Zertifikat bei Lets Encrypt per Powershell beantragen

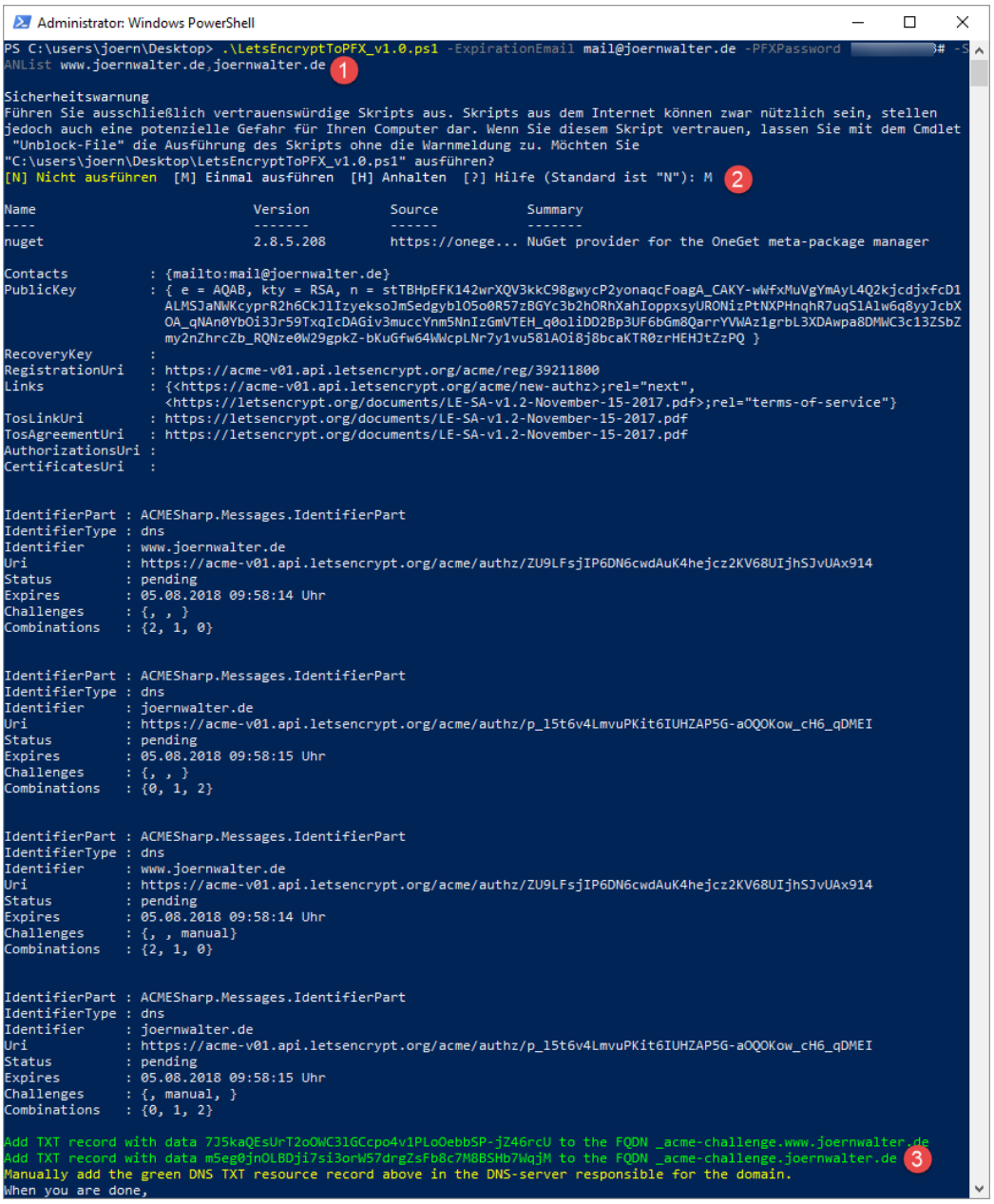
Es ist recht einfach sich ein Zertifikat über Let's Encrypt ausstellen zu lassen.

