WinRM Certificate Authentication

- Ansible Configuration
- Certificate Generation
- Windows Configuration
  - Import Certificate to the Certificate Store
  - Mapping Certificate to a Local Account
  - Using Ansible

WinRM certificate authentication is a method of authenticating to a Windows host using X.509 certificates instead of a username and password.

Certificate authentication does have some disadvantages compared to SSH key based authentication such as:

- it can only be mapped to a local Windows user, no domain accounts
- the username and password must be mapped to the certificate, if the password changes, the cert will need to be re-mapped
- an administrator on the Windows host can retrieve the local user password through the certificate mapping
- Ansible cannot use encrypted private keys, they must be stored without encryption
- Ansible cannot use the certs and private keys stored as a var, they must be a file

Ansible Configuration

Certificate authentication uses certificates as keys similar to SSH key pairs. The public and private key is stored on the Ansible control node to use for authentication. The following example shows the hostvars configured for certificate authentication:

```
# psrp
ansible_connection: psrp
```

```
ansible_psrp_auth: certificate
ansible_psrp_certificate_pem: /path/to/certificate/public_key.pem
ansible_psrp_certificate_key_pem: /path/to/certificate/private_key.pem

# winrm
ansible_connection: winrm
ansible_winrm_transport: certificate
ansible_winrm_cert_pem: /path/to/certificate/public_key.pem
ansible_winrm_cert_key_pem: /path/to/certificate/private_key.pem
```

Certificate authentication is not enabled by default on a Windows host but can be enabled by running the following in PowerShell:

```
Set-Item -Path WSMan:\localhost\Service\Auth\Certificate -Value $true
```

The private key cannot be encrypted due to a limitation of the underlying Python library used by Ansible.

> For enabling certificate authentication with a TLS 1.3 connection, Python 3.8+, 3.7.1, or 3.6.7 and Python package urllib3>=2.0.7 or newer are required.

Certificate Generation

The first step of using certificate authentication is to generate a certificate and private key. The certificate must be generated with the following properties:

- `Extended Key Usage` must include `clientAuth (1.3.6.1.5.5.7.3.2)`
- `Subject Alternative Name` must include `otherName` entry for `userPrincipalName (1.3.6.1.4.1.311.20.2.3)`

The `userPrincipalName` value can be anything but in this guide we will use the value `$USERNAME@localhost` where `$USERNAME` is the name of the user that the certificate will be mapped to.

This can be done through a variety of methods, such as OpenSSL, PowerShell, or Active Directory Certificate Services. The following example shows how to generate a certificate using OpenSSL:

```
# Set the username to the name of the user the certificate will be mapped to
USERNAME="local-user"
```

```
cat > openssl.conf << EOL
distinguished_name = req_distinguished_name

[req_distinguished_name]
[v3_req_client]
extendedKeyUsage = clientAuth
subjectAltName = otherName:1.3.6.1.4.1.311.20.2.3;UTF8:${USERNAME}@localhost
EOL

openssl req \
    -new \
    -sha256 \
    -subj "/CN=${USERNAME}" \
    -newkey rsa:2048 \
    -nodes \
    -keyout cert.key \
    -out cert.csr \
    -config openssl.conf \
    -reqexts v3_req_client

openssl x509 \
    -req \
    -in cert.csr \
    -sha256 \
    -out cert.pem \
    -days 365 \
    -extfile openssl.conf \
    -extensions v3_req_client \
    -key cert.key

rm openssl.conf cert.csr
```

