PowerShell Code Signing

PowerShell Code Signing - Introduction

- The following, PowerShell-related file types can have embedded Authenticode signatures:
  - ps1, psm1, psd1, ps1xml, psc1, cdxml, mof
  - Implemented in pwrshsip.dll
- Code signing within PowerShell is performed with Set-AuthenticodeSignature.
- PowerShell also supports the creation of catalog files for module integrity/distribution.
- Code signing is the basis for Constrained Language Mode enforcement.

Why Sign Your Code?

- Incorrect answer:
  - For Execution Policy enforcement
  - To attest that your code is not malicious

- Correct answers:
  - To permit code to execute per application whitelisting policies
    - For PowerShell code, the distinction between what runs in FullLanguage versus ConstrainedLanguage mode
  - To sign trusted 3rd party code that doesn’t ship signed properly
  - To attest origin and integrity of the code that you ship

Code Signing - Retrieval

- Get-AuthenticodeSignature
    - Only retrieves information about the leaf certificate in the chain
        - Only retrieves the first leaf cert. Code can be co-signed by one or more certificates.
    - If a file is catalog-signed and Authenticode-signed, it will only display catalog signer information.
        - Hack: Stop and disable the CryptSvc service to retrieve the Authenticode signature in this scenario.
    - IsOSBinary properly is nice
- Get-SystemDriver (included in ConfigCI module - 10 Enterprise only)
    - Poorly named and poorly designed
    - Retrieves information for all co-signers and all certificates in the chain.
    - Useful for building Device Guard policies and performing advanced signing research.

Authenticode-signed PowerShell Code

```
Write-Host "Hello, world!"

# SIG # Begin signature block
# MIINGwYJKoZIhvcNAQcCoIINDDCCDQgCAQExCzAJBgUrDgMCGgUAMGkGCisGAQQB
# gjcCAQSgWzBZMDQGCisGAQQBgjcCAR4wJgIDAQABBAfzDtgWUsITrck0sYpfvNR
# AgEAAgEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQU4DKhMYGXS4TiU/cEc7JJL5ka
# IrGgggpdMIIFJTCCBA2gAwIBAgIQC3a50UwDDdtgAcMiPPsVjTANBgkqhkiG9w0B
# AQsFADByMQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYD
# VQQLExB3d3cuZGlnaWNlcnQuY29tMTEwLwYDVQQDEyhEaWdpQ2VydCBTSEEyIEFz
...
```

- Prepending data to the signature block will result in a hash mismatch.
- Appending data to the signature block will invalidate the signature. Think about why...

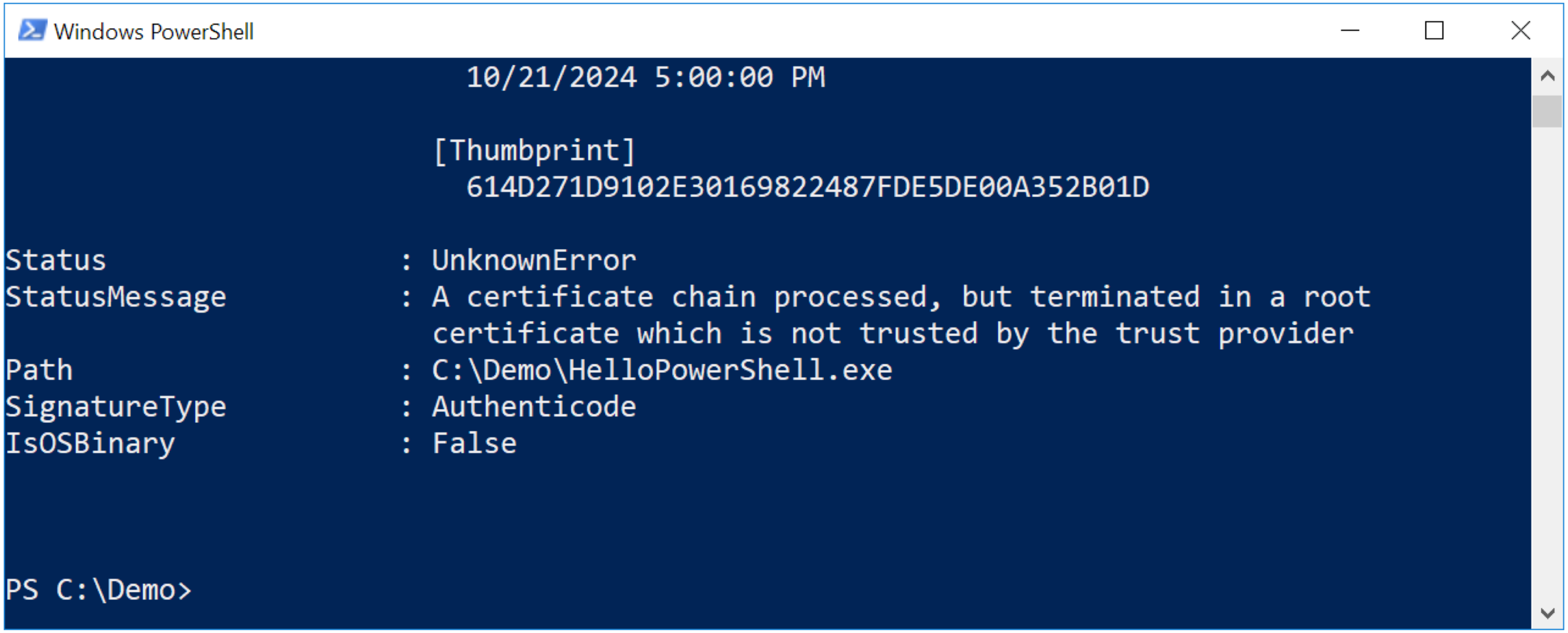
Code Signing - Self-Signed Cert Creation

```
$Arguments = @{
        Subject = 'CN=My Self-signed Code Signing'
        Type = 'CodeSigningCert'
        KeySpec = 'Signature'
        KeyUsage = 'DigitalSignature'
        FriendlyName = 'My Self-signed Code Signing'
        NotAfter = ((Get-Date).AddYears(3))
        CertStoreLocation = 'Cert:\CurrentUser\My'
}

$TestCodeSigningCert = New-SelfSignedCertificate @Arguments
```

