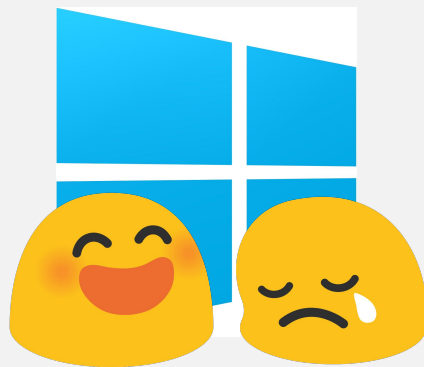




Windows 10
2 Steps Forward,
1 Step Back

James Forshaw @tiraniddo
Ruxcon 2015



Obligatory Background Slide

- Researcher in Google's Project Zero team
- Specialize in Windows
 - Especially local privilege escalation
- Never met a logical vulnerability I didn't like



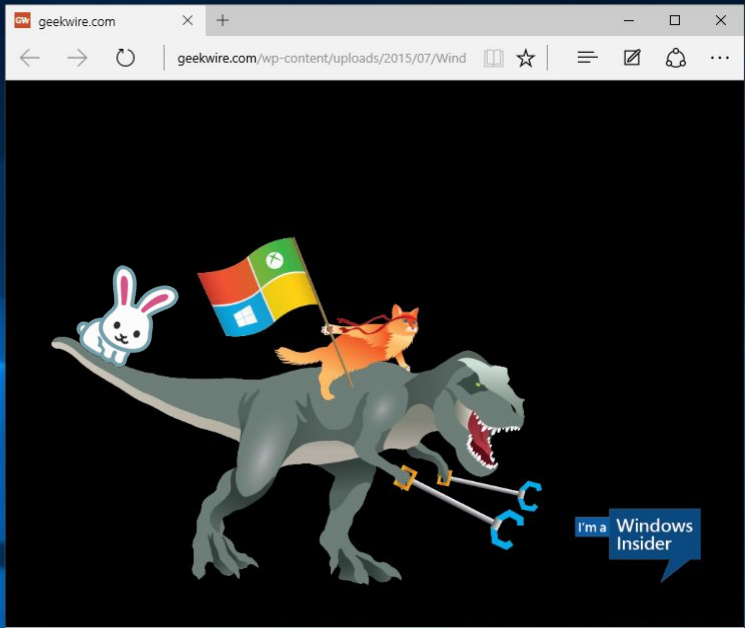
What I'm Going to Talk About



- Some research on Windows 10 from the early preview builds
- Why Windows 10 is awesome for security
- Except for when it isn't!
- Very much looking at things from a local privilege escalation perspective



Recycle Bin



user

Life at a glance

Play and explore

Most used

- Get Started
- Get Skype
- Maps
- People
- Calculator
- Alarms & Clock

Calendar	Mail	Xbox	Music	Film & TV
Microsoft Edge	Photos	Search	Money	News
Partly Sunny 15° 16° 12° London	Phone Compa...	OneNote	Store	Microsoft Solitaire Collection
				Get Office

File Explorer

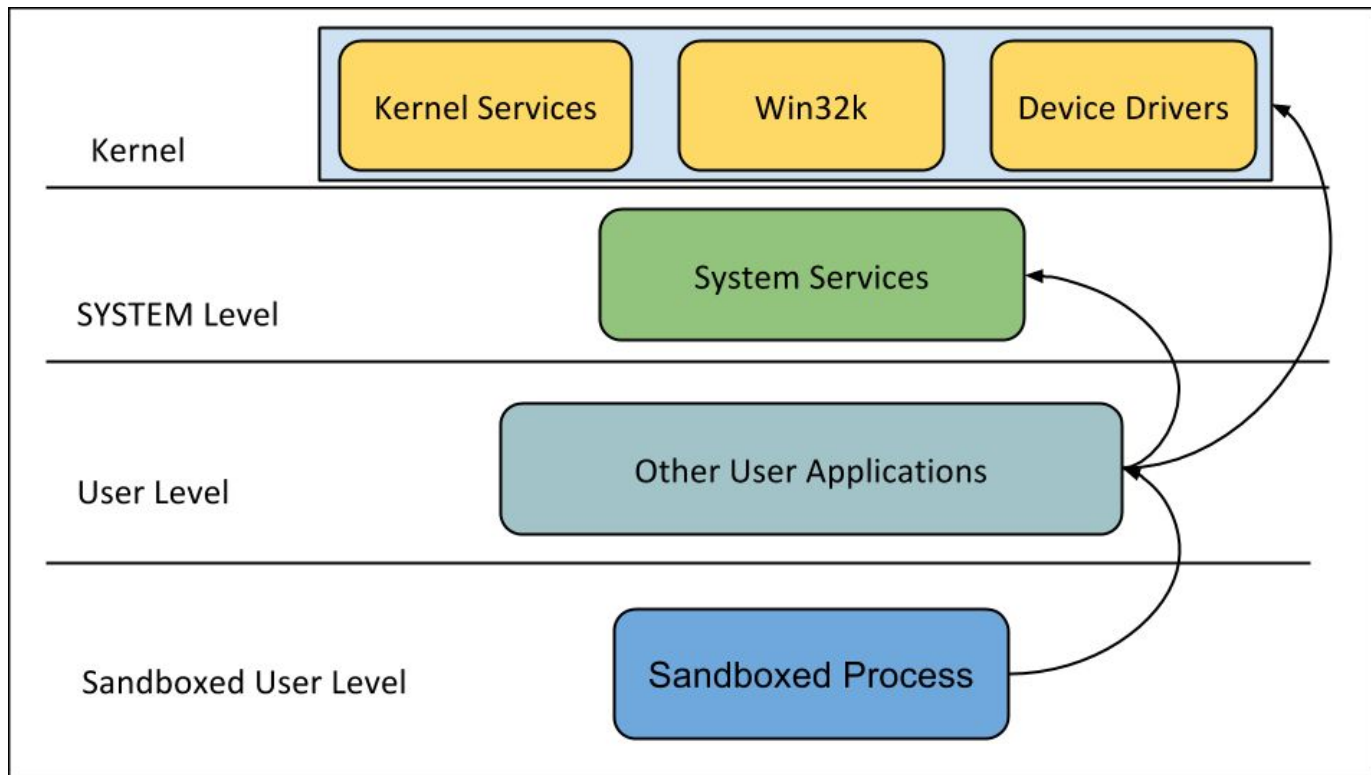
Settings

Power

All apps

New

Windows Local Attack Surface



Local System Vulnerabilities are Dead!



The screenshot shows a Windows Administrator command prompt window with the following text:

