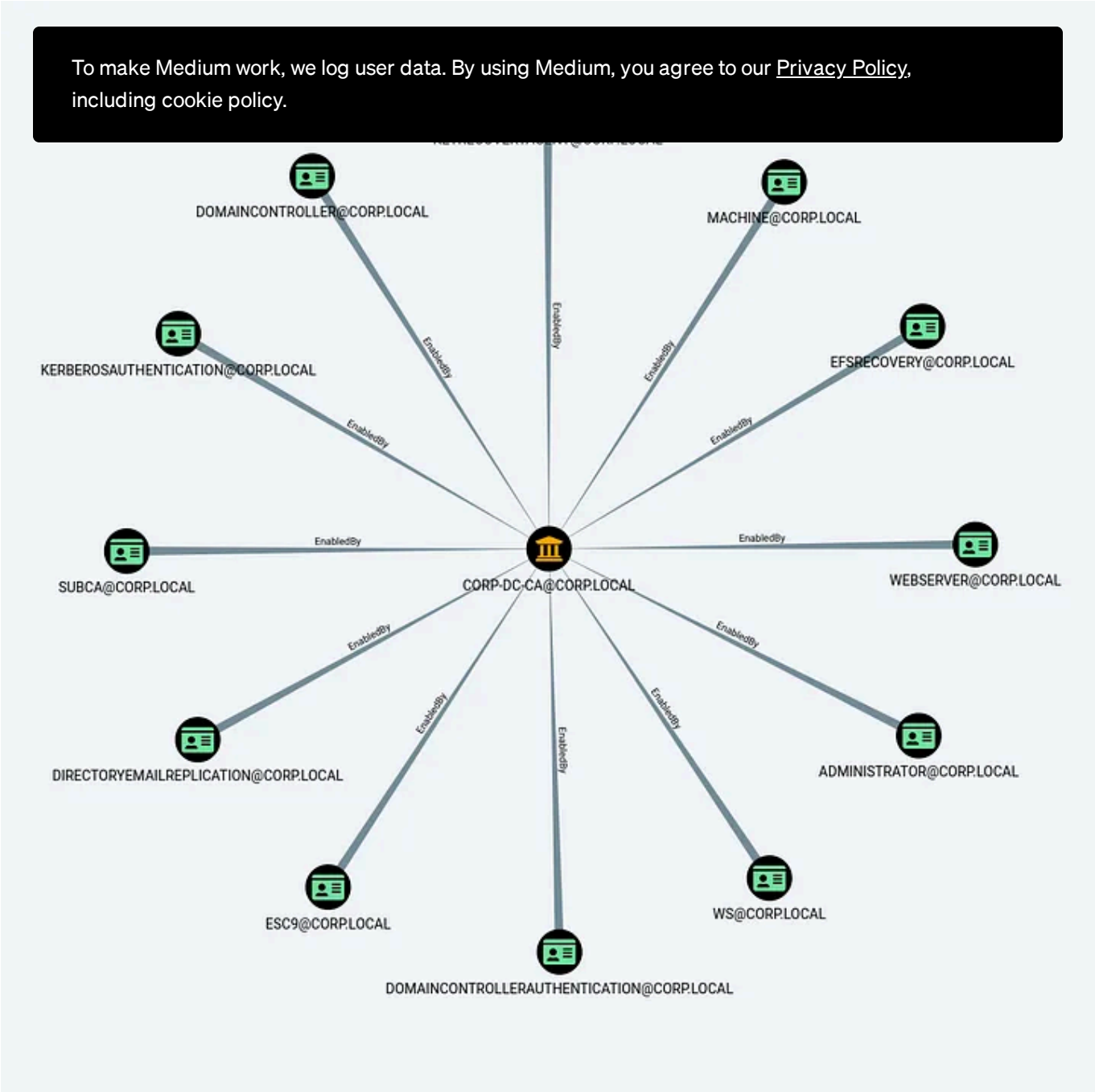
Certipy 4.0: ESC9 & ESC10, BloodHound GUI, New Authentication and Request Methods — and more!