Signing Code with PowerShell

```
Add-Type -TypeDefinition @'
using System;

public class Test {
                public static void Main(string[] args) {
        Console.WriteLine("Hello, PowerShell!");
                Console.ReadKey();
                }
}
'@ -OutputAssembly HelloPowerShell.exe

$MySigningCert = ls Cert:\CurrentUser\My\ | ? { $_.Subject -eq 'CN=My Self-signed Code Signing' }
Set-AuthenticodeSignature -Certificate $MySigningCert -TimestampServer 'http://timestamp.digicert.com' -
FilePath .\HelloPowerShell.exe
```

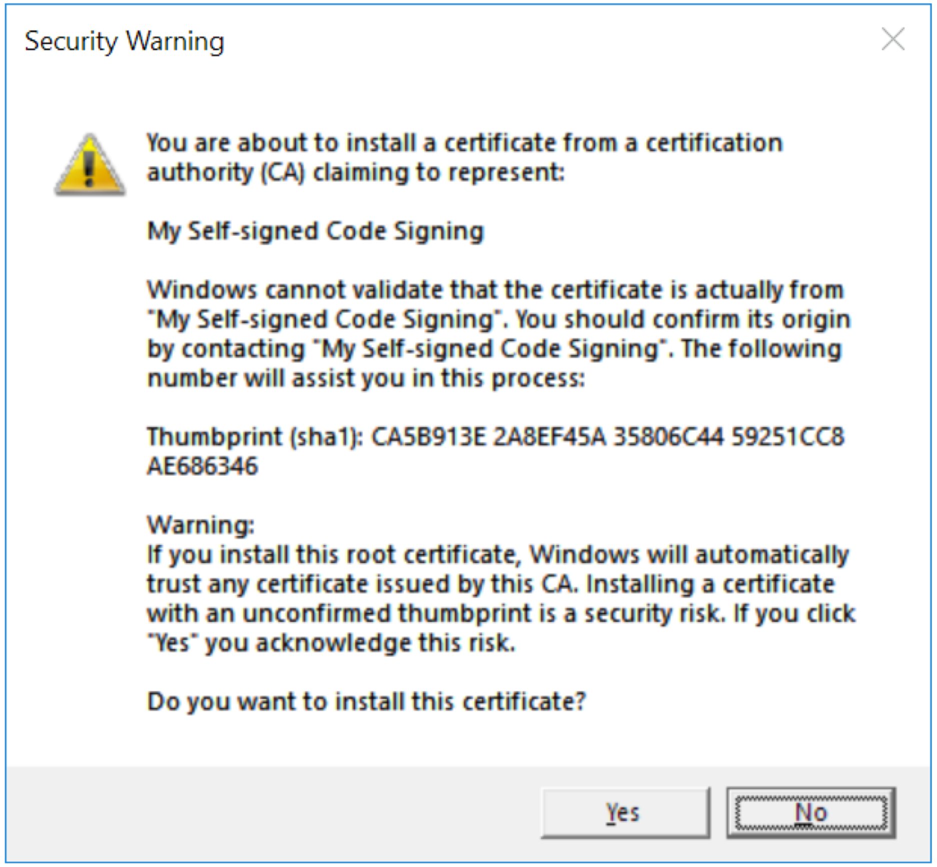
Signing Code with PowerShell



```
                          10/21/2024 5:00:00 PM

                        [Thumbprint]
                          614D271D9102E30169822487FDE5DE00A352B01D

Status                  : UnknownError
StatusMessage           : A certificate chain processed, but terminated in a root
                          certificate which is not trusted by the trust provider
Path                    : C:\Demo\HelloPowerShell.exe
SignatureType           : Authenticode
IsOSBinary              : False



PS C:\Demo>
```

