PowerShell scripts signen

Auteur: Mitch Kalf
Versie: 1.0
Datum: 19-05-2022

INHOUDSOPGAVE

Inleiding

PowerShell is een krachtig hulpmiddel om taken binnen de infrastructuur uit te voeren. Het is daarom van belang dat de code niet gewijzigd is nadat het PowerShell script is ontwikkeld. Om dit te waarborgen kan een PowerShell script gesigned worden. In deze blog vertel ik hier meer over en vooral ook hoe je een PowerShell script kunt signen.

Wat is het signen van PowerShell scripts

Door een script te signen (ondertekenen) met een certificaat wordt de integriteit hiervan gewaarborgd. Dit laat zien dat de code origineel is en dat deze niet door een derde partij/ een virus is gewijzigd.

Waarom zou je dit willen?

Door Code Signing toe te passen in de organisatie, kun je alleen bepaalde gebruikers of groepen toegang geven om scripts te maken. Hierdoor weet je dat een kundig iemand de scripts heeft gemaakt en is het te herleiden wie of welke groep deze heeft gemaakt. Daarnaast zorgt het voor de integriteit van de code. Als de code na het signen is aangepast dan zal de code niet starten.

Is het veilig?

Standaard staat PowerShell op Restricted. Dit betekent dat er geen scripts uitgevoerd kunnen worden op het systeem. Dit is voor een organisatie niet wenselijk want het uitvoeren van scripts is vrijwel altijd noodzakelijk. Door deze instelling aan te passen naar AllSigned, weet je zeker dat alleen code uitgevoerd wordt die vooraf wel ondertekend is. Dit houdt het onbedoeld uitvoeren van scripts of het uitvoeren van gewijzigde scripts tegen.
Het weerhoudt gebruikers helaas niet om zelf PowerShell op te starten, de code direct in PowerShell te typen en zo uit te voeren.

Twee manieren van Code Signing distributie

Er zijn twee soorten aanpak voor de distributie van code signing certificaten.

1. Een certificaat per groep
2. Een certificaat per persoon

Hieronder worden deze opties beschreven.

Een certificaat per groep

Door een certificaat aan een groep toe te kennen is het overzichtelijker om alle certificaten te beheren. Door deze manier te kiezen kun je een gebruiker lid maken van een groep en door middel van een Group Policy (GPO) het certificaat pushen naar de machine. Hierdoor houd je een overzicht van certificaten per groep en de leden van deze groep.
Voordelen:

- Eén certificaat per groep om te beheren.
- Makkelijke distributie zonder tussenkomst van gebruikers

Nadelen:

- Je weet niet wie het bestand heeft gesigned, alleen de groep.
- Je kan een certificaat niet individueel intrekken (b.v. als een medewerker uit dienst gaat)

Een certificaat per persoon

Door een certificaat per persoon te genereren heb je je gebruikers allemaal in kaart. Iedereen heeft zijn eigen certificaat en dit is te bekijken per script.
Voordeel:

- Je ziet wie welke code heeft gemaakt.
- Je kan een certificaat individueel intrekken.

Nadeel:

- Veel certificaten in je organisatie.
- Vereist tussenkomst van de gebruiker.

Daarnaast kun je voor evaluatie- en testdoeleneinden ook gebruik maken van een eigen gegenereerd certificaat. Dit wordt hieronder beschreven.

Zelf een certificaat aanmaken

Het zelf aanmaken van een code signing certificaat is ook mogelijk.
Let wel op, dit is niet per definitie veilig aangezien je zelf goedkeurt zonder controle van een derde partij.

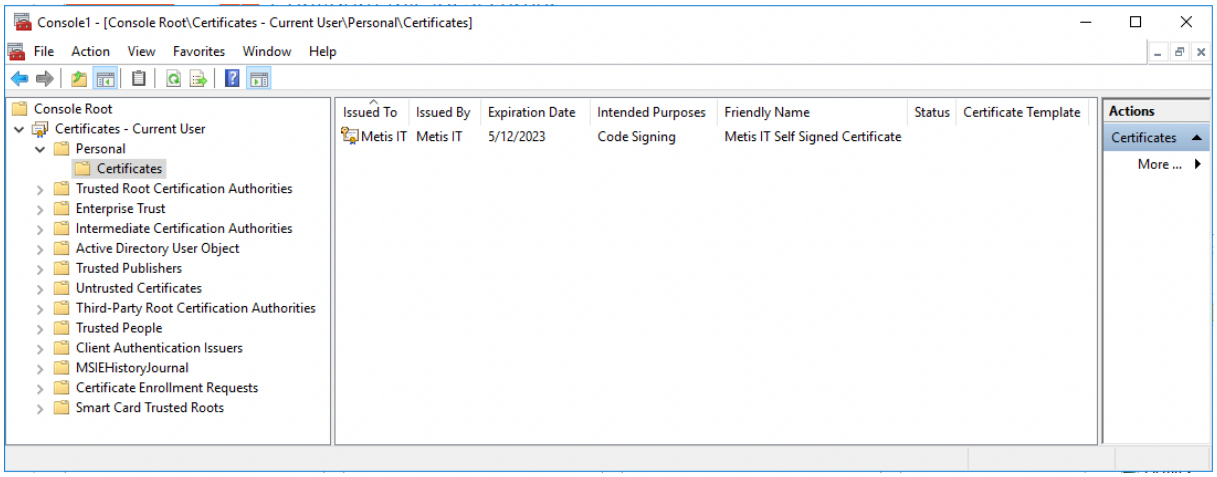
Eerst starten we PowerShell op als Administrator.
Vul hierna de volgende code in:

```
$createCertificate = New-SelfSignedCertificate -FriendlyName "Metis IT Self Signed Certificate" -CertStoreLocation Cert:\LocalMachine\My -Subject "Metis IT" -Type CodeSigningCert
```