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>calc

C:\Windows\system32>
```

Overlaid on the command prompt is a Process Explorer window from Sysinternals. A dialog box is open in the foreground with the text:

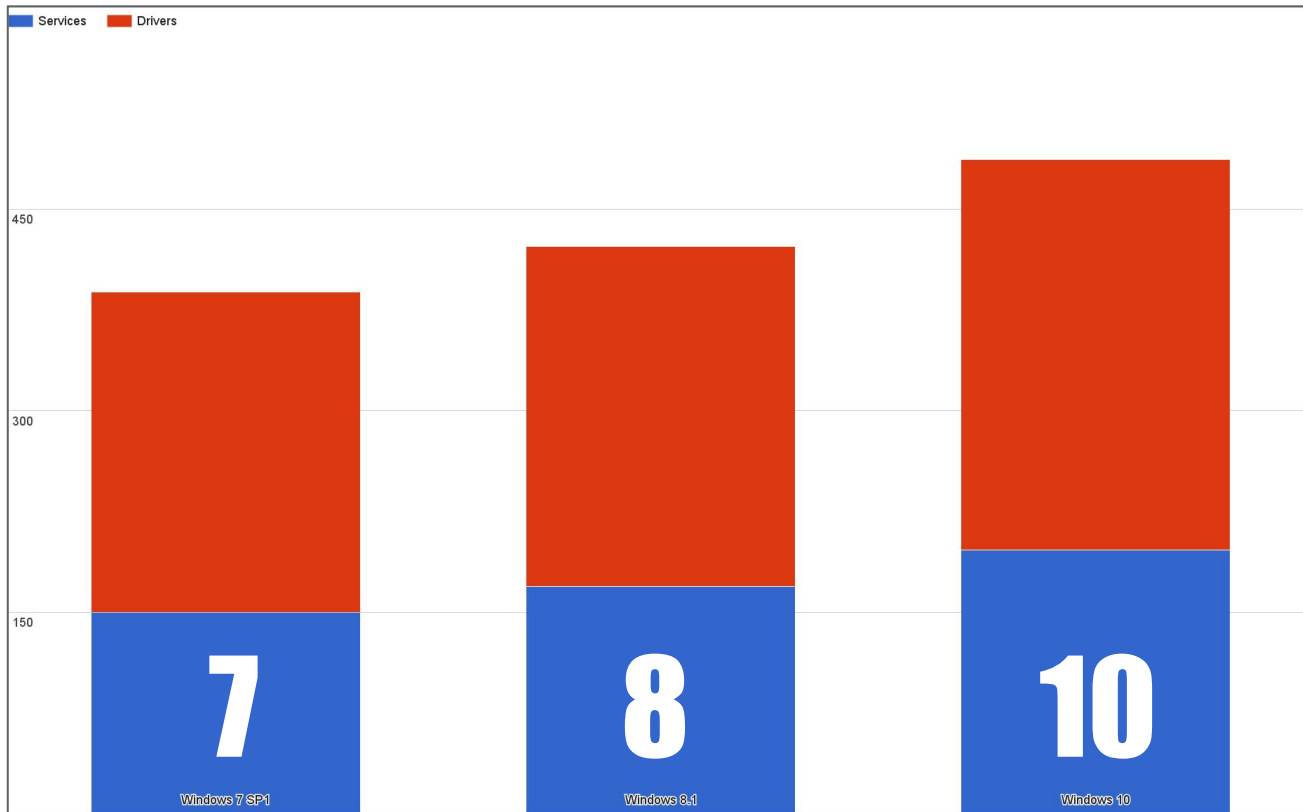
You'll need a new app to open this calculator

OK

The Process Explorer window shows a list of processes with columns for CPU, Private Bytes, Working Set, PID, and Desc. The status bar at the bottom indicates: CPU Usage: 6.11%, Commit Charge: 29.44%, Processes: 47, Physical Usage: 47.19%.

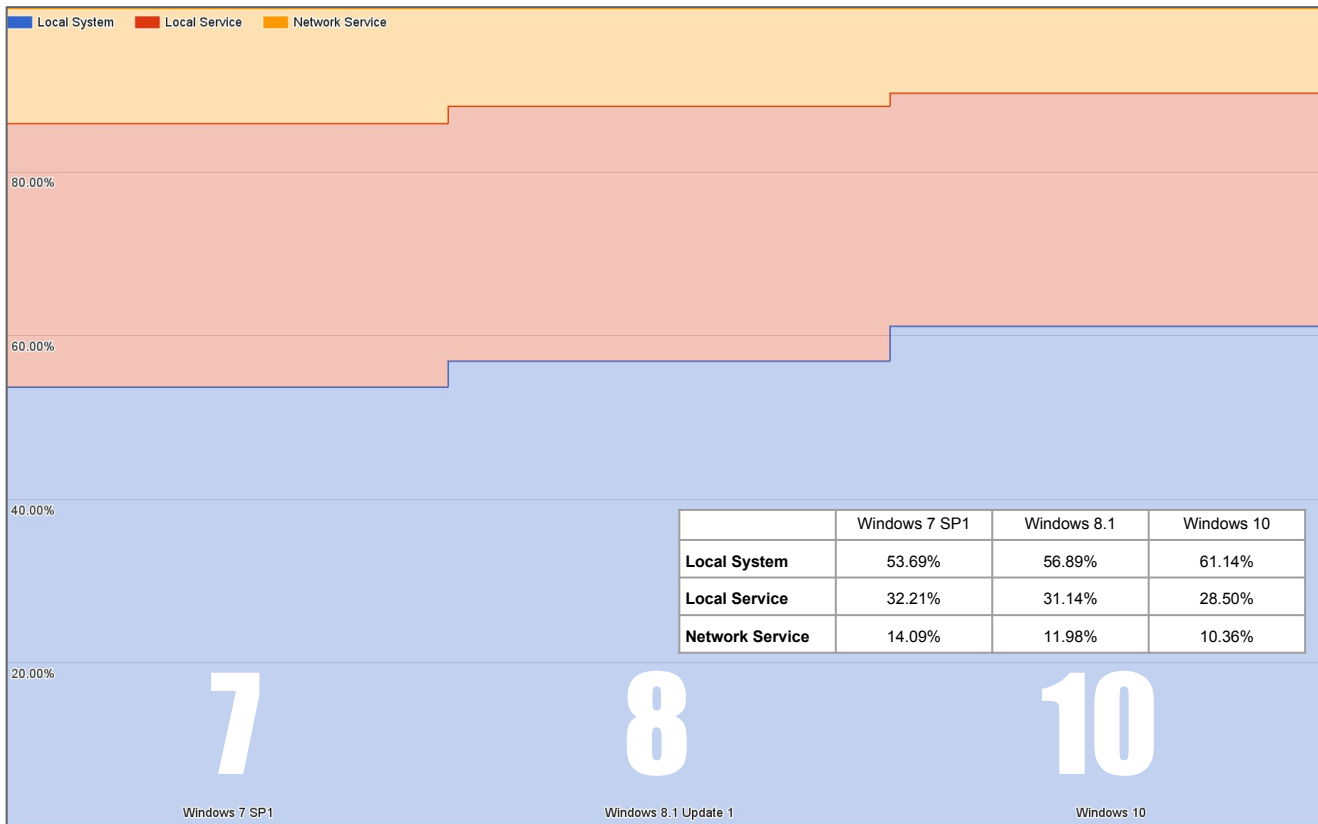
Process	CPU	Private Bytes	Working Set	PID	Desc
System Idle Process	93.89	0 K	4 K	0	
System		32 K	6,428 K	4	
smss.exe		0 K	0 K	n/a	Hardw
csrss.exe		28 K	784 K	268	Windc
wininit.exe		84 K	3,160 K	348	Client
services.exe		96 K	3,996 K	408	Windc
svchost.exe		96 K	5,752 K	512	Servic
svchost.exe		84 K	15,772 K	584	Host F
RuntimeBroker.exe		40 K	33,688 K	2156	Runtim
cmd.exe		20 K	3,152 K	1308	Windc
conhost.exe		64 K	15,252 K	2336	Conso
ShellExperienceHost.exe	Suspended	13,688 K	45,372 K	2528	Windc
SearchUI.exe	Suspended	62,196 K	110,288 K	2748	Search
WmiPrvSE.exe		1,764 K	7,428 K	4020	WMI
OpenWith.exe		4,840 K	23,844 K	3248	Pick a
svchost.exe		3,200 K	7,732 K	616	Host F
svchost.exe	0.01	21,364 K	36,384 K	768	Host F
taskhostw.exe		11,672 K	21,248 K	1664	Host F
sihost.exe		4,280 K	17,680 K	1244	Shell I
svchost.exe	0.01	35,784 K	43,516 K	832	Host F
VMBox.Service.exe		1,900 K	6,064 K	884	Virtual
svchost.exe		1,896 K	7,380 K	896	Host F
svchost.exe	0.01	11,816 K	17,248 K	904	Host F

System Services and Drivers



	Windows 7 SP1	Windows 8.1	Windows 10
Services	150	169	196
Drivers	238	253	291

Service Privilege Levels



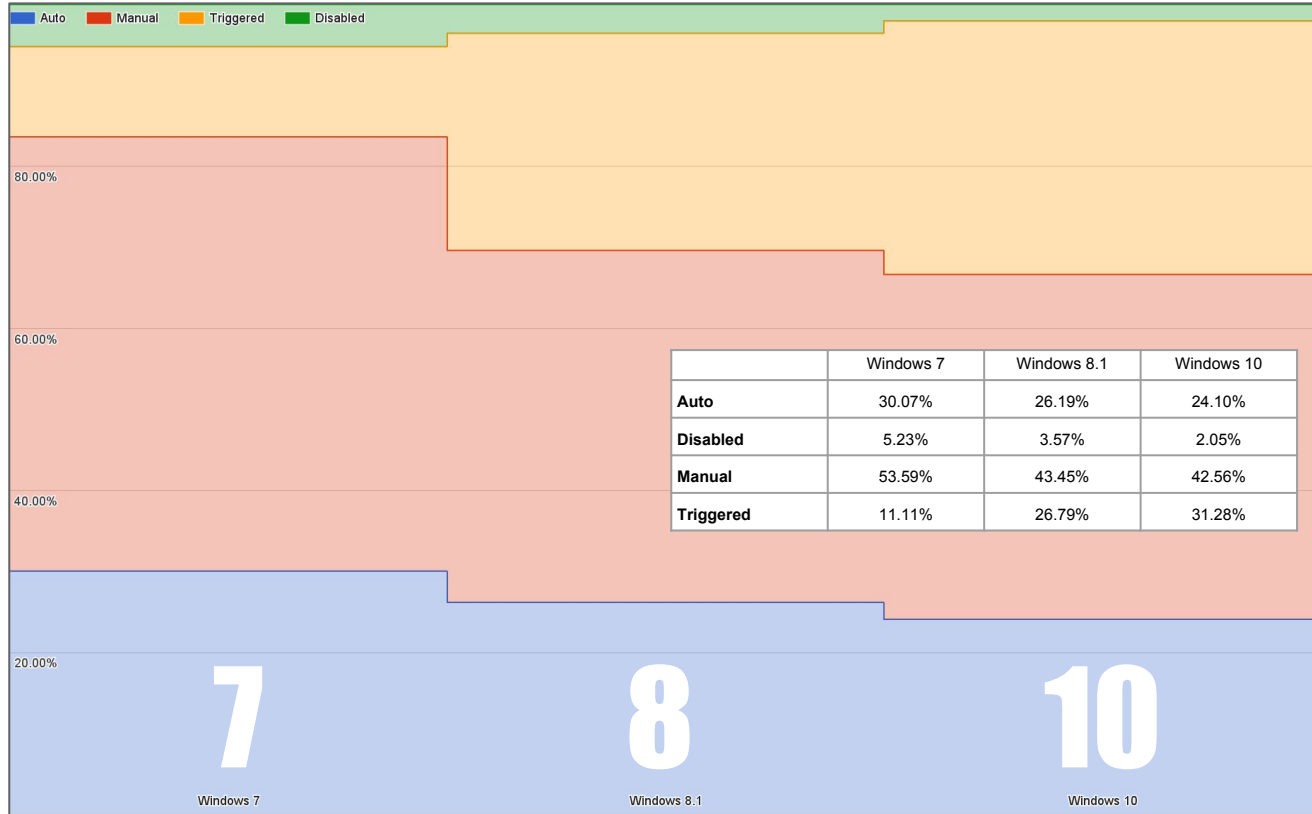
SVCHOST Running as User?



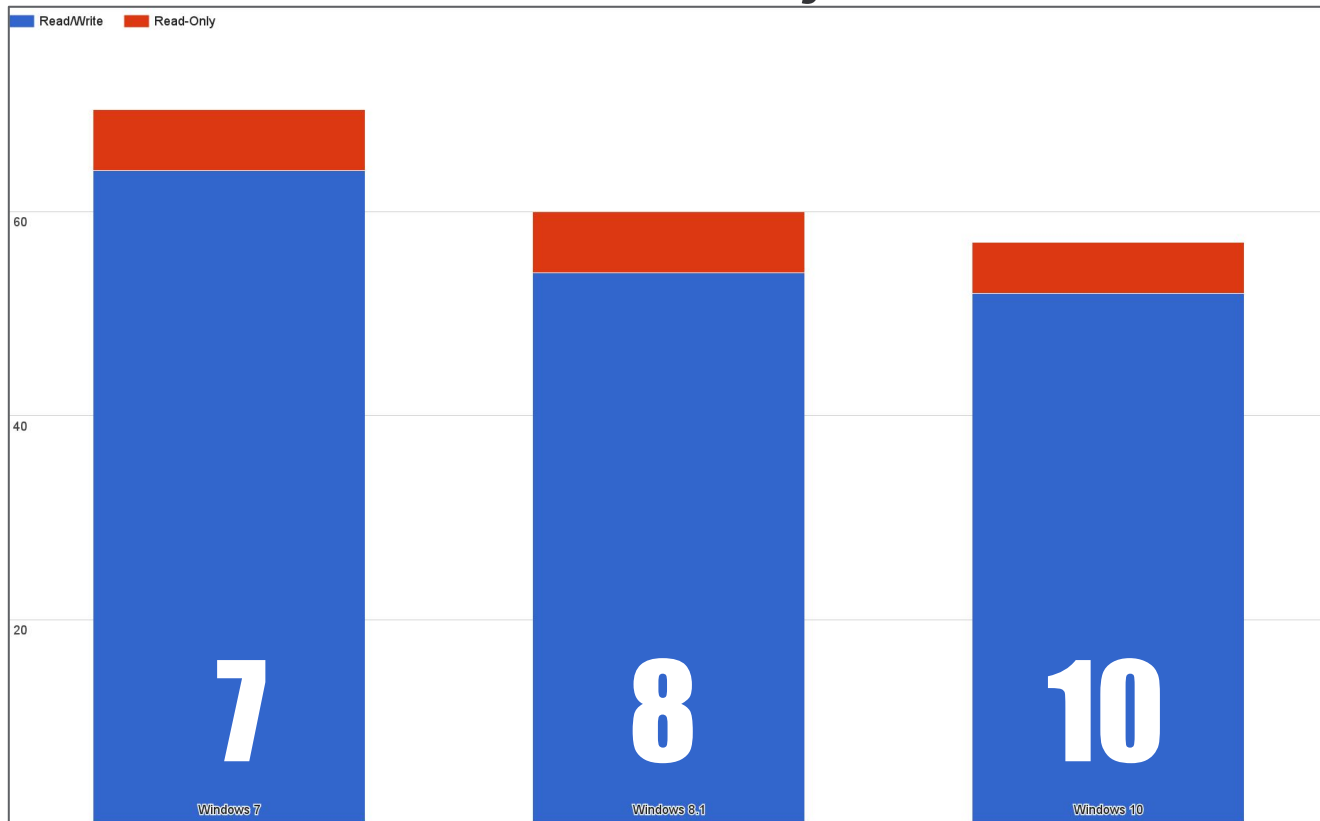
svchost.exe	NT AUTHORITY\LOCAL SERVICE
svchost.exe	NT AUTHORITY\NETWORK SERVICE
spoolsv.exe	NT AUTHORITY\SYSTEM
svchost.exe	NT AUTHORITY\LOCAL SERVICE
svchost.exe	NT AUTHORITY\SYSTEM
svchost.exe	NT AUTHORITY\SYSTEM
MsMpEng.exe	NT AUTHORITY\SYSTEM
SearchIndexer.exe	NT AUTHORITY\SYSTEM
svchost.exe	DESKTOP-F9T0PCQ\user
NisSrv.exe	NT AUTHORITY\LOCAL SERVICE
svchost.exe	NT AUTHORITY\SYSTEM
lsass.exe	NT AUTHORITY\SYSTEM

Malware?
Nope!

Service Start Mode

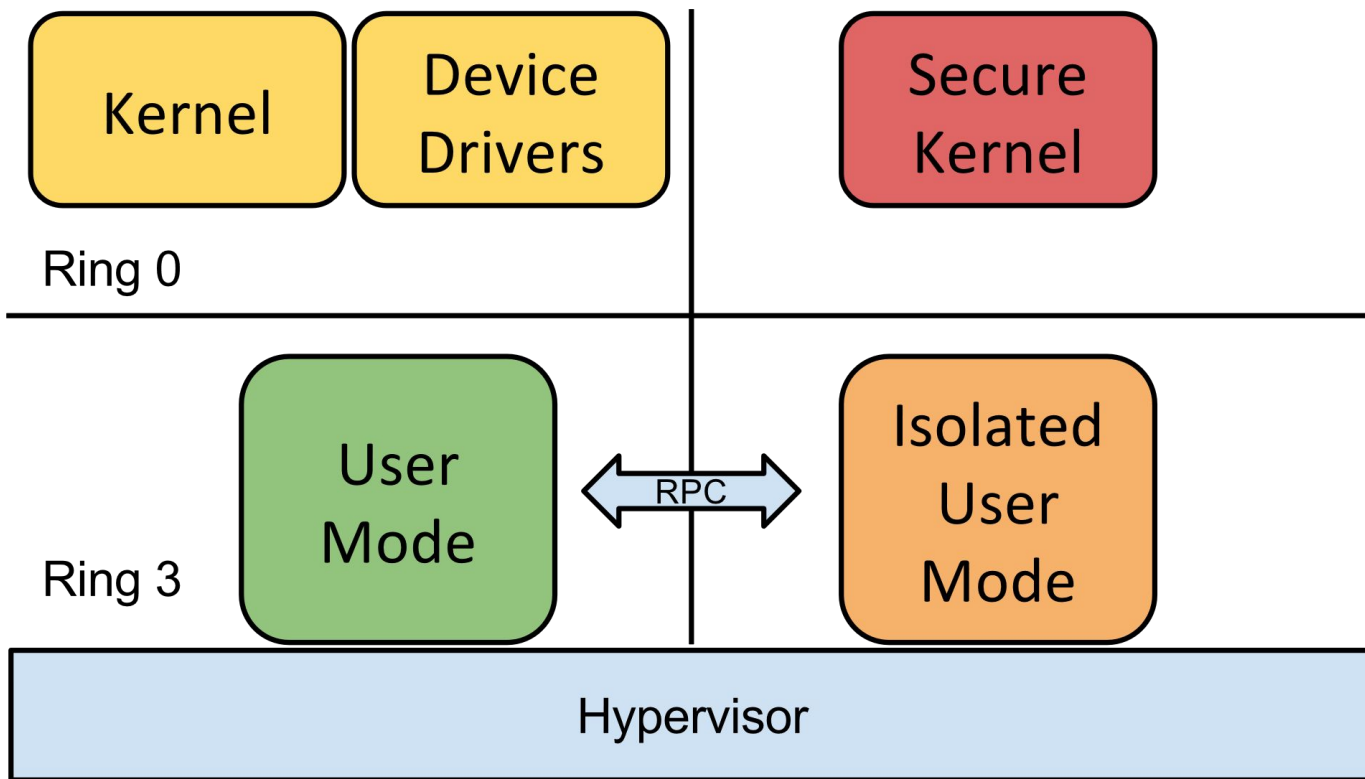


Accessible Device Objects



	Windows 7	Windows 8.1	Windows 10
Read/Write	64	54	52
Read-Only	6	6	5

Isolated User Mode



Isolated LSASS