Das Powershell Skript habe ich auf den Desktop abgelegt.

Starte die Powershell und führe das Skript .\LetsEncrypttoPFX_1.0.ps1 mit den Optionen .\LetsEncrypttoPFX_1.0.ps1 –ExpirationEmail mail@joernwalter.de –PFXPassword xxxxxxxx –SanList www.joernwalter.de,joernwalter.de aus.

Der Schalter -ExpirationEmail wird für die Ablaufbenachrichtigung benötigt, -PFX Password schützt den Privaten-Schlüssel vor unbefugten Zugriff und –SanList steht für Alternativer Antragstellername (Subject Alternative Name.)

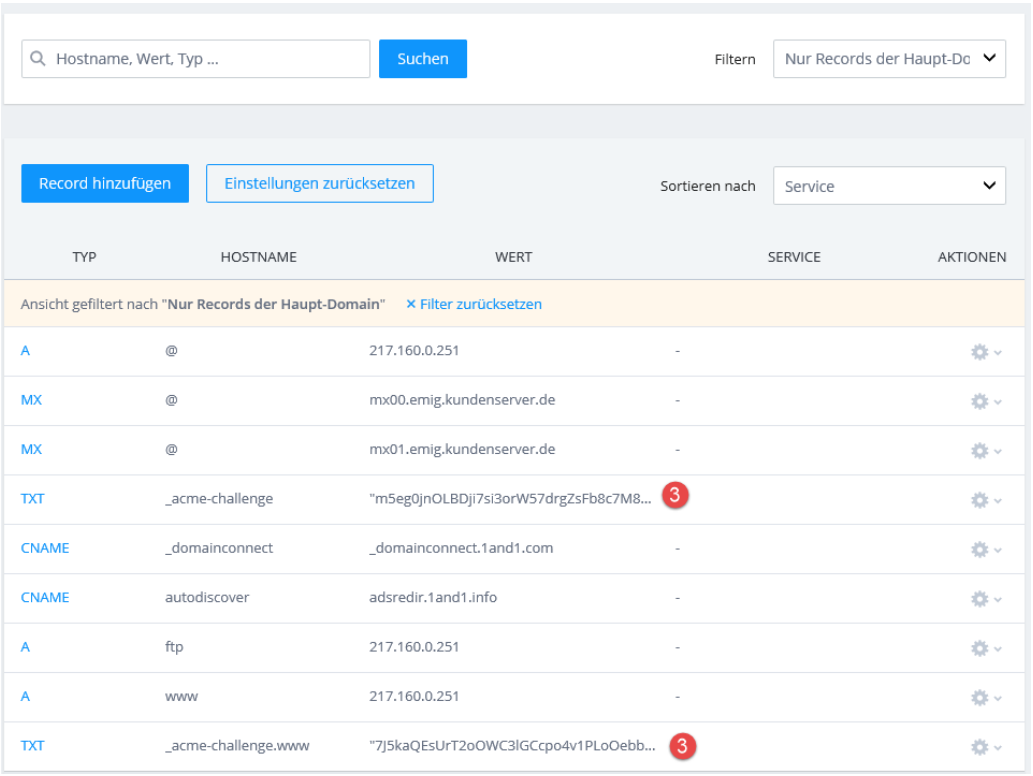
Bestätige bei Punkt 2 die Ausführung mit „M".