Oliver Lyak · Follow
Published in IFCR · 20 min read · Aug 4, 2022

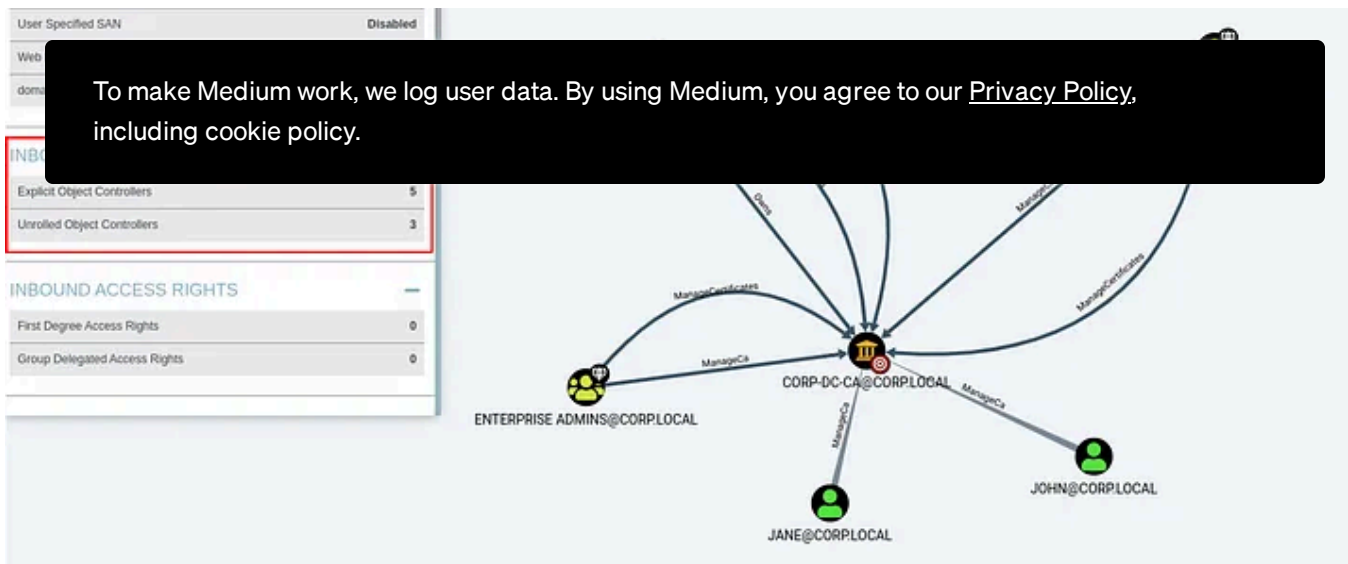
A new version of Certipy has been released along with a forked BloodHound GUI that has PKI support! In this blog post, we will look at some of the major new features of Certipy, which includes LDAPS (Schannel) and SSPI authentication, new request options and methods, and of course support for the forked BloodHound GUI that I changed to have new nodes, edges, and prebuilt queries for AD CS. At the end of the blog post, we will also look at the two new privilege escalation techniques for AD CS: ESC9 and ESC10.

BloodHound x Certipy

The BloodHound team has delivered many impressive updates, and according to their release post on version 4.1 and version 4.2, Active Directory Certificate Services (AD CS) abuse primitives are on their road map



This graph was drawn by simply selecting the CA node and then clicking on "See Enabled Templates", as shown below.



The same is possible for certificate templates. Simply select the template and click “See Certificate Authorities”.

Want to see enrollment rights or object controllers? Also one click away.

Even though the forked BloodHound GUI was mainly focused on PKI integration, I decided to add a few features that I personally like. For instance, you can now hover your mouse over a query and click the little “Copy” button to copy the query to your clipboard.

from the upstream version so you don't miss out on those features. If you ha

Old Is New Again

Now, back to Certipy. I have reintroduced and improved some old features of Certipy that I previously removed related to Certipy's `find` command. For text and JSON based output, Certipy will now check for ESC1, ESC2, ESC3, ESC4, and the new ESC9 on certificate templates, and ESC6, ESC7, and ESC8 on certificate authorities based on the current user's nested group memberships. Furthermore, if `ms-DS-MachineAccountQuota` is not `0` (default: `10`) then Certipy will act as if the current user is also a member of the `Domain Computers` group, since the user will most likely be able to add a new domain computer. In addition to this, the `find` command now accepts the `-vulnerable` parameter to only show vulnerable certificate templates, and `-hide-admins` to hide administrators from the permissions for a cleaner output. These options only apply to text and JSON based output (`-text` and `-json`) and does not affect the BloodHound data.