The following example shows how to generate a certificate using PowerShell:

```
# Set the username to the name of the user the certificate will be mapped to
$username = 'local-user'

$clientParams = @{
    CertStoreLocation = 'Cert:\CurrentUser\My'
    NotAfter          = (Get-Date).AddYears(1)
```

```
    Provider          = 'Microsoft Software Key Storage Provider'
    Subject           = "CN=$username"
    TextExtension     =
@("2.5.29.37={text}1.3.6.1.5.5.7.3.2","2.5.29.17={text}upn=$username@localhost")
    Type              = 'Custom'
}
$cert = New-SelfSignedCertificate @clientParams
$certKeyName =
[System.Security.Cryptography.X509Certificates.RSACertificateExtensions]::GetRSAPrivateKey(
    $cert).Key.UniqueName

# Exports the public cert.pem and key cert.pfx
Set-Content -Path "cert.pem" -Value @(
    "-----BEGIN CERTIFICATE-----"
    [Convert]::ToBase64String($cert.RawData) -replace ".{64}", "$&`n"
    "-----END CERTIFICATE-----"
)
$certPfxBytes = $cert.Export('Pfx', '')
[System.IO.File]::WriteAllBytes("$pwd\cert.pfx", $certPfxBytes)

# Removes the private key and cert from the store after exporting
$keyPath = [System.IO.Path]::Combine($env:AppData, 'Microsoft', 'Crypto', 'Keys',
$certKeyName)
Remove-Item -LiteralPath "Cert:\CurrentUser\My\$($cert.Thumbprint)" -Force
Remove-Item -LiteralPath $keyPath -Force
```

As PowerShell cannot generate a PKCS8 PEM private key, we need to use OpenSSL to convert the `cert.pfx` file to a PEM private key:

```
openssl pkcs12 \
    -in cert.pfx \
    -nocerts \
    -nodes \
    -passin pass: |
    sed -ne '/-BEGIN PRIVATE KEY-/,/-END PRIVATE KEY-/p' > cert.key
```

The `cert.pem` is the public key and the `cert.key` is the plaintext private key. These files must be accessible by the Ansible control node to use for authentication. The private key does not need to be present on the Windows node.

Windows Configuration

Once the public and private key has been generated we need to import and trust the public key and configure the user mapping on the Windows host. The Windows host does not need access to the private key, only the public key `cert.pem` needs to be accessible to configure the certificate authentication.

Import Certificate to the Certificate Store

For Windows to trust the certificate it must be imported into the `LocalMachine\TrustedPeople` certificate store. You can do this by running the following:

```
$cert = [System.Security.Cryptography.X509Certificates.X509Certificate2]::new("cert.pem")

$store = Get-Item -LiteralPath Cert:\LocalMachine\TrustedPeople
$store.Open('ReadWrite')
$store.Add($cert)
$store.Dispose()
```

If the cert is self-signed, or issued by a CA that is not trusted by the host, you will need to import the CA certificate into the trusted root store. As our example uses a self-signed cert, we will import that certificate as a trusted CA but in a production environment you would import the CA that signed the certificate.

```
$cert = [System.Security.Cryptography.X509Certificates.X509Certificate2]::new("cert.pem")

$store = Get-Item -LiteralPath Cert:\LocalMachine\Root
$store.Open('ReadWrite')
$store.Add($cert)
$store.Dispose()
```

Mapping Certificate to a Local Account

Once the certificate has been imported into the `LocalMachine\TrustedPeople` store, the WinRM service can create the mapping between the certificate and a local account. This is done by running the following:

```
# Will prompt for the password of the user.
$credential = Get-Credential local-user
```

```
$cert = [System.Security.Cryptography.X509Certificates.X509Certificate2]::new("cert.pem")
$certChain = [System.Security.Cryptography.X509Certificates.X509Chain]::new()
[void]$certChain.Build($cert)
$caThumbprint = $certChain.ChainElements.Certificate[-1].Thumbprint

$certMapping = @{
    Path       = 'WSMan:\localhost\ClientCertificate'
    Subject    = $cert.GetNameInfo('UpnName', $false)
    Issuer     = $caThumbprint
    Credential = $credential
    Force      = $true
}
New-Item @certMapping
```

The `Subject` is the value of the `userPrincipalName` in the certificate SAN entry. The `Issuer` is the thumbprint of the CA certificate that issued our certificate. The `Credential` is the username and password of the local user we are mapping the certificate to.

Using Ansible

The following Ansible playbook can be used to create a local user and map the certificate provided to use for certificate authentication. It needs to be called `username` and `cert_pem` variable set to the name of the user to create and the path to the public key PEM file that was generated. This playbook expects `cert_pem` to be a self signed certificate, if using a certificate issued by a CA, you will have to edit it so it copies that across and imports it to the `LocalMachine\Root` store instead.

yaml/winrm_cert_auth_setup.yaml

□□□ #2
□ □□□ □□□ 2026-10-31 14:40:04 UTC
□ □□□ □□□ 2026-10-31 14:40:05 UTC