```
mimikatz 2.0 alpha x64 (oe.eo)
Authentication Id : 0 ; 14627436 (00000000:00df326c)
Session          : Interactive from 3
User Name        : Johan
Domain           : VIAMONSTRA
Logon Server     : DC01
Logon Time       : 6/20/2015 4:52:57 PM
SID              : S-1-5-21-219932437-4172800940-1518705232-1117

msv :
[00000003] Primary
* Username : Johan
* Domain   : VIAMONSTRA
* Flags    : I01/N01/L00/S01
* LSA Isolated Data: NtlmHash
Unk-Key   : a698ec9ad1063bbd7dc1244b0ab0a621f4bc81efb957342eca0f74e1304f24abec5c368e9257678838b7613843bbfa6
Encrypted: f3222d9994ce547133f5688e14d83fe1d7da2e74063f7f6378951e7f0a17ee32f1475c9ea4b010214ddb748f8c9a3746
344fe203

[00010000] CredentialKeys
* RootKey : 1899d56912dd70428d72f297eeae26444e96ec401354d034795831641a23d0d
* DPAPI   : fc2278b6029da175a614fc3597f0ed8e

tspkg :
wdigest :
* Username : Johan
* Domain   : VIAMONSTRA
* Password : (null)

kerberos :
* Username : Johan
* Domain   : CORP.VIAMONSTRA.COM
* Password : (null)