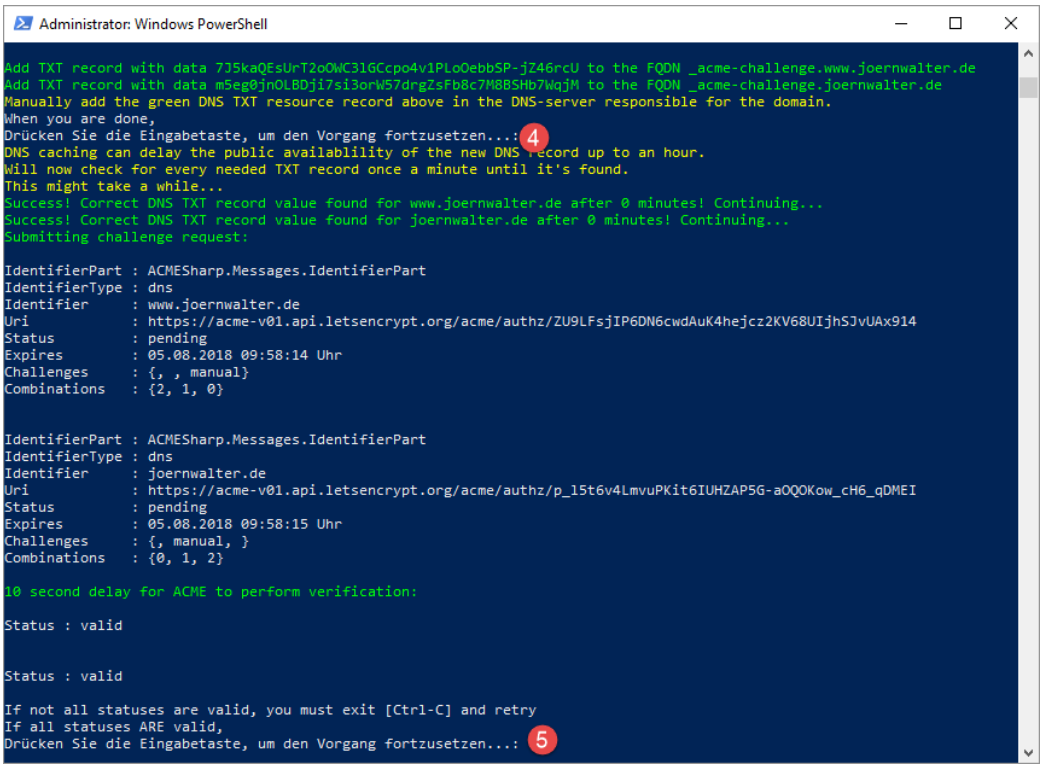
Zertifikat bei Lets Encrypt per Powershell beantragen

Punkt 3 verlangt zur Verifizierung der Eigentümerschaft der angegebenen Domain, in diesem Fall www.joernwalter.de und joernwalter.de einen Eintrag in DNS.

Dazu müssen wie angegeben 2 neue Records erstellt werden. Sobald diese angelegt sind, kann die Verifizierung gestartet werden.



Mit der Eingabe-Taste starten wir die Verifizierung. Es wird geprüft, ob die beide neuen Records mit den vorgegebenen Werten angelegt wurden.