Adding a Trusted Root Certificate

```
$MySigningCert = ls Cert:\CurrentUser\My\ | ? {
$_.Subject -eq 'CN=My Self-signed Code Signing' }

Export-Certificate -FilePath exported_cert.cer -Cert
$MySigningCert

Import-Certificate -FilePath exported_cert.cer -
CertStoreLocation Cert:\CurrentUser\Root

Get-AuthenticodeSignature HelloPowerShell.exe
```


Security Warning

You are about to install a certificate from a certification authority (CA) claiming to represent:

My Self-signed Code Signing

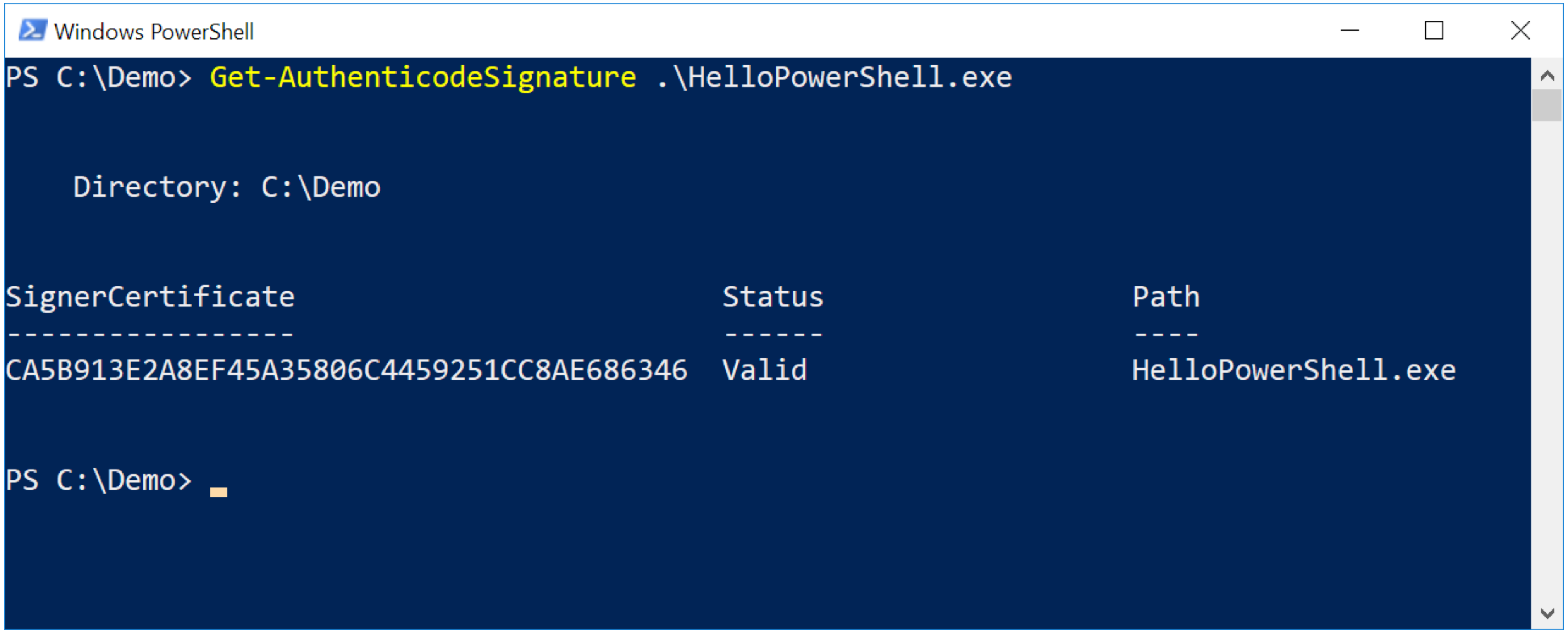
Windows cannot validate that the certificate is actually from "My Self-signed Code Signing". You should confirm its origin by contacting "My Self-signed Code Signing". The following number will assist you in this process:

Thumbprint (sha1): CA5B913E 2A8EF45A 35806C44 59251CC8 AE686346

Warning:
If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.