ssp : KO
credman :
```

```
C:\Windows\system32\cmd.exe
C:\>whoami
pc0003\admin
C:\>
```

Image from <http://deploymentresearch.com/Research/Post/490/Enabling-Virtual-Secure-Mode-VSM-in-Windows-10-Enterprise-Build-10130>

But Sadly

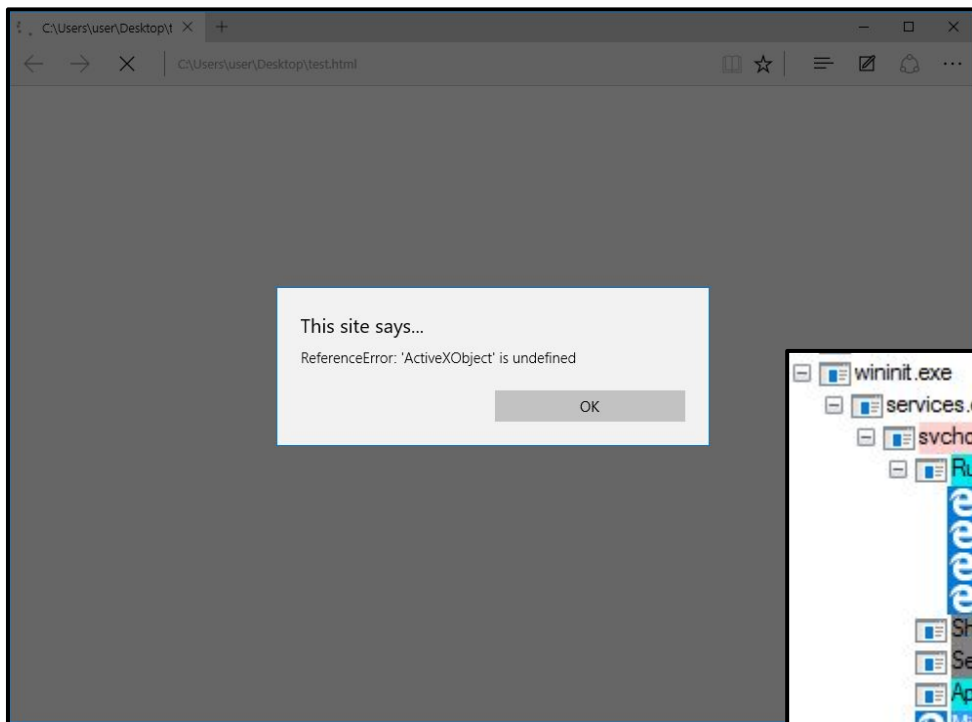


- Not available in consumer builds only Enterprise
- Can't use your own code to isolate anything
- Very restrictive use

Edge Browser



Microsoft Edge Security

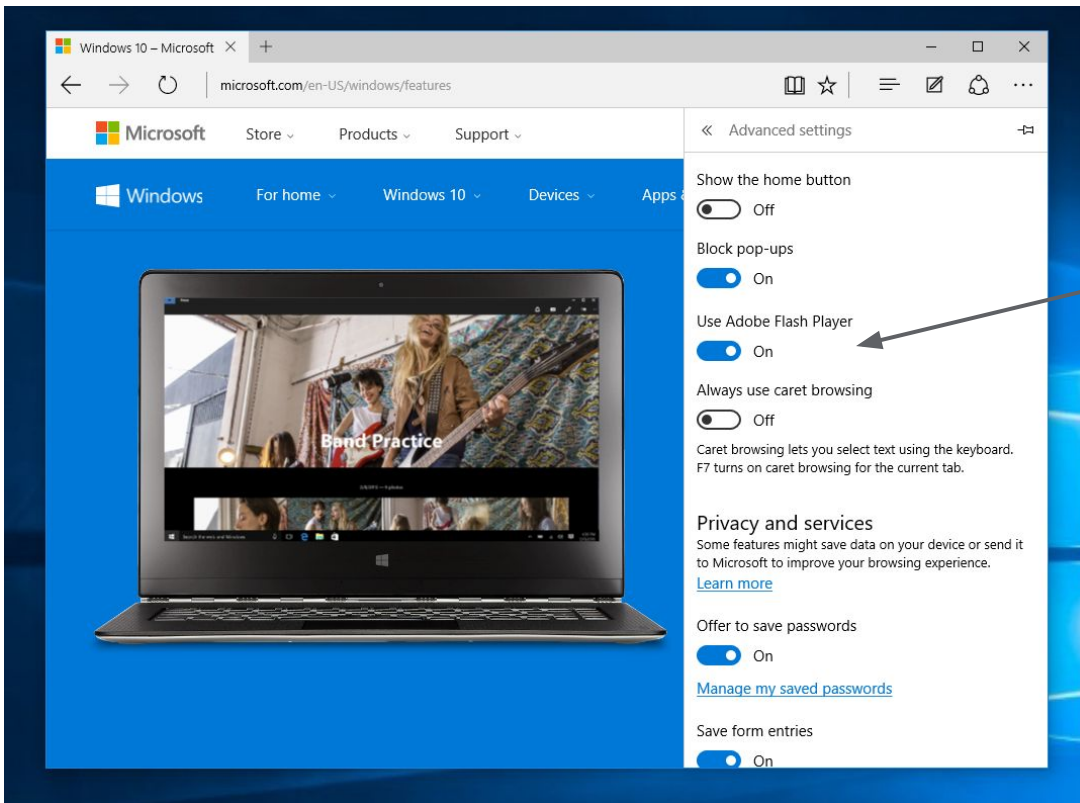


ActiveX is gone(ish)

AppContainer Sandbox Always On

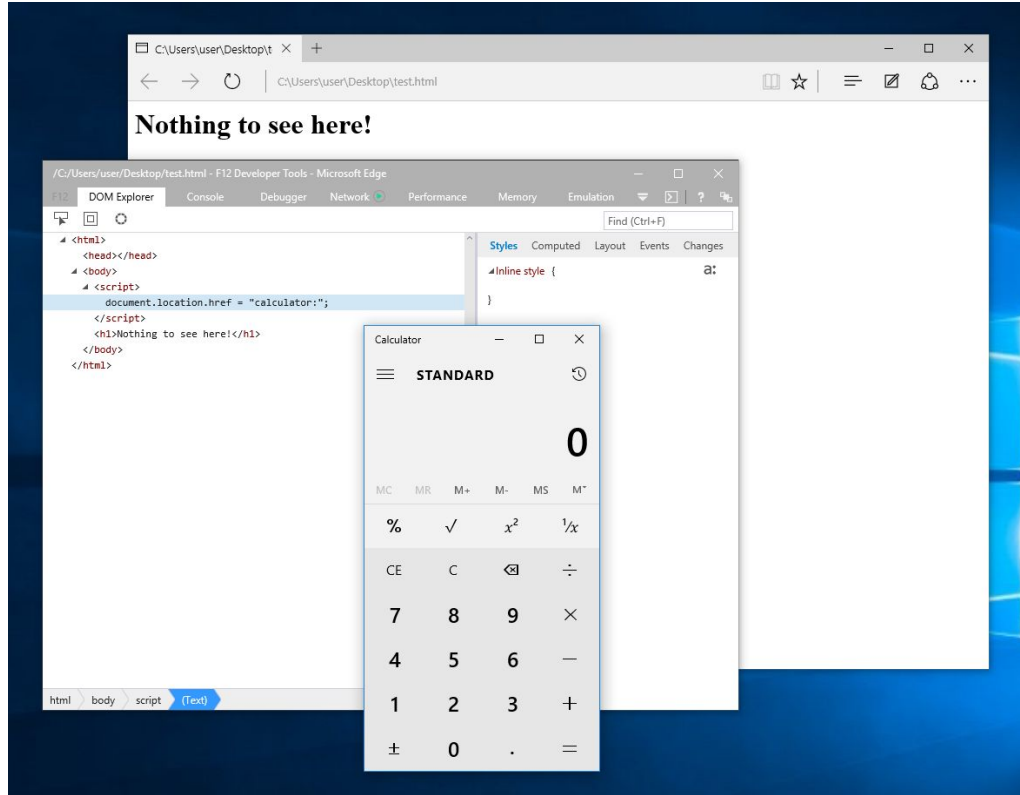
wininit.exe	System
services.exe	System
svchost.exe	System
RuntimeBroker.exe	Medium
Microsoft EdgeCP.exe	AppContainer
Microsoft EdgeCP.exe	AppContainer
Microsoft EdgeCP.exe	AppContainer
Microsoft EdgeCP.exe	AppContainer
ShellExperienceHost.exe	AppContainer
SearchUI.exe	AppContainer
ApplicationFrameHost.exe	Medium
Microsoft Edge.exe	AppContainer
browser_broker.exe	Medium

Microsoft Edge and Flash



Nope!

Has No One Learnt from the Past?



Guess Trident Wasn't a Suitable Base?

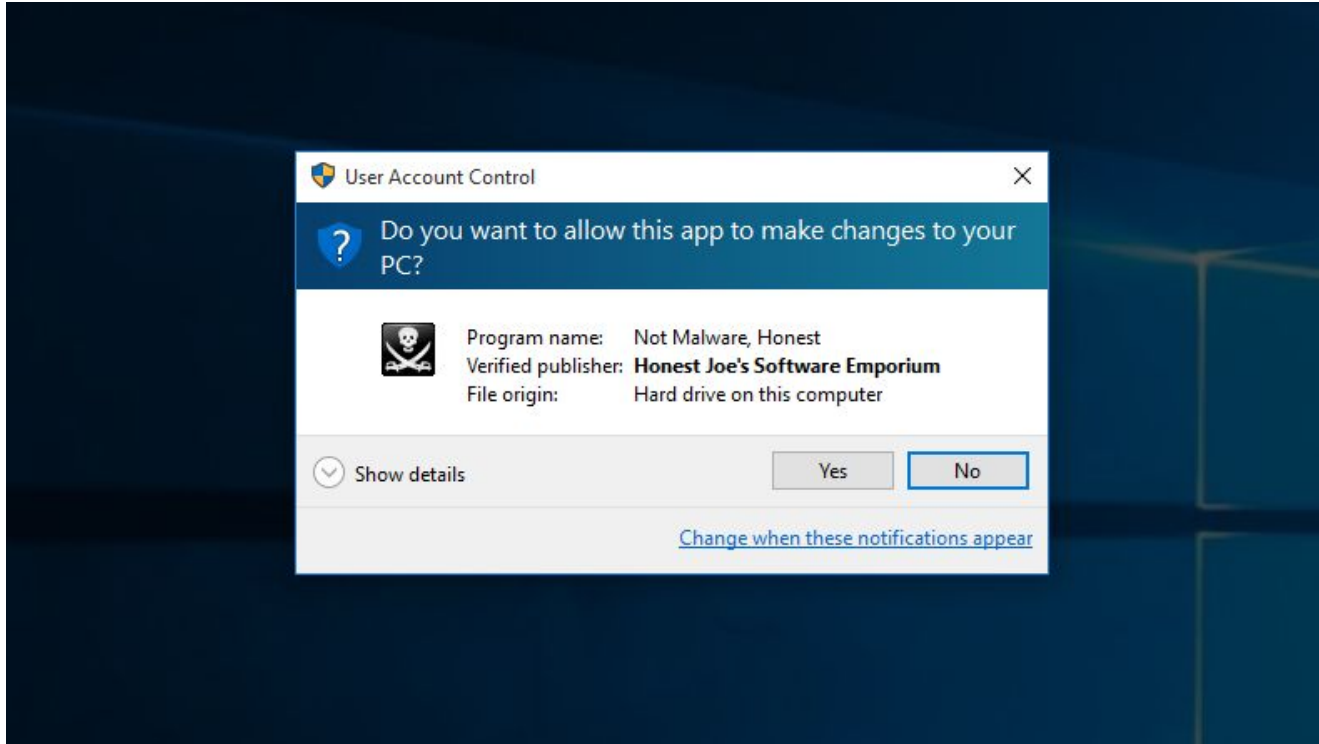