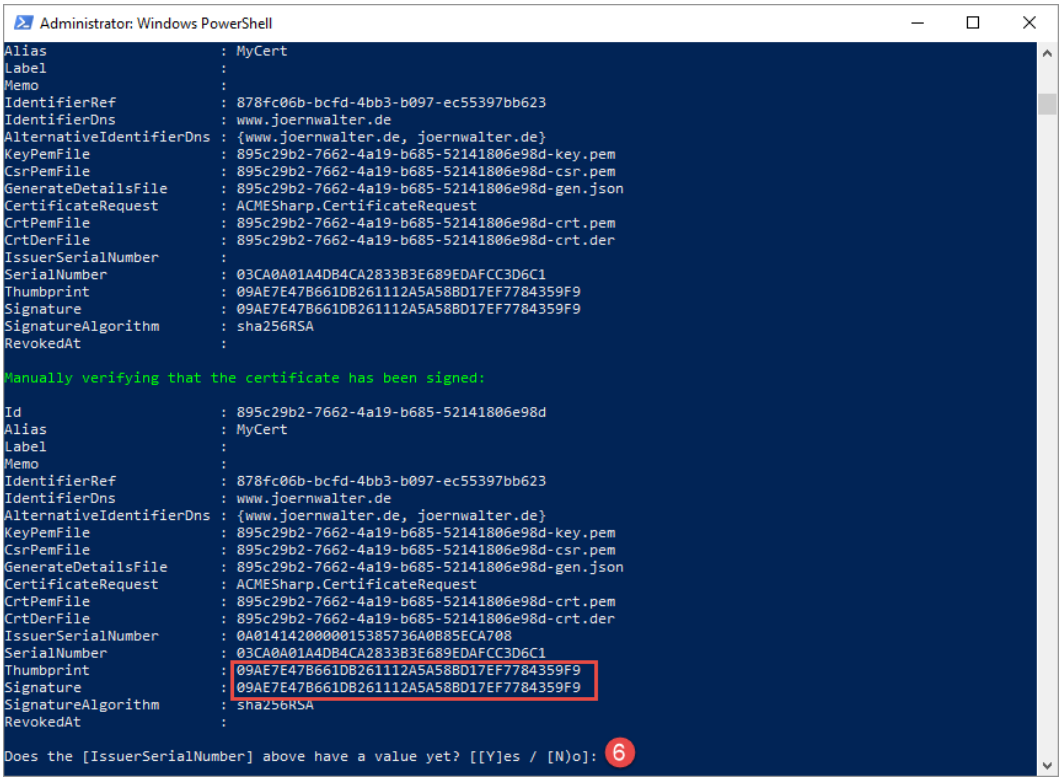


Zertifikat bei Lets Encrypt per Powershell beantragen

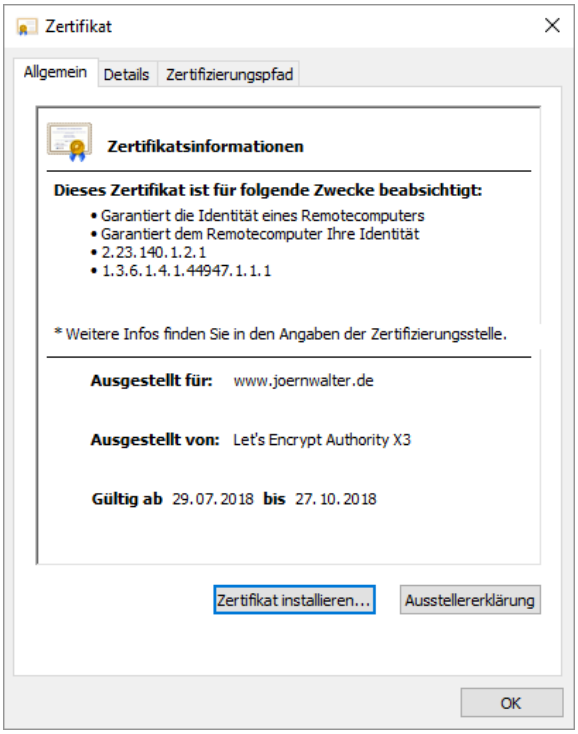
Sobald die Verifizierung positiv (Status = valid) verlaufen ist, drücken wir die Eingabetaste und das Zertifikat wird ausgestellt.

Manuelle Überprüfung: Überprüfen den Thumbprint (Fingerprint) aus der Powershell mit dem im Zertifikat.

Stimmt dieser überein drücken wir erneut die Eingabetaste.