Do you want to install this certificate?

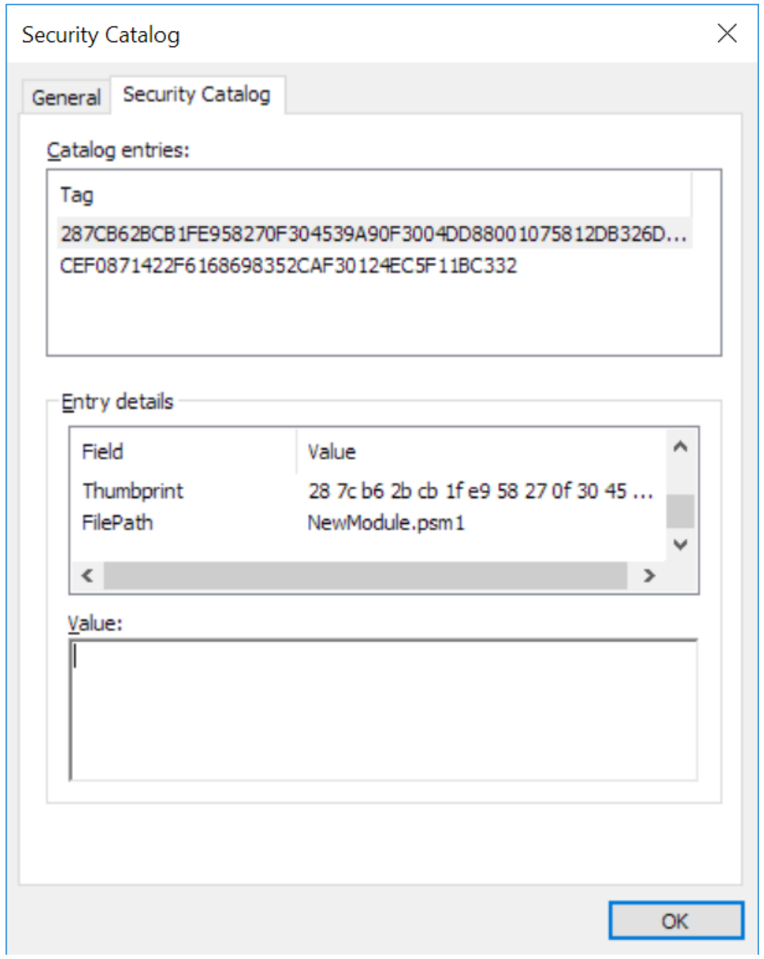
Adding a Trusted Root Certificate



```
PS C:\Demo> Get-AuthenticodeSignature .\HelloPowerShell.exe


    Directory: C:\Demo


SignerCertificate                         Status                   Path
-----------------                         ------                   ----
CA5B913E2A8EF45A35806C4459251CC8AE686346  Valid                    HelloPowerShell.exe


PS C:\Demo> _
```

Catalog Signing

- Catalog-signing (versus Authenticode) permits signing of any file type regardless of “signability”.
- A catalog file is effectively a list of hashes that can be signed.
- When publishing modules to the PowerShell Gallery, integrity validation is performed when a module is signed.
- The process is not documented but it’s pretty straightforward.



Catalog entries:

Tag
287CB62BCB1FE958270F304539A90F3004DD88001075812DB326D...
CEF0871422F6168698352CAF30124EC5F11BC332

Entry details

Field	Value
Thumbprint	28 7c b6 2b cb 1f e9 58 27 0f 30 45 ...
FilePath	NewModule.psm1

Value:

Catalog Signing

```
mkdir NewModule
'Write-Host "This is an awesome module!!!"' | Out-File .\NewModule\NewModule.psm1

New-FileCatalog -CatalogVersion 2 -CatalogFilePath .\NewModule.cat -Path .\NewModule\
Move-Item -Path .\NewModule.cat -Destination .\NewModule\

Test-FileCatalog -FilesToSkip .\NewModule\NewModule.cat -CatalogFilePath
.\NewModule\NewModule.cat -Detailed

$MySigningCert = ls Cert:\CurrentUser\My\ | ? { $_.Subject -eq 'CN=My Self-signed Code Signing' }

Set-AuthenticodeSignature -Certificate $MySigningCert -TimestampServer
'http://timestamp.digicert.com' -FilePath .\NewModule\NewModule.cat
```

This would be a good time to attempt Lab: Code Signing