```
h0wl / edge_crash.html
Last active 9 days ago

edge_crash.html Raw

1 <!-- based on https://connect.microsoft.com/IE/feedback/details/1683347/ms-edge-combination-of-iframe-anchor-h
2 <!doctype html>
3 <html>
4 <head>
5 <script>
6 function boom() {
7     var iframe = document.getElementById('iframe1');
8     document.location.href = '#';
9     iframe.parentNode.removeChild(iframe);
10    alert(history.state);
11 }
12 </script>
13 <body>
14 <div id="div1">
15 <iframe id="iframe1" width="400" height="400" src="http://bing.com"></iframe>
16 </div>
17 Perform any interaction within the iframe (such as click a link) and after that click the crash button
18 <input type="button" value="crash" onclick="boom();">
19 </body>
20 </html>
```

User Account Control



They've Fixed Some Bugs I've Reported



<https://code.google.com/p/google-security-research/issues/detail?id=156>

★ Issue 156: Windows: Task Scheduler S4U Logon Elevation of Privilege

7 people starred this issue and may be notified of changes.

Status: WontFix

Owner:

Closed: Jan 2015

Cc:

PublicOn-2015-Jan-16

Vendor-Microsoft

Product-Windows

Severity-High

Finder-forshaw

Reported-2014-Nov-5

CCProjectZeroMembers

Deadline-90

MSRC-20968

Windows: Task Scheduler S4U Logon Elevation of Privilege
Platform: Windows 8.1 Update 32/64 bit (7 not tested)
Class: Elevation of Privilege

The windows task scheduler allows a split token administrator to register a task which runs as a batch job from a

★ Issue 220: Windows: AppInfo AiCheckSecureApplicationDirectory Bypass

2 people starred this issue and may be notified of changes.

Status: WontFix

Owner:

Closed: Feb 2015

Cc:

Vendor-Microsoft

Product-Windows

Severity-Medium

Finder-forshaw

Reported-2014-Dec-15

CCProjectZeroMembers

Deadline-90

MSRC-21233

Windows: AppInfo AiCheckSecureApplicationDirectory Bypass
Platform: Windows 8.1 Update, Windows 7
Class: Security Bypass/EoP

Summary:

The AppInfo service handles requests for UAC elevation. There's an issue with the checking of secure directories which allows a user to install a UIAccess application without requiring full access to a secure directory leading to the potential for EoP

Description:

When the AppInfo service launches a UIAccess or auto elevated application it checks that the location of the executable file is in certain secure directories. For example it checks \Windows and \Windows\System32 as well as Program Files. However because some directories are writable by normal users it explicitly excludes them. This

<https://code.google.com/p/google-security-research/issues/detail?id=220>

UAC Auto Elevation Directory Check



c:\windows\ ↓	app.exe
------------------	---------

ALLOWED

c:\windows\tracing\ ↓	app.exe
--------------------------	---------

BANNED

Folder Permissions



Advanced Security Settings for Windows

Name: C:\Windows
Owner: TrustedInstaller Change

Permissions Auditing Effective Access

User/ Group: limiteduser (USER-WIN81-PC\limiteduser) Select a user

View effective access

Effective access	Permission	Access limited by
✗	Full control	File Permissions
🔓	Traverse folder / execute file	
🔓	List folder / read data	
🔓	Read attributes	
🔓	Read extended attributes	
✗	Create files / write data	File Permissions
✗	Create folders / append data	File Permissions
✗	Write attributes	File Permissions
✗		
✗		

Apply

✗	Create files / write data
✗	Create folders / append data
✗	Write attributes
✗	Write extended attributes

Advanced Security Settings for tracing

Name: C:\Windows\tracing
Owner: SYSTEM Change

Permissions Auditing Effective Access

User/ Group: limiteduser (USER-WIN81-PC\limiteduser) Select a user

View effective access

Effective access	Permission	Access limited by
✗	Full control	File Permissions
🔓	Traverse folder / execute file	
🔓	List folder / read data	
🔓	Read attributes	
🔓	Read extended attributes	
🔓	Create files / write data	
🔓	Create folders / append data	
🔓	Write attributes	
✗		

🔓	Create files / write data
🔓	Create folders / append data
🔓	Write attributes
🔓	Write extended attributes

AiCheckSecureApplicationDirectory Bypass



- Need to be able to write a file with a secure path
- How can we write to C:\Windows without writing to C:\Windows?

c:\windows\	malicious.exe
-------------	---------------



ALLOWED

c:\windows\	????
-------------	------



ALLOWED?

NTFS Alternate Data Streams FTW!



c:\windows\ tracing:malicious.exe



ALLOWED

- Only need FILE_WRITE_DATA/FILE_ADD_FILE access right on directory to created named stream.

Didn't Fix All my UAC Bypasses Though



★ **Issue 219: Windows: NtUserGetClipboardAccessToken Token Leak**
2 people starred this issue and may be notified of changes.

<p>Status: Fixed Owner: Closed: Mar 10 Cc: Vendor: Microsoft Product: Windows-Kernel Severity: medium Finder: forshaw Reported: 2014-Dec-15 CC: ProjectZeroMembers Deadline: 90 MSRC: 21232 CVE: 2015-0078 MS15: 023</p>	<p>Windows: NtUserGetClipboardAccessToken Token Leak Platform: Windows 8.1 Update (Windows 7 not vulnerable) Class: Security Bypass/EoP</p> <p>Summary: The NtUserGetClipboardAccessToken win32k system call exposes the access token of the last user to lower-privileged users. It can also be used to open an anonymous impersonation thread token which normally OpenThreadToken shouldn't be able to do.</p> <p>Description: The NtUserGetClipboardAccessToken method opens a token object which is captured during the NtUserCloseClipboard system call assuming that the caller has written something to the clipboard.</p> <p>The actual code goes something like:</p>
--	---

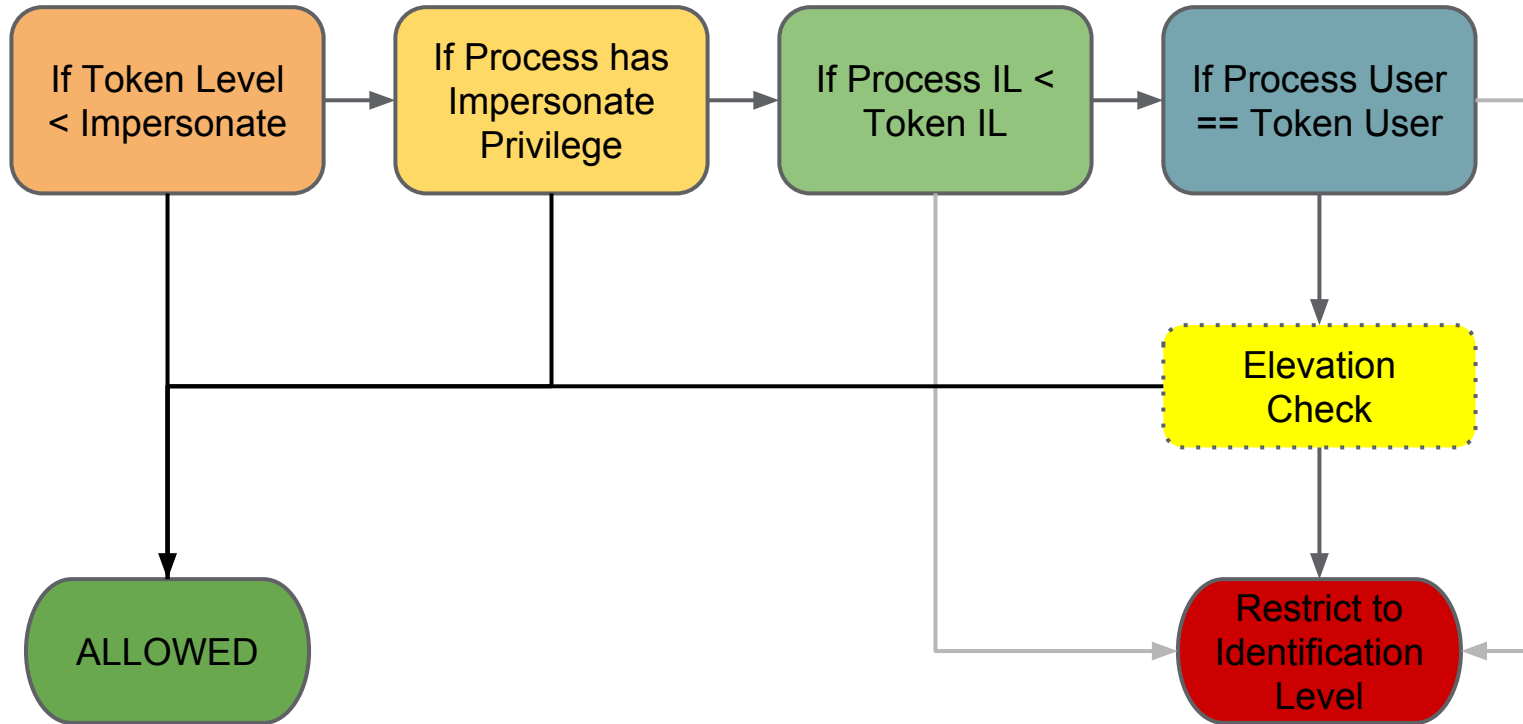
<https://code.google.com/p/google-security-research/issues/detail?id=219>



DEMO

Elevated Token Capture

Well MS Almost Did



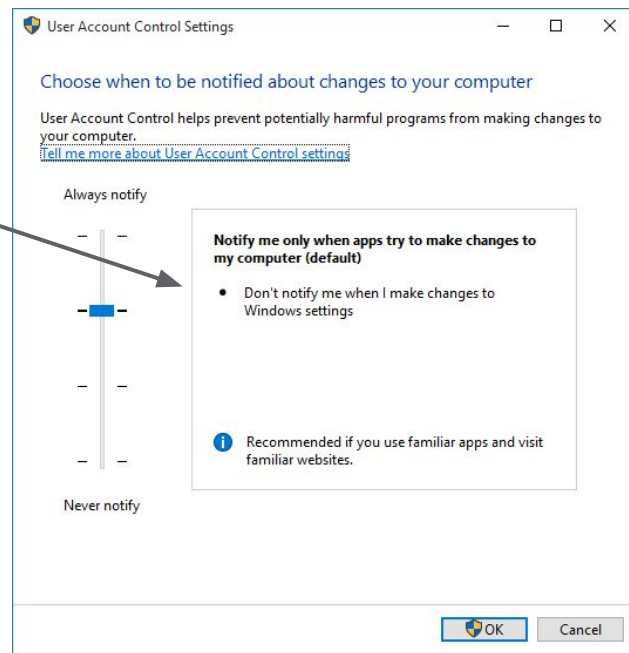
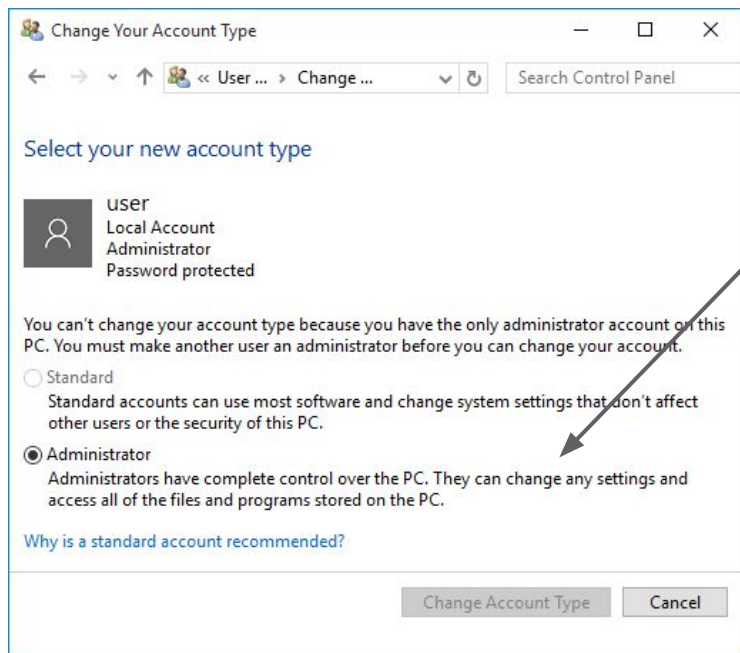
Elevated Token Impersonation



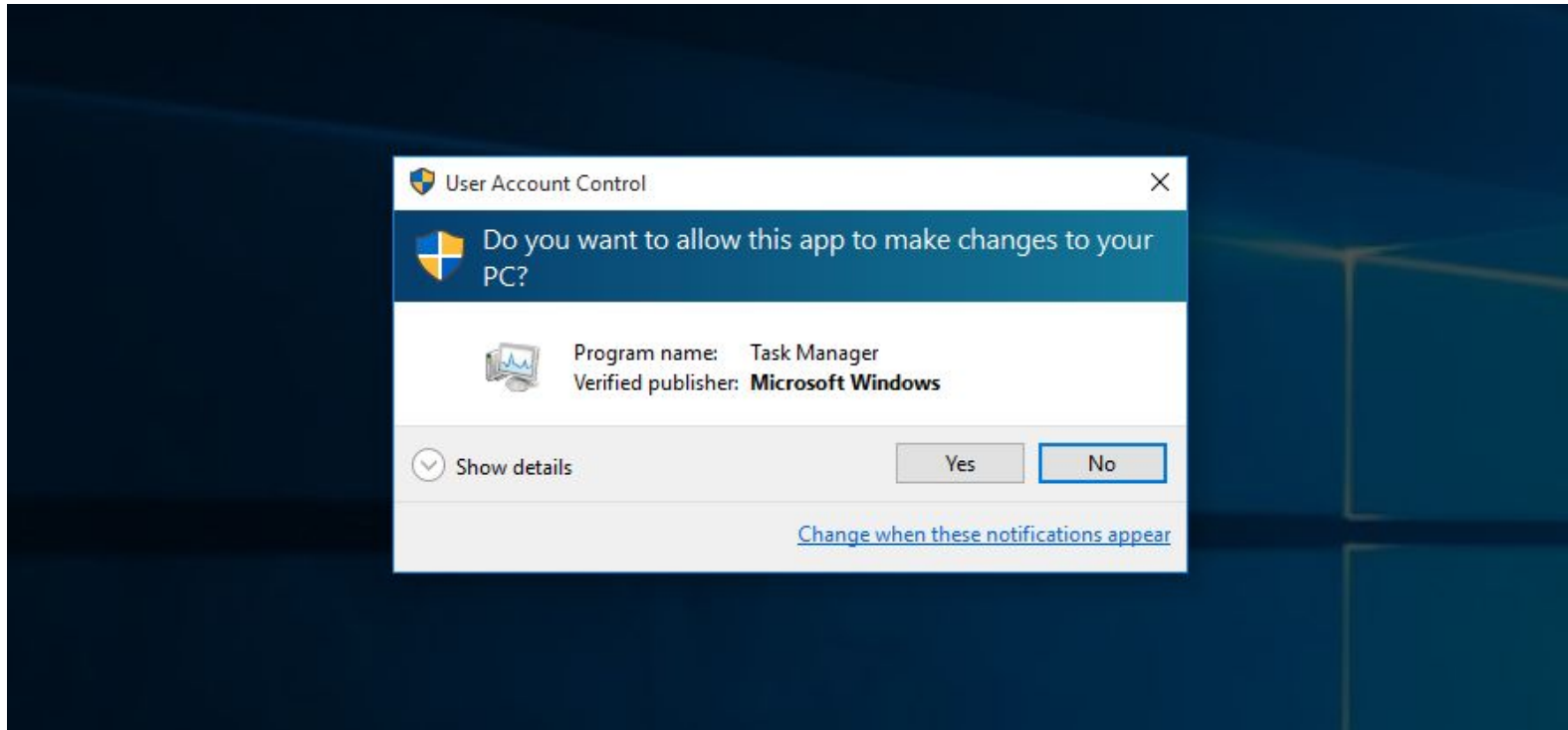
- Blocks impersonating an elevated token unless process token is also elevated
- Must be enabled in SeCompatFlags kernel flag