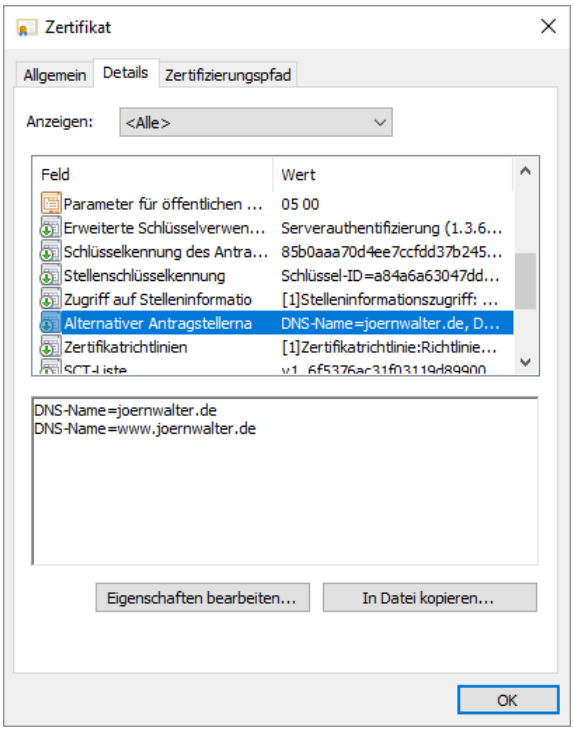


Dazu klicken wir auf den Tab Details:

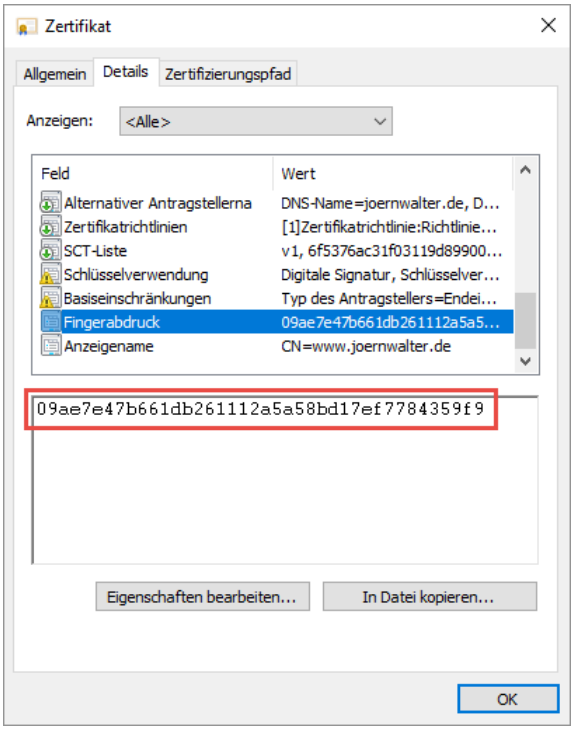


Zertifikat bei Lets Encrypt per Powershell beantragen

Hier finden wir die SANList (Alternativer Antragsteller)