De uitleg van alle commando's:

```
$createCertificate =                                  <- dit is een variabele die we later kunnen hergebruiken

New-SelfSignedCertificate                             <- Het commando om een certificaat aan te vragen

-FriendlyName "Metis IT Self Signed Certificate"      <- De omschrijving van het certificaat

-CertStoreLocation Cert:\CurrentUser\My               <- De locatie waar het certificaat komt te staan

-Subject "Metis IT"                                   <- De naam van het certificaat

-Type CodeSigningCert                                 <- Het type certificaat wat we nodig hebben
```

Dit bovenstaande commando zorgt ervoor dat er een Code Signing certificaat is aangemaakt zoals in onderstaand voorbeeld. Dit certificaat kunnen we gebruiken om ons script te signen.



Als volgende stap moeten we het certificaat ook in de Trusted Root Certification Authorities plaatsen. Dit zorgt ervoor dat je het zelfgemaakte certificaat vertrouwt om een code te mogen signen.

Om dit te doen, voer je het volgende commando uit.

```
$rootStore =
[System.Security.Cryptography.X509Certificates.X509Store]::new("Root","CurrentUser"
)
$rootStore.Open("ReadWrite")
$rootStore.Add($createCertificate)
$rootStore.Close()
```

De uitleg van alle commando's:

```
$rootStore =
[System.Security.Cryptography.X509Certificates.X509Store]::new("Root","CurrentUser"
) ← Dit zorgt ervoor dat er een variabele gemaakt wordt naar de juiste map

$rootStore.Open("ReadWrite") ← Open de map voor lezen en schrijven

$rootStore.Add($createCertificate) ← Kopieer het certificaat welke in de
$createCertificate variabele staat van de eerste stap

$rootStore.Close() ← sluit de map
```

Nu staat het Code Signing certificaat ook in de juiste folder zodat je mag code signen. Doe je deze stap niet, dan zal je een UnknownError krijgen.

Als volgende stap gaan we het certificaat in een variabele zetten, zodat we alleen de variabele hoeven te gebruiken.

```
$codeCertificate = Get-ChildItem Cert:\CurrentUser\My -CodeSigningCert | where
{$_.Subject -eq "CN=Metis IT"}
```

De uitleg van alle commando's:

```
$codeCertificate =                      ← De naam van de variabele
Get-ChildItem Cert:\CurrentUser\My      ← We pakken het certificaat uit de juiste
                                          folder
-CodeSigningCert                        ← We willen het CodeSigning certificaat
| where {$_.Subject -eq "CN=Metis IT"}  ← We zoeken het certificaat waarvan het
                                          Subject gelijk is aan de naam die we bij
                                          stap 1 hebben ingevuld.
```

Nu zijn we klaar om onze code te signen. Dat doen we door de volgende code uit te voeren:

```
Set-AuthenticodeSignature -Certificate $codeCertificate -FilePath
C:\Scripts\Signed.ps1 -TimestampServer http://timestamp.comodoca.com
```

De uitleg van alle commando's:

