GitHub - GhostPack/Certify: Active Directory certificate abuse.

By bytewreck

Archived: 2026-04-05 21:56:54 UTC

Certify is a C# tool to enumerate and abuse misconfigurations in Active Directory Certificate Services (AD CS).

@harmj0y and @tifkin_ are the primary authors of Certify and the the associated AD CS research (blog and whitepaper).

Table of Contents

Usage

A command overview and comprehensive usage details can be found on the wiki.

Defensive Considerations

Certify was released at Black Hat 2021 with our "Certified Pre-Owned: Abusing Active Directory Certificate Services" talk.

See our whitepaper for prevention and detection guidance.

Compile Instructions

We are not planning on releasing binaries for Certify, so you will have to compile yourself :)

Certify has been built against .NET 4.7.2 and is compatible with Visual Studio 2022 Community Edition. Simply open up the project .sln, choose "Release", and build.

Sidenote: Running Certify Through PowerShell

If you want to run Certify in-memory through a PowerShell wrapper, first compile the Certify and base64-encode the resulting assembly:

```
[Convert]::ToBase64String([IO.File]::ReadAllBytes("C:\Temp\Certify.exe")) | Out-File -Encoding ASCII C:\Temp\Ce
```

Certify can then be loaded in a PowerShell script with the following (where "aa..." is replaced with the base64-encoded Certify assembly string):

```
$CertifyAssembly = [System.Reflection.Assembly]::Load([Convert]::FromBase64String("aa..."))
```

The Main() method and any arguments can then be invoked as follows:

```
[Certify.Program]::Main("enum-templates --filter-enabled --filter-vulnerable".Split())
```

Sidenote Sidenote: Running Certify Over PSRemoting

Due to the way PSRemoting handles output, we need to redirect stdout to a string and return that instead. Luckily, Certify has a function to help with that.

If you follow the instructions in Sidenote: Running Certify Through PowerShell to create a Certify.ps1, append something like the following to the script:

```
[Certify.Program]::MainString("enum-templates --filter-enabled --filter-vulnerable")
```

You should then be able to run Certify over PSRemoting with something like the following:

```
 $s = New-PSSession dc.theshire.local
Invoke-Command -Session $s -FilePath C:\Temp\Certify.ps1
```

Alternatively, Certify's `/outfile:C:\FILE.txt` argument will redirect all output streams to the specified file.

Reflections

On the subject of public disclosure, we self-embargoed the release of our offensive tooling (Certify as well as ForgeCert) for ~45 days after we published our whitepaper in order to give organizations a chance to get a grip on the issues surrounding Active Directory Certificate Services. We also preemptively released some Yara rules/IOCs for both projects and released the defensive-focused PSPKIAudit PowerShell project along with the whitepaper. However, we have found that organizations and vendors have historically often not fixed issues or built detections for "theoretical" attacks until someone proves something is possible with a proof of concept.

Acknowledgments

Certify used a few resources found online as reference and inspiration:

- This post on requesting certificates from C#.

- This gist for SAN specification.
- This StackOverflow post on exporting private keys.
- This PKISolutions post on converting pkiExpirationPeriod.
- This section of MS-CSRA describing enrollment agent security DACLs.

The AD CS work was built on work from a number of others. The whitepaper has a complete treatment, but to summarize:

- Benjamin Delpy for his extensive work on smart cards/certificates with Mimikatz and Kekeo.
- PKI Solutions for their excellent posts on PKI in Active Directory, as well as their PSPKI PowerShell module, which our auditing toolkit is based on.
- The "Windows Server 2008 – PKI and Certificate Security" book by Brian Komar.
- The following open technical specifications provided by Microsoft:
    - [MS-CERSOD]: Certificate Services Protocols Overview
    - [MS-CRTD]: Certificate Templates Structure
    - [MS-CSRA]: Certificate Services Remote Administration Protocol
    - [MS-ICPR]: ICertPassage Remote Protocol
    - [MS-WCCE]: Windows Client Certificate Enrollment Protocol
- Christoph Falta's GitHub repo which covers some details on attacking certificate templates, including virtual smart cards as well as some ideas on ACL based abuses.
- CQURE's "The tale of Enhanced Key (mis)Usage" post which covers some Subject Alternative Name abuses.
- Keyfactor's 2016 post "Hidden Dangers: Certificate Subject Alternative Names (SANs)"
- @Elkement's posts "Sizzle @ hackthebox – Unintended: Getting a Logon Smartcard for the Domain Admin!" and "Impersonating a Windows Enterprise Admin with a Certificate: Kerberos PKINIT from Linux" detail certificate template misconfigurations.
- Carl Sörqvist wrote up a detailed, and plausible, scenario for how some of these misconfigurations happen titled "Supply in the Request Shenanigans".
- Ceri Coburn released an excellent post in 2020 on "Attacking Smart Card Based Active Directory Networks" detailing some smart card abuse and Certify additions.
- Brad Hill published a whitepaper titled "Weaknesses and Best Practices of Public Key Kerberos with Smart Cards" which provided some good background on Kerberos/PKINIT from a security perspective.