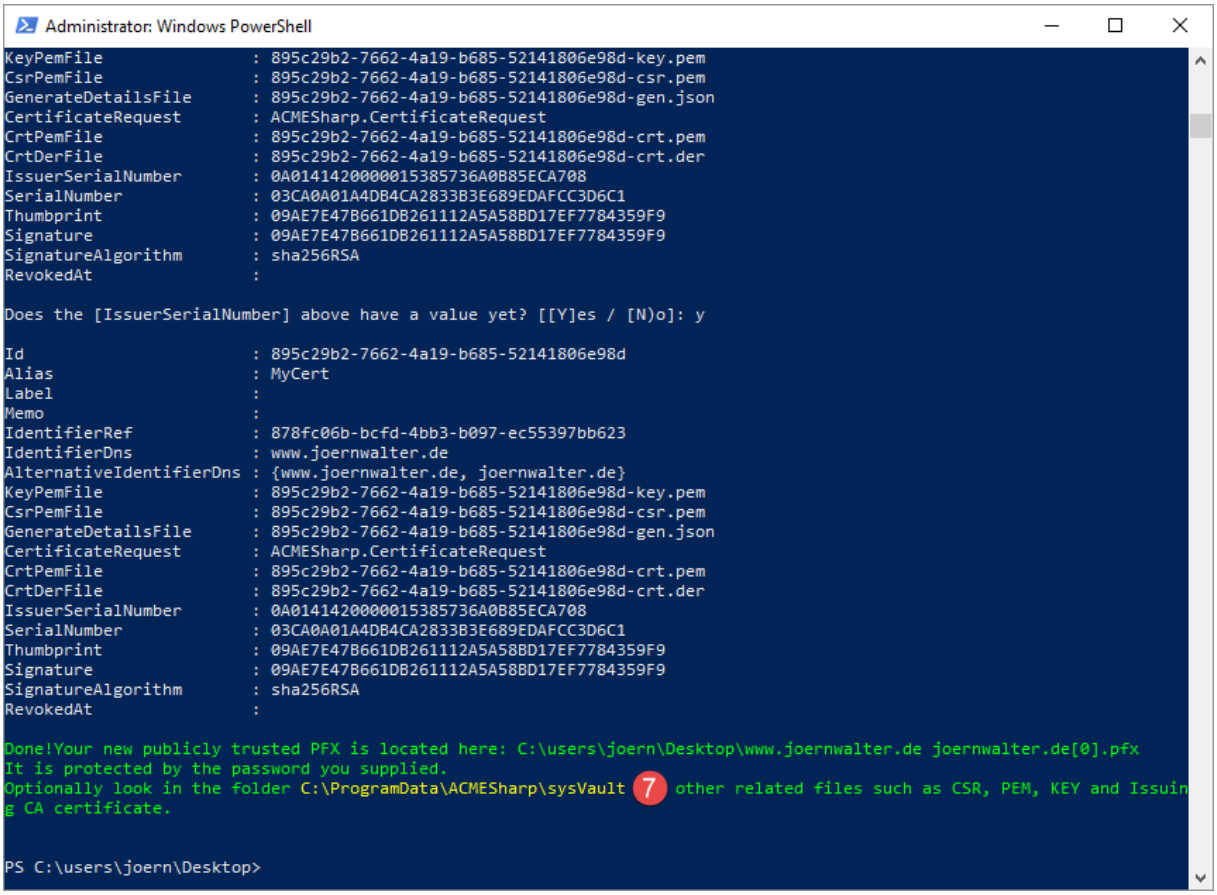
...und den Fingerprint.

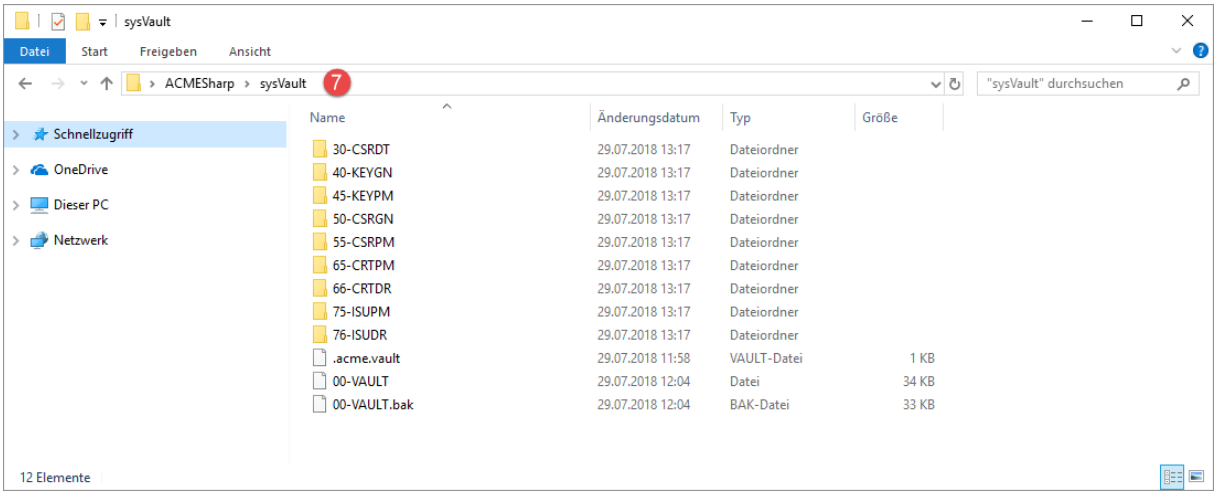


Zertifikat bei Lets Encrypt per Powershell beantragen

Unter Punkt 7 wird uns der Pfad angezeigt, unter dem wir weitere Details zur Zertifikatsbeantragung finden, wie z.B. den Öffentlichen Schlüssel, die Zertifikatsanfrage (CSR), das Zwischenzertifikat usw.