```
if (SeTokenIsElevated(ImpersonationToken)) {  
    if ((SeCompatFlags & 1)  
        && !SeTokenIsElevated(ProcessToken)) {  
        return STATUS_PRIVILEGE_NOT_HELD;  
    }  
}
```

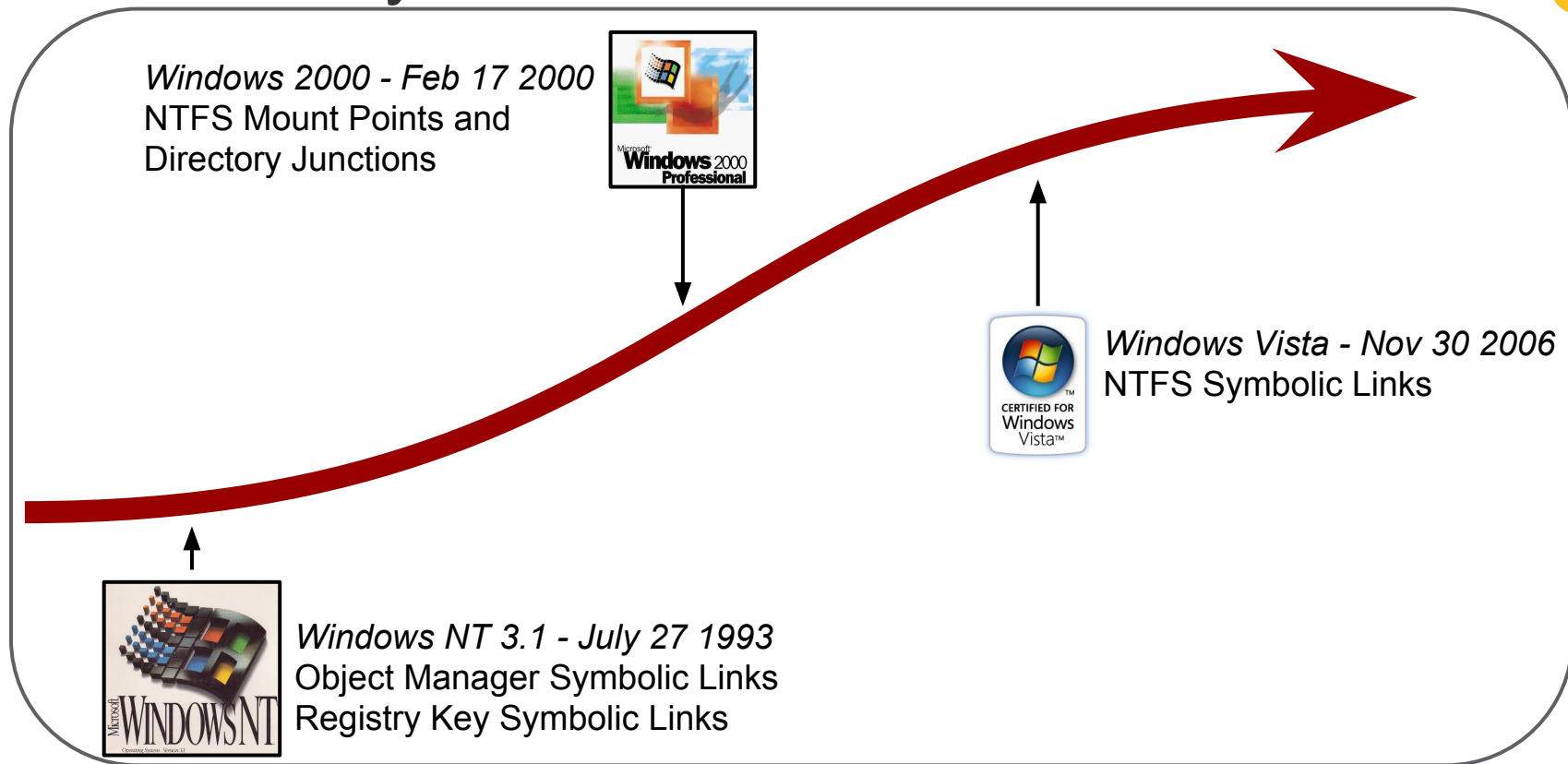
In The End Still the “Wrong” Default IMO!



If You Change Task Manager Needs a Prompt



Windows Symbolic Links

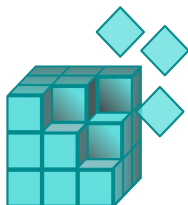


Mitigated in Sandboxes



NTFS Mount Points

LIMITED



Registry Key Symbolic Links

BANNED



Object Manager Symbolic Links

LIMITED

Mitigations Backported



Microsoft Security Bulletin MS15-090 - Important

2 out of 3 rated this helpful - [Rate this topic](#)

Vulnerabilities in Microsoft Windows Could Allow Elevation of Privilege (3060716)

Published: August 11, 2015

Version: 1.0

▲ Executive Summary

This security update resolves vulnerabilities in Microsoft Windows. The vulnerabilities could allow elevation of privilege if an attacker logs on to an affected system and runs a specially crafted application or convinces a user to open a specially crafted file that invokes a vulnerable sandboxed application, allowing an attacker to escape the sandbox.

This security update is rated Important for all supported releases of Microsoft Windows except Windows 10, which is not affected. For more information, see the **Affected Software** section.

The security update addresses the vulnerabilities by correcting how Windows Object Manager handles object symbolic links created by a sandbox process, by preventing improper interaction with the registry by sandboxed applications, and by preventing improper interaction with the filesystem by sandboxed applications. For more information about the vulnerabilities, see the **Vulnerability Information** section.

Mount Point Mitigation Bypass