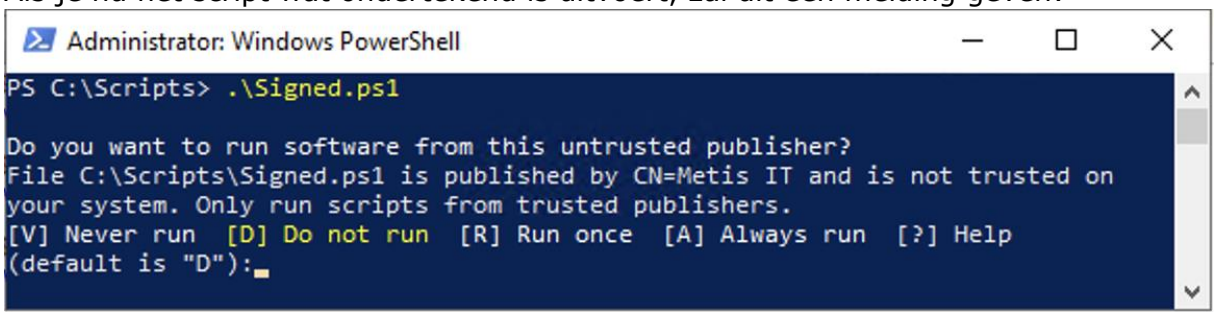
```
Set-AuthenticodeSignature                   ← Het commando dat we iets
                                              willen signen
-Certificate $codeCertificate               ← Hier laden we de variabele
                                              voor welk certificaat we moeten
                                              gebruiken
-FilePath C:\Scripts\Signed.ps1             ← Welk bestand willen we
                                              signen
-TimestampServer http://timestamp.comodoca.com  ← Een timestamp server
                                                  specificeren.
```

De TimestampServer (tijdsstempel) geeft de exacte tijd weer waarop het certificaat aan het bestand is toegevoegd. Een tijdstempel voorkomt dat het script mislukt als het certificaat verloopt, omdat gebruikers en programma's kunnen controleren of het certificaat geldig was op het moment van ondertekening. Hierdoor is het script nog steeds uitvoerbaar, ook als het certificaat is verlopen.

Nu kunnen we PowerShell instellen om alleen maar ondertekende scripts te accepteren. Dit doe je door het volgende commando uit te voeren:

Set-ExecutionPolicy AllSigned.

Als je nu het script wat ondertekend is uitvoert, zal dit een melding geven:

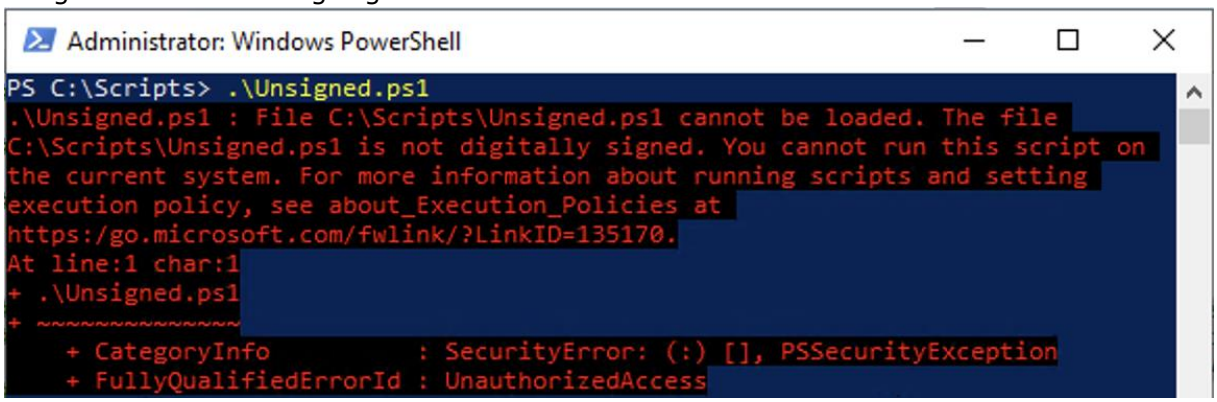


Om geen melding zal laten zien dat het een onbekende uitgever is dien je tijdens het eerste keer draaien van het script op [A] te drukken. Door op [A] te drukken wordt het certificaat in de Trusted Publishers geplaatst.
Ook kan je het volgende commando uitvoeren om het betreffende certificaat in de Trusted Publishers map te zetten.

Het commando:

```
$rootStore =
[System.Security.Cryptography.X509Certificates.X509Store]::new("TrustedPublishers",
"CurrentUser")
$rootStore.Open("ReadWrite")
$rootStore.Add($createCertificate)
$rootStore.Close()
```

Als we nu het script uitvoeren, zal deze direct werken.
Als je nog een script aanmaakt en deze noem je Unsigned.ps1, dan zal de volgende melding naar voren komen. Dit betekent dat het script niet uitgevoerd kan worden aangezien deze niet is gesigned.



Het bovenstaande voorbeeld is geschikt om in een test en/of development omgeving te gebruiken en geeft een eerste indruk wat er mogelijk is met code signen.

In het volgende deel zal ik uitleggen hoe we een persoonsgebonden certificaat kunnen aanvragen op basis van een Active Directory Certificate Authority in een organisatie.