate template need to allow for authentication?

Check: Login events using certificates created with the certificate template.

Remediation: *Remove EKV that enables authentication*

6. Could future certificate requests wait for a manual approval?

Check: If it is feasible that the certificate request has to wait for a Tier Zero principal to manually approve the request.

Remediation: *Enable manager approval*

What is Next?

As you can tell from the title of this blog post, this is the first of a series on ADCS attack paths in BloodHound. So stay tuned, as we will dive into some of the more advanced ADCS escalations and how you can identify them using BloodHound.

We are very eager to get your feedback. Please join us in the [BloodHound Slack](#) or report any issues on the [BloodHound GitHub repo](#).