New Authentication Methods

Scannel (LDAPS)

Our good friends at FalconForce recently published a blog post on how to detect "UnPACing" — the technique used by Certipy and Rubeus during PKINIT Kerberos authentication to retrieve the NT hash. In the Certified Pre-Owned whitepaper, the authors, Will Schroeder and Lee Christensen, mention that Active Directory supports certificate authentication over two protocols by default: Kerberos and Secure Channel (Schannel). One protocol that supports client authentication via Schannel is LDAPS (LDAP over SSL/TLS) — assuming AD CS has been setup. As such, this is exactly what I've implemented into Certipy.

Once you've obtained your shiny new certificate, run the `auth` command like you'd usually do, but this time, specify the `-ldap-shell` option to drop into an interactive LDAP shell with a limited set of commands that should be

certificate during the StartTLS upgrade. It is worth noting that the type of ce
re

This new feature is also relevant for ESC10 (see later in the post).

Windows Integrated Authentication (SSPI)

Now, imagine you just got code execution on a domain-joined machine. You *could* run your C2 agent, open a SOCKS proxy connection, and then run Certipy through that. The problem in this scenario is that you don't know the credentials of your current user context. This has happened to me a few times. Instead, let me introduce Certipy's new SSPI integration.

The first step is to get Certipy on your target machine. You could install Python and then Certipy, or you could just use something like PyInstaller (`pyinstaller ./Certipy.spec`) to pack it into an executable. Once you've done that, you can run all your usual commands, but instead of specifying username, password, and domain, you can just pass the `-sspi` option. This will make Certipy use your current user's domain context for authentication by using Windows APIs to retrieve Kerberos tickets.

The same thing can be achieved by using `-print` with the `auth` command, and then passing the Base64 ticket to Certipy's new `ptt` command in the `-`

The new `ptt` command can be used to inject tickets from a file or command line, but it can also be used to request a new TGT using credentials and

Some users of Certipy reported that during an engagement, they could only re[redacted]. Currently, it supports both HTTP and HTTPS, but only with password or NTLM authentication. To request a certificate through the web interface, simply pass the `-web` option to your usual `req` command.

Double SAN

A feature request was sent to me to allow specifying a DNS host name instead of a UPN for the old `-alt` parameter. As such, the `-alt` parameter has been removed in favor of the two new parameters `-upn` and `-dns`. And it turns out that you can even specify both parameters in a single request.

Certipy will now print out all the account identifications found in the certificate. Now, what would happen if we tried to authenticate with this

Yes, we can. As shown above, two different NT hashes were returned depending on the identification used. It is of course also possible to only specify a single identification.

Key Archival and Key Size

A user reported that a template had a different minimum key size than the one that was generated by Certipy. This will yield the error `CERTSRV_E_KEY_LENGTH`. Certipy now accepts the `-key-size` parameter to specify a different key size, as shown below.

This is a whole different type of request and protocol, that includes retrieving the CA Exchange Certificate, crafting undocumented ASN1 structures, encrypting the private key, and a few more headaches. Nonetheless, I wouldn't want this single flag to stand in my (or your) way to becoming domain administrator during an engagement.

Other Features

You might also encounter some other unmentioned features — which might not seem that useful — that is merely a result of my own research. For instance, it's possible to renew a certificate using an old certificate with the `-renew` parameter. Since I had already implemented all the structures and functionality, I thought I'd just add it to Certipy.

New Escalations

To understand the new escalations, we must first understand Microsoft's patch for CVE-2022–26923.

My previously reported AD privilege escalation vulnerability "Certifried" (CVE-2022–26923) actually contained four different cases. The case described in my previous blog post was that it was possible to simply duplicate the DNS host name of a machine account. This would work from a

`usorPrincipalName` matches the UPN in the certificate, it will try to find an ac

las g the certificate request, it will build the UPN in the certificate based on the `sAMAccountName`. On top of this, if the KDC cannot find an account where the `sAMAccountName` matches the UPN in the certificate, it will simply add a `$` at the end and try again, as we know from CVE-2021–42287/CVE-2021–42278.

So how did Microsoft fix this? First of all, they made sure that the "Validated write to DNS host name" permission on a machine account now only accepts a value that matches the `sAMAccountName` property. This means that it is still possible to duplicate the DNS host name of a domain controller (or another machine account) if you have `GenericWrite` over a machine account, as shown below.

This was tested against a fully patched domain controller where `john` only had `GenericWrite` over `johnpc$`.