```
NTSTATUS IopXxxControlFile(...) {
    if (CtlCode == FSCTL_SET_REPARSE_POINT) {
        PREPARSE_DATA_BUFFER buffer = ...
        if (NumberOfBytes >= 4 &&
            buffer->ReparseTag == IO_REPARSE_TAG_MOUNT_POINT &&
            RtlIsSandboxedToken(&SubjectSecurityContext, AccessMode) {
            status = FsRtlValidateReparsePointBuffer(NumberOfBytes, buffer);
            if (!NT_SUCCESS(status)) {
                return status
            }
            name.Length = name.MaximumLength = buffer->SubstituteNameLength;
            name.Buffer = &buffer->PathBuffer[0];
            InitializeObjectAttributes(&obja, &name, OBJ_FORCE_ACCESS_CHECK|OBJ_KERNEL_HANDLE);
            status = ZwOpenFile(&FileHandle, FILE_GENERIC_WRITE, &obja, ..., FILE_DIRECTORY_FILE);
            if (!NT_SUCCESS(status)) {
                return status;
            }
            ZwClose(FileHandle);
        }
    }
}
```

Mount Point Mitigation Bypass



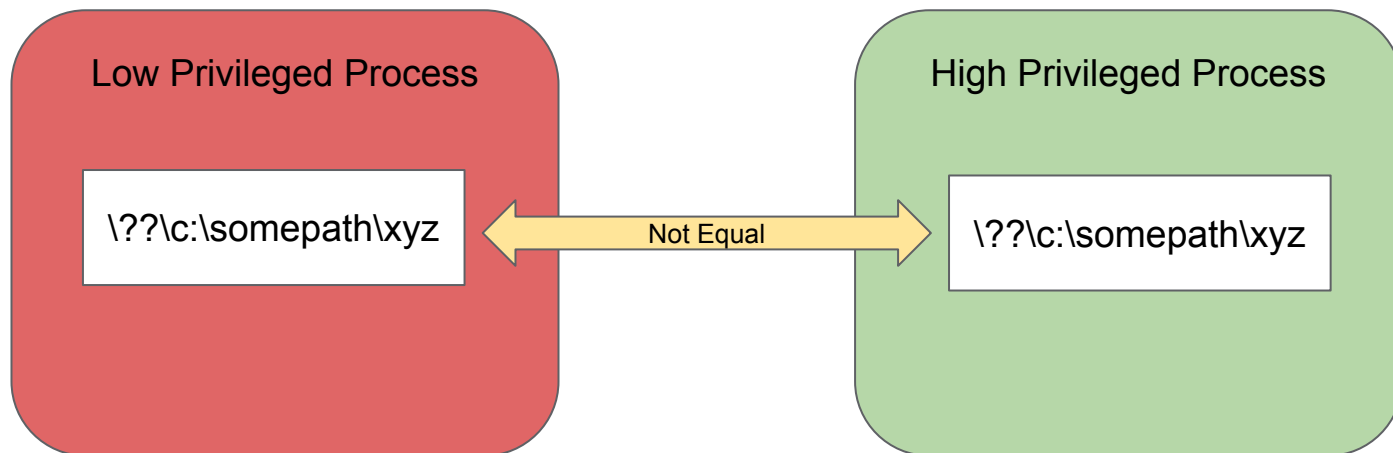
```
NTSTATUS IopXxxControlFile(...) {
    if (CtlCode == FSCTL_SET_REPARSE_POINT) {
        PREPARSE_DATA_BUFFER buffer = ...
        if (NumberOfBytes >= 4 &&
            buffer->ReparseTag == IO_REPARSE_TAG_MOUNT_POINT &&
            RtlIsSandboxedToken(&SubjectSecurityContext, AccessMode) {
            status = FsRtlValidateReparsePointBuffer(NumberOfBytes, buffer);
            if (!NT_SUCCESS(status)) {
                return status;
            }
            name.Length = name.MaximumLength = buffer->SubstituteNameLength;
            name.Buffer = &buffer->PathBuffer[0];
            InitializeObjectAttributes(&obja, &name, OBJ_FORCE_ACCESS_CHECK|OBJ_KERNEL_HANDLE);
            status = ZwOpenFile(&FileHandle, FILE_GENERIC_WRITE, &obja, ..., FILE_DIRECTORY_FILE);
            if (!NT_SUCCESS(status)) {
                return status;
            }
            ZwClose(FileHandle);
        }
    }
}
```

Mount Point Mitigation Bypass

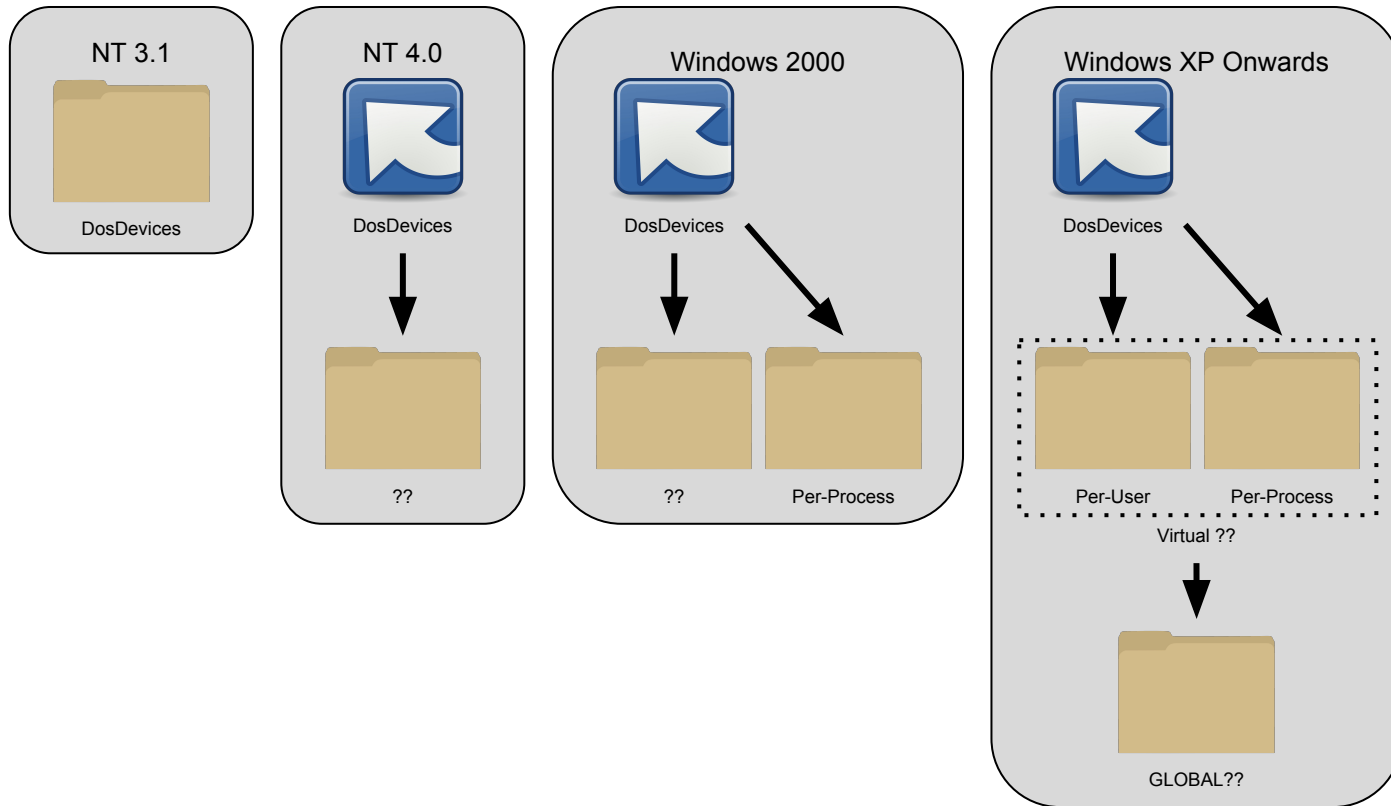


