Self-signed certificate generator (PowerShell)

DescriptionThis script is an enhanced open-source PowerShell implementation of deprecated makecert.exe tool and utilizes the most modern certificate API — CertEnroll.

Download New-SelfSignedCertificateEx.zip

Ratings	(39)
Downloaded	39,262 times
Favorites	Add to favorites
Category	Security
Sub category	Script Signing
Updated	9/11/2016
License	TechNet terms of use
Share it:	
Tags	Powershell, Certificate, makecert

Description

This script is an enhanced open-source PowerShell implementation of deprecated makecert.exe tool and utilizes the most modern certificate API — CertEnroll.

The script is intended for test environments to ensure that particular application is properly configured to use digital certificates before the application is deployed in a production environment.

The self-signed certificate generator consist of a single PowerShell function named "New-SelfSignedCertificateEx". Import the function in to current PowerShell session and call the function with desired parameters.

Script parameters

The script defines the following parameters:

- **Subject** — specifies the certificate subject in a X500 distinguished name format. Example: *CN=Test Cert, OU=Sandbox*
- **NotBefore** — Specifies the date and time when the certificate become valid. By default previous day date is used.
- **NotAfter** — Specifies the date and time when the certificate expires. By default, the certificate is valid for 1 year.
- **SerialNumber** — Specifies the desired serial number in a hex format. Example: *01a4ff2*
- **ProviderName** — Specifies the Cryptography Service Provider (CSP) name. You can use either legacy CSP and Key Storage Providers (KSP). By default "*Microsoft Enhanced Cryptographic Provider v1.0*" CSP is used.
- **AlgorithmName** — Specifies the public key algorithm. By default RSA algorithm is used. *RSA* is the only algorithm supported by legacy CSPs. With key storage providers (KSP) you can use CNG algorithms, like ECDH. For CNG algorithms you must use full name:
  ECDH_P256
  ECDH_P384
  ECDH_P521

  In addition, **KeyLength** parameter must be specified explicitly when non-RSA algorithm is used.
- **KeyLength** — Specifies the key length to generate. By default 2048-bit key is generated.
- **KeySpec** — Specifies the public key operations type. The possible values are: Exchange and Signature. Default value is Exchange.
- **EnhancedKeyUsage** — Specifies the intended uses of the public key contained in a certificate. You can specify either, EKU friendly name (for example '*Server Authentication*') or object identifier (OID) value (for example '*1.3.6.1.5.5.7.3.1*').
- **KeyUsage** — Specifies restrictions on the operations that can be performed by the public key contained in the certificate. Possible values (and their respective integer values to make bitwise operations) are:
  EncipherOnly
  CrlSign
  KeyCertSign
  KeyAgreement
  DataEncipherment
  KeyEncipherment
  NonRepudiation
  DigitalSignature
  DecipherOnly

  you can combine key usages values by using bitwise OR operation. when combining multiple flags, they must be enclosed in quotes and separated by a comma character. For example, to combine KeyEncipherment and DigitalSignature flags you should type: "*KeyEncipherment, DigitalSignature*".

  If the certificate is CA certificate (see **IsCA** parameter), key usages extension is generated automatically with the following key usages: *Certificate Signing, Off-line CRL Signing, CRL Signing*.

- **SubjectAlternativeName** — Specifies alternative names for the subject. Unlike Subject field, this extension allows to specify more than one name. Also, multiple types of alternative names are supported. The cmdlet supports the following SAN types:
  RFC822 Name
  IP address (both, IPv4 and IPv6)
  Guid
  Directory name
  DNS name

- **IsCA** — Specifies whether the certificate is CA (IsCA = $true) or end entity (IsCA = $false) certificate. If this parameter is set to $false, PathLength parameter is ignored. If this parameter is set to $true, **Basic Constraints** extension is marked as critical.

- **PathLength** — Specifies the number of additional CA certificates in the chain under this certificate. If **PathLength**parameter is set to zero, then no additional (subordinate) CA certificates are permitted under this CA.

- **CustomExtension** — Specifies the custom extension to include to a self-signed certificate. This parameter must not be used to specify the extension that is supported via other parameters. In order to use this parameter, the extension must be formed in a collection of initialized X509Extension objects.