Der Private Schlüssel liegt nun auf dem Desktop und kann schlussendlich über die MMC,IIS usw. importiert und weiterverarbeitet (eingesetzt) werden.





Zertifikat bei Lets Encrypt per Powershell beantragen

Powershell-Skript:

```
#Requires -RunAsAdministrator

param(
    [Parameter(Position = 0)]
    [string]$ExpirationEmail = $(throw "Parameter -ExpirationEmail is required."),
    [string]$PFXPassword = $(throw "Parameter -PFXPassword is required."),
    [string[]]$SANList = $(throw "Parameter -SANList is required.")
    )

################################################################################
##########
#
# Standard Disclaimer:
# I take no responsibility for what this script does, test before running in
production.
#
# If you do not run PowerShell v5 or later you need to manually install
PackageManagement
# PowerShell Modules found here:
# http://go.microsoft.com/fwlink/?LinkID=746217&clcid=0x409
#
# The Domain verification is done by manually adding DNS TXT records.
# The email is ONLY used for sending reminders when certificate is about to expire.
# Renewal is not supported (but a new cert with the same names can be issued).
#
# See my blog for more information: MSSEC.SE
#
# Tom Aafloen, Onevinn, v1.0, 2017-12-04
#
################################################################################
##########


# Relocate any previous files to backup folder, in case this is a re-enroll or a
retry
$RandomSuffix = Get-Random
Rename-Item "$env:ProgramData\ACMESharp\sysVault"
"$env:ProgramData\ACMESharp\sysVault_backup_$RandomSuffix" -Force -ErrorAction
SilentlyContinue


# Install the PackageProvider NuGet
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force


# Install Let's Encrypt PowerShell cmdlets (ACMESharp) using NuGet
# AllowClobber is nessecary on newer Windows versions, since Get-Certificate was
added to builtin PowerShell
# But AllowClobber is an unknown parameter in older PS versions and throws an
exception.
try
{
    Install-Module -Name ACMESharp -Force -AllowClobber
}
catch
{
    Install-Module -Name ACMESharp -Force
}


# Import Let's Encrypt PowerShell cmdlets (ACMESharp)
Import-Module ACMESharp


# Prepare ACMEVault
Initialize-ACMEVault
New-ACMERegistration -Contacts mailto:$ExpirationEmail -AcceptTos


# Create ACME Identifiers for all domains (there are referenced later when making
requests)
foreach ($SAN in $SANList)
{
    New-ACMEIdentifier -Dns $SAN -Alias $SAN
```

```
}


# Create Challenge Data (to prove domain ownership)
foreach ($SAN in $SANList)
{
    Complete-ACMEChallenge $SAN -ChallengeType dns-01 -Handler manual
}


# Show challenge instructions for all DNS names
foreach ($SAN in $SANList)
{
    $recordvalue = ((Update-ACMEIdentifier $SAN -ChallengeType dns-01).Challenges |
Where-Object {$_.Type -eq "dns-01"} | select Challenge -ExpandProperty
Challenge).RecordValue
    Write-Host "Add TXT record with data $recordvalue to the FQDN _acme-
challenge.$SAN" -ForegroundColor Green
}
Write-Host "Manually add the green DNS TXT resource record above in the DNS-server
responsible for the domain." -ForegroundColor Yellow
Write-Host "When you are done,"
Pause


# Checking each DNS TXT record, proceed only when found
Write-Host "DNS caching can delay the public availablility of the new DNS record up
to an hour." -ForegroundColor Yellow
Write-Host "Will now check for every needed TXT record once a minute until it's
found." -ForegroundColor Yellow
Write-Host "This might take a while..." -ForegroundColor Yellow
foreach ($SAN in $SANList)
{
    $DNSvalueCorrect = $false
    $tries = 0
    $ExpectedDNSValue = ((Update-ACMEIdentifier $SAN -ChallengeType dns-
01).Challenges | Where-Object {$_.Type -eq "dns-01"} | select Challenge -
ExpandProperty Challenge).RecordValue
    while (-not $DNSvalueCorrect)
{
        Clear-DnsClientCache
        $CurrentDNSValue = (Resolve-DnsName -Name "_acme-challenge.$SAN" -Type TXT
-NoHostsFile -WarningAction SilentlyContinue -ErrorAction SilentlyContinue).Strings
        if($ExpectedDNSValue -eq $CurrentDNSValue)
        {
            $DNSvalueCorrect = $true
        }
        if(-not $DNSvalueCorrect)
        {
            Write-Host "DNS TXT record value $ExpectedDNSValue still not found for
_acme-challenge.$SAN (after $tries minutes). Retrying..." -ForegroundColor Yellow
            Start-Sleep -Seconds 60
            $tries++
        }
    }

    Write-Host "Success! Correct DNS TXT record value found for $SAN after $tries
minutes! Continuing..." -ForegroundColor Green
}


# Submit challenge requests for all domain names
Write-Host "Submitting challenge request:" -ForegroundColor Green
foreach ($SAN in $SANList)
{
    Submit-ACMEChallenge $SAN -ChallengeType dns-01
}


# Wait for ACME to perform verification
Write-Host "10 second delay for ACME to perform verification:" -ForegroundColor
Green
Start-Sleep -Seconds 10

# Refresh state and verify status
foreach ($SAN in $SANList)
{
```