On top of this, Microsoft implemented the new `szOID_NTDS_CA_SECURITY_EXT` security extension for issued certificates, which will embed the `objectSid` property of the requester. Furthermore, Microsoft created the new registry key values (`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Schannel`) `CertificateMappingMethods` and

As mentioned earlier, Active Directory supports certificate authentication ov[redacted] su[redacted] `CertificateMappingMethods` correspond to Kerberos and Schannel, respectively.

Certificates can either be mapped via implicit or explicit mappings. For explicit mappings, the `altSecurityIdentities` property on an account object is configured to contain identifiers for a certificate, for instance the issuer and serial number. This way, when a certificate is used for authentication via explicit mapping, it must be signed by a trusted CA and then match the values specified in the `altSecurityIdentities`. On the other hand, when a certificate is used for authentication via implicit mapping, then the information from the certificate's Subject Alternative Name (SAN) extension is used to map the certificate to an account, either the UPN or DNS field.

However, Schannel and Kerberos don't use the same techniques for mapping a certificate implicitly. Let's take a look at how a certificate is mapped implicitly for each protocol.

Kerberos Certificate Mapping

The new registry key value (`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Kdc`) `StrongCertificateBindingEnforcement` is by default set to `1` now. Before the patch, this key did not exist, but the old value was `0`, i.e. strong certificate binding was not enforced. This value can either be set to `0`, `1`, or `2`.

Microsoft is planning on setting this value to `2` by default on May 9, 2023 an

str

used for authentication via Kerberos.

So, let's say that the value is set to `0`; how is a certificate then implicitly mapped? For this blog post, we are not interested in explicit mapping (`altSecurityIdentities`). When a certificate is used for authentication via Kerberos, the KDC will first verify that it is issued by a trusted CA and that the certificate can be used for client authentication. For implicit mappings, the KDC will then try to map the certificate to an account either via the UPN or DNS SAN value.

If the certificate contains a UPN with the value `john@corp.local`, the KDC will first try to see if there exists a user with a `userPrincipalName` property value that matches. If not, it checks if the domain part `corp.local` matches the Active Directory domain. If there is no domain part in the UPN SAN, i.e. the UPN is just `john`, then no validation is performed. Next, it will try to map the user part `john` to an account where the `sAMAccountName` property matches. If this also fails, it will try to add a `$` to the end of the user part, i.e. `john$`, and try the previous step again (`sAMAccountName`). This means that a certificate with a UPN value can actually be mapped to a machine account.

If the certificate contains a DNS SAN and not a UPN SAN, then the KDC will split the DNS name into a user part and a domain part, i.e. `johnpc.corp.local` becomes `johnpc` and `corp.local`. The domain part is then validated to match the Active Directory domain, and the user part will be appended by a `$` and then mapped to an account where the `sAMAccountName` property matches, i.e.

bit set. The new default value is `0x18` (`0x8` and `0x10`), whereas the old value wa

- `0x0001` — Subject/Issuer certificate mapping (explicit)
- `0x0002` — Issuer certificate mapping (explicit)
- `0x0004` —SAN certificate mapping (implicit)
- `0x0008` — S4U2Self certificate mapping (Kerberos)
- `0x0010` — S4U2Self explicit certificate mapping (Kerberos)

So, Schannel actually doesn't support the new `szOID_NTDS_CA_SECURITY_EXT` extension directly. Instead, it will use S4U2Self to map the certificate via Kerberos, which then supports the `szOID_NTDS_CA_SECURITY_EXT` extension. However, this is performed as the last step if the other supported mappings fail. This means that if certificate contains a UPN or DNS name, and the `CertificateMappingMethods` contains the `0x4` value, then the `szOID_NTDS_CA_SECURITY_EXT` certificate extension and `StrongCertificateBindingEnforcement` registry value will have absolutely no influence on the certificate mapping via Schannel. This is a bit more interesting to us, since Microsoft officially suggested setting this registry key value to the old value `0x1f` (all of the above methods) as an alternative to manually mapping all certificates if the security updates caused authentication issues: "If you experience authentication failures with Schannel-based server applications, we suggest that you perform a test. Add or modify the **CertificateMappingMethods** registry key value on the domain controller and set it to 0x1F and see if that addresses the issue."

Now that we understand the patch for CVE-2022–26923, let's look at the new

- Certificate contains the `CT_FLAG_NO_SECURITY_EXTENSION` flag in the `msPKI-`

- Certificate specifies any client authentication EKU

Abuse

Please see the "Examples" section for a practical example. To abuse this misconfiguration, the attacker needs `GenericWrite` over any account A that is allowed to enroll in the certificate template to compromise account B (target).