- **SignatureAlgorithm** — Specifies signature algorithm used to sign the certificate. By default 'SHA1' algorithm is used. Currently signature algorithms are limited to:
  MD5
  SHA1
  SHA256
  SHA384
  SHA512

- **FriendlyName** — Specifies friendly name for the certificate.

- **StoreLocation** — Specifies the store location to store self-signed certificate. Possible values are: '*CurrentUser*' and '*LocalMachine*'. 'CurrentUser' store is intended for user certificates and computer (as well as CA) certificates must be stored in 'LocalMachine' store.

- **Path** — Specifies the path to a PFX file to export a self-signed certificate.

- **Password** — Specifies the password for PFX file.

- **AllowSMIME** — Enables Secure/Multipurpose Internet Mail Extensions for the certificate.

- **Exportable** — Marks private key as exportable. Smart card providers usually do not allow exportable keys.

And here are several useful examples:

Examples

Creates a self-signed certificate intended for code signing and which is valid for 5 years. Certificate is saved in the Personal store of the current user account:

PowerShell

```
New-SelfsignedCertificateEx -Subject "CN=Test Code Signing" -EKU "Code Signing" -
KeySpec "Signature" `
-KeyUsage "DigitalSignature" -FriendlyName "Test code signing" -
NotAfter $([datetime]::now.AddYears(5))
```

Creates a self-signed SSL certificate with multiple subject names and saves it to a file. Additionally, the certificate is saved in the Personal store of the Local Machine store. Private key is marked as exportable, so you can export the certificate with a associated private key to a file at any time. The certificate includes SMIME capabilities:

PowerShell

```
New-SelfsignedCertificateEx -Subject "CN=www.domain.com" -
EKU "Server Authentication", "Client authentication" `
-KeyUsage "KeyEncipherment, DigitalSignature" -
SAN "sub.domain.com","www.domain.com","192.168.1.1" `
-AllowSMIME -Path C:\test\ssl.pfx -Password (ConvertTo-SecureString "P@ssw0rd" -AsPlainText -
Force) -Exportable `
-StoreLocation "LocalMachine"
```

Creates a self-signed SSL certificate with multiple subject names and saves it to a file. Additionally, the certificate is saved in the Personal store of the Local Machine store. Private key is marked as exportable, so you can export the certificate with a associated private key to a file at any time. Certificate uses Elliptic Curve Cryptography (ECC) key algorithm ECDH with 256-bit key. The certificate is signed by using SHA256 algorithm:

PowerShell

```
New-SelfsignedCertificateEx -Subject "CN=www.domain.com" -
EKU "Server Authentication", "Client authentication" `
-KeyUsage "KeyEncipherment, DigitalSignature" -
SAN "sub.domain.com","www.domain.com","192.168.1.1" `
-StoreLocation "LocalMachine" -ProviderName "Microsoft Software Key Storae Provider" -
AlgorithmName ecdh_256 `
-KeyLength 256 -SignatureAlgorithm sha256
```

Generates self-signed certificate for CA. Actually, this example creates a root CA certificate which is valid for 5 years:

PowerShell

```
New-SelfsignedCertificateEx -Subject "CN=Test Root CA, OU=Sandbox" -IsCA $true -ProviderName `
"Microsoft Software Key Storage Provider" -Exportable
```

Feedback

If you found bugs, have suggestions or questions, you are welcome in Q&A section.

Verified on the following platforms

Windows 10	**Yes**
Windows Server 2012	**Yes**
Windows Server 2012 R2	**Yes**
Windows Server 2008 R2	**Yes**
Windows Server 2008	**Yes**
Windows Server 2003	**No**
Windows 8	**Yes**
Windows 7	**Yes**
Windows Vista	**Yes**
Windows XP	**No**
Windows 2000	**No**

This script is tested on these platforms by the author. It is likely to work on other platforms as well. If you try it and find that it works on another platform, please add a note to the script discussion to let others know.

Online peer support

For online peer support, join The Official Scripting Guys Forum! To provide feedback or report bugs in sample scripts, please start a new discussion on the Discussions tab for this script.

46,301
Points
Top 0.1

Vadims Podans
MCC, MVP
Joined Aug 2008

View contributions

19
Show activity