Zertifikat bei Lets Encrypt per Powershell beantragen

```
    (Update-ACMEIdentifier $SAN -ChallengeType dns-01).Challenges | Where-Object
{$_.Type -eq "dns-01"} | select Status
}


Write-Host "If not all statuses are valid, you must exit [Ctrl-C] and retry"
Write-Host "If all statuses ARE valid,"
Pause


# Create certificate request
Write-Host "Creating certificate request:" -ForegroundColor Green
New-ACMECertificate $SANList[0] -Generate -AlternativeIdentifierRefs $SANList -
Alias MyCert


# Submit certificate request
Write-Host "Submitting certificate request:" -ForegroundColor Green
Submit-ACMECertificate MyCert


# Verify that the certificate has been signed
# Room for improvement: Automated check for IssuerSerialNumber
Write-Host "Manually verifying that the certificate has been signed:" -
ForegroundColor Green
$confirmation = $false
while ($confirmation -ne "y")
{
    Update-ACMECertificate MyCert
    $confirmation = Read-Host "Does the [IssuerSerialNumber] above have a value
yet? [[Y]es / [N]o]"
    if ($confirmation -eq 'y') {break}
}


# Export signed certificate (and private key) to pfx
$CurrentPath = Get-Location
Get-ACMECertificate MyCert -ExportPkcs12 "$CurrentPath\$SANList.pfx" -
CertificatePassword $PFXPassword
Write-Host "Done!" -ForegroundColor Green -NoNewline
Write-Host "Your new publicly trusted PFX is located here:
$CurrentPath\$SANList[0].pfx" -ForegroundColor Green
Write-Host "It is protected by the password you supplied." -ForegroundColor Green
Write-Host "Optionally look in the folder " -NoNewline -ForegroundColor Green
Write-Host "$env:ProgramData\ACMESharp\sysVault " -ForegroundColor Yellow -
NoNewline
Write-Host "for other related files such as CSR, PEM, KEY and Issuing CA
certificate." -ForegroundColor Green


<# Optional:
To cleanup installed packages and packageproviders (might need to close and re-open
PS first):
Get-Package "ACMESharp" | Uninstall-Package -Force
Remove-Item -Path "C:\Program Files\PackageManagement\ProviderAssemblies\nuget" -
Recurse -Force
#>
```