ESC10 — Weak Certificate Mappings

Description

ESC10 refers to two registry key values on the domain controller.

`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Schannel` `CertificateMappingMethods`. Default value `0x18` (`0x8 | 0x10`), previously `0x1F`.

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Kdc` `StrongCertificateBindingEnforcement`. Default value `1`, previously `0`.

Case 1

`StrongCertificateBindingEnforcement` set to `0`

Case 2

- `StrongCertificateBindingEnforcement` set to `1` (default) or `0`
- [redacted] `msPKI-Enrollment-Flag` value
- Certificate specifies any client authentication EKU

Requisites:

- `GenericWrite` over any account A to compromise any account B

In this case, `John@corp.local` has `GenericWrite` over `Jane@corp.local`, and we wish to compromise `Administrator@corp.local`. `Jane@corp.local` is allowed to enroll in the certificate template `ESC9` that specifies the `CT_FLAG_NO_SECURITY_EXTENSION` flag in the `msPKI-Enrollment-Flag` value.

First, we obtain the hash of `Jane` with for instance Shadow Credentials (using our `GenericWrite`).

Next, we change the `userPrincipalName` of `Jane` to be `Administrator`. Notice

Notice that the `userPrincipalName` in the certificate is `Administrator` and that the

Then, we change back the `userPrincipalName` of `Jane` to be something else, like her original `userPrincipalName` `Jane@corp.local`.

Now, if we try to authenticate with the certificate, we will receive the NT hash of the `Administrator@corp.local` user. You will need to add `-domain <domain>` to your command line since there is no domain specified in the certificate.

And voilà.

ESC10(Case 1)

Conditions:

- `StrongCertificateBindingEnforcement` set to `0`

Next, we change the `userPrincipalName` of `Jane` to be `Administrator`. Notice that we're leaving out the `@corp.local` part.

This is not a constraint violation, since the `Administrator` user's `userPrincipalName` is `Administrator@corp.local` and not `Administrator`.

Now, we request any certificate that permits client authentication, for instance the default `User` template. We must request the certificate as `Jane`.

ESC10(Case 2)

Conditions:

- `CertificateMappingMethods` contains `UPN` bit flag (`0x4`)

Requisites:

- `GenericWrite` over any account A to compromise any account B without a `userPrincipalName` property (machine accounts and built-in domain administrator `Administrator`)

In this case, `John@corp.local` has `GenericWrite` over `Jane@corp.local`, and we wish to compromise the domain controller `DC$@corp.local`.

First, we obtain the hash of `Jane` with for instance Shadow Credentials (using our `GenericWrite`).

Then, we change back the `userPrincipalName` of `Jane` to be something else, like her original `userPrincipalName` (`Jane@corp.local`).

Now, since this registry key applies to Schannel, we must use the certificate for authentication via Schannel. This is where Certipy's new `-ldap-shell` option comes in.

If we try to authenticate with the certificate and `-ldap-shell`, we will notice that we're authenticated as `u:CORP\DC$`. This is a string that is sent by the server.

Alternatively, we can also compromise any user account where there is no `userPrincipalName` set or where the `userPrincipalName` doesn't match the `sAMAccountName` of that account. From my own testing, the default domain administrator `Administrator@corp.local` doesn't have a `userPrincipalName` set by default, and this account should by default have more privileges in LDAP than domain controllers.

Conclusion

Written by Oliver Lyak

Follow

320 Followers · Editor for IFCR

Twitter: https://twitter.com/ly4k_ Github: https://github.com/ly4k/

More from Oliver Lyak and IFCR

Oliver Lyak in IFCR

Certifried: Active Directory Domain Privilege Escalation (CVE-2022-...

In this blog post, we'll dive into a recently patched Active Directory Domain Privilege...

May 10, 2022 148 1

Oliver Lyak in IFCR

Certipy 2.0: BloodHound, New Escalations, Shadow Credentials,...

As the title states, the latest release of Certipy contains many new features, techniques and...

Feb 19, 2022 137

Ur
Techniques on Windows: An...

The Windows Antimalware Scan Interface (AMSI) is a pivotal component in Microsoft's...

Sep 30

One Leg and Find Out

According to new research, the time you can stand on one leg is the best marker of...

Oct 23 6.5K 151

See more recommendations

Help Status About Careers Press Blog Privacy Terms Text to speech Teams