```
NTSTATUS IopXxxControlFile(...) {
    if (CtlCode == FSCTL_SET_REPARSE_POINT) {
        PREPARSE_DATA_BUFFER buffer = ...
        if (NumberOfBytes >= 4 &&
            buffer->ReparseTag == IO_REPARSE_TAG_MOUNT_POINT &&
            RtlIsSandboxedToken(&SubjectSecurityContext, AccessMode) {
            status = FsRtlValidateReparsePointBuffer(NumberOfBytes, buffer);
            if (!NT_SUCCESS(status)) {
                return status;
            }
            name.Length = name.MaximumLength = buffer->SubstituteNameLength;
            name.Buffer = &buffer->PathBuffer[0];
            InitializeObjectAttributes(&obja, &name, OBJ_FORCE_ACCESS_CHECK|OBJ_KERNEL_HANDLE);
            status = ZwOpenFile(&FileHandle, FILE_GENERIC_WRITE, &obja, ..., FILE_DIRECTORY_FILE);
            if (!NT_SUCCESS(status)) {
                return status;
            }
            ZwClose(FileHandle);
        }
    }
}
```

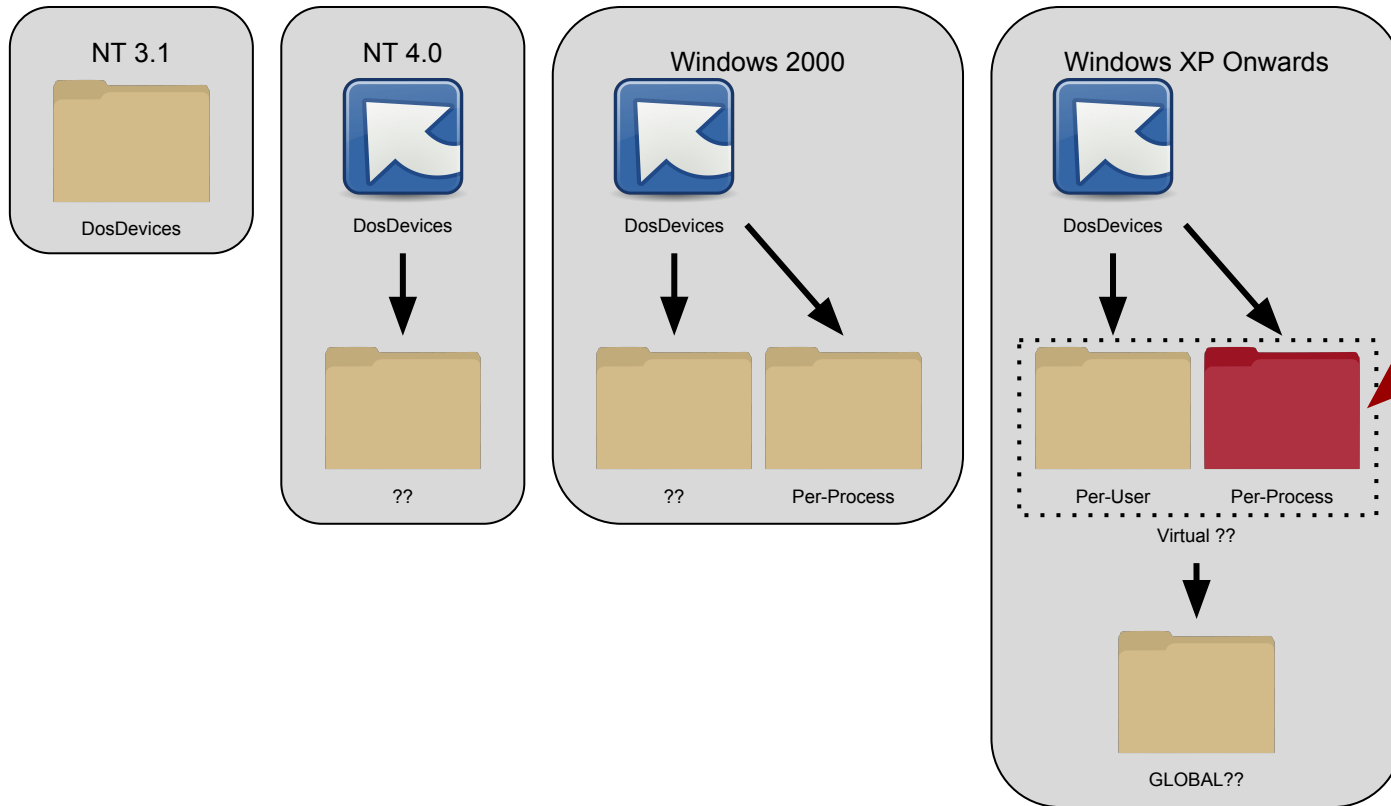
Time of check-Time of use



DosDevices History

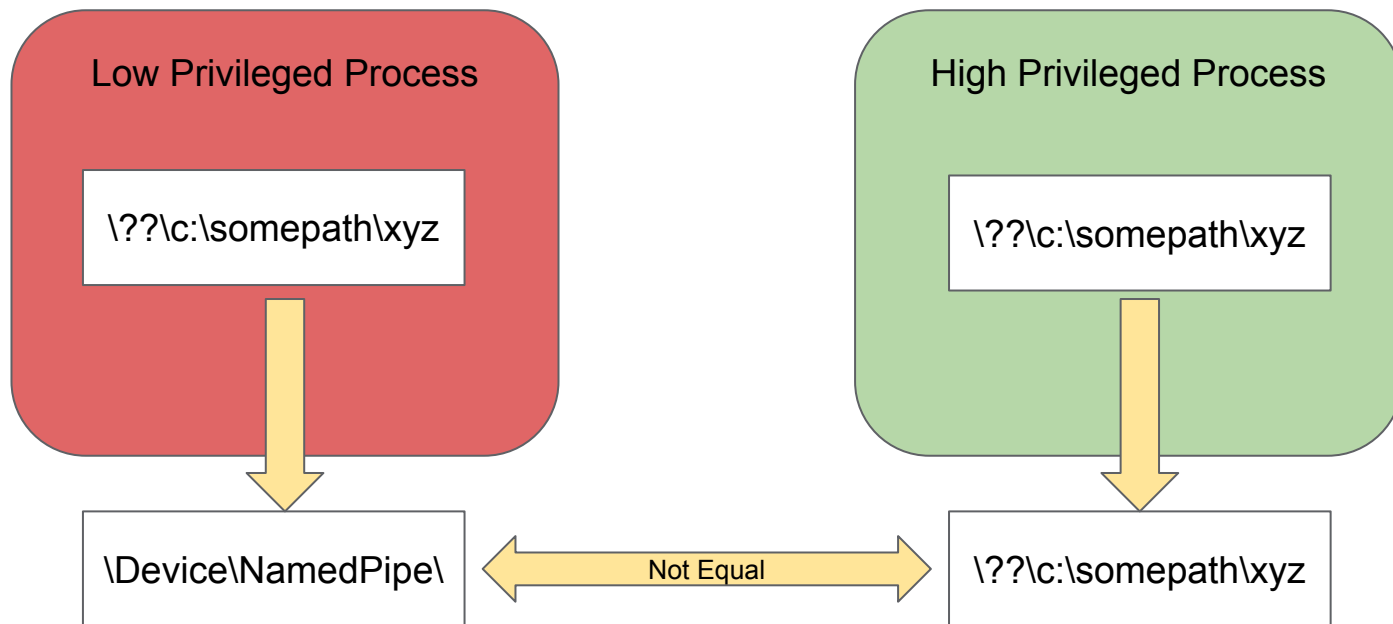


DosDevices History



Use This!

Abusing Per-Process Device Map



<https://code.google.com/p/google-security-research/issues/detail?id=486>

Sandbox Winter is Coming!



xrefs to RtlIsSandboxedToken

Direction	Type	Address	Text
Up	p	ObpParseSymbolicLink+17B	call RtlIsSandboxedToken
Up	p	NtCreateSymbolicLinkObject+177	call RtlIsSandboxedToken
Do...	p	CmSetValueKey+7F8	call RtlIsSandboxedToken
Do...	p	IopXxxControlFile+121C	call RtlIsSandboxedToken
Do...	p	NtDuplicateToken+137	call RtlIsSandboxedToken
Do...	p	NtDuplicateToken+14C	call RtlIsSandboxedToken
Do...	p	NtSetInformationProcess+1279	call RtlIsSandboxedToken
Do...	p	CmpCheckCreateAccess+A0	call RtlIsSandboxedToken
Do...	p	SepFilterToken+A8C	call RtlIsSandboxedToken
Do...	p	SepFilterToken+AA0	call RtlIsSandboxedToken

OK Cancel Search Help

Line 1 of 10

New in October Kernel Release



DEMO

NTFS Mount Point Mitigation Bypass

Win32k Hardening



The image shows a Google search for "win32k" with several search results. The top result is a page titled "win32k.sys" with a blue background and white text. The text is a mix of German and English, discussing system errors and hardware compatibility. Below this, there are several other search results, some showing error messages and technical information. One result shows a table of kernel components:

Architecture	File Name	File Path	File Size
x86	ntoskrnl	C:\Windows\System32\ntoskrnl.exe	1,234,567
x86	hal	C:\Windows\System32\hal.dll	123,456
x86	kernel	C:\Windows\System32\kernel.dll	789,012
x86	usermode	C:\Windows\System32\usermode.dll	345,678

Another result shows a list of kernel components in a table format:

Architecture	File Name	File Path	File Size
x86	ntoskrnl	C:\Windows\System32\ntoskrnl.exe	1,234,567
x86	hal	C:\Windows\System32\hal.dll	123,456
x86	kernel	C:\Windows\System32\kernel.dll	789,012
x86	usermode	C:\Windows\System32\usermode.dll	345,678

The search results also include several error messages and technical information, such as "The problem seems to be caused by the following file: win32k.sys" and "The problem seems to be caused by the following file: win32k.sys".

Fonts Are Bad



google-security-research
Google Security Research

Project Home Wiki Issues Source Export to GitHub

New issue Search All issues for font vendor:Microsoft Product:Kernel Search Advanced search Search tips Subscriptions

1 - 19 of 19 List Grid

Owner	Summary + Labels
mjurc...@google.com	Windows Kernel ATMF.DLL DoS via unlimited CharString program execution CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL out-of-bounds reads from the input CharString stream CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL off-by-x oob reads/writes relative to the operand stack CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL kernel pool memory disclosure via uninitialized transient array CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL read/write-what-where in LOAD and STORE operators CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL pool-based buffer overflow in Counter Control Hints CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL pool-based buffer underflow due to integer overflow in STOREWV CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL unlimited out-of-bounds stack manipulation via BLEND operator CCProjectZeroMembers
mjurc...@google.com	Windows Kernel win32k.sys TTF font processing: pool-based buffer overflow in the IUP[] program instruction CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL OTF font processing: pool-based buffer overflow with malformed GPOS table CCProjectZeroMembers
mjurc...@google.com	Windows Kernel win32k.sys TTF font processing: pool-based buffer overflow in win32k!scf_ApplTranslation CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL out-of-bounds reads from the input CharString stream CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL invalid memory access due to malformed CFF table (ATMF.DLL+0x34072 / ATMF.DLL+0x3407b) CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL invalid memory access due to malformed CFF table (ATMF.DLL+0x3440b / ATMF.DLL+0x3440e) CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL write to uninitialized address due to malformed CFF table CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL out-of-bounds read due to malformed Name INDEX in the CFF table CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMF.DLL out-of-bounds read due to malformed FDSelect offset in the CFF table CCProjectZeroMembers
mjurc...@google.com	Windows Kernel win32k.sys TTF font processing: out-of-bounds pool memory access in win32k!fsc_RemoveDups CCProjectZeroMembers
mjurc...@google.com	Windows Kernel win32k.sys TTF font processing: out-of-bounds pool write in win32k!fsc_BLTHoriz CCProjectZeroMembers

Making it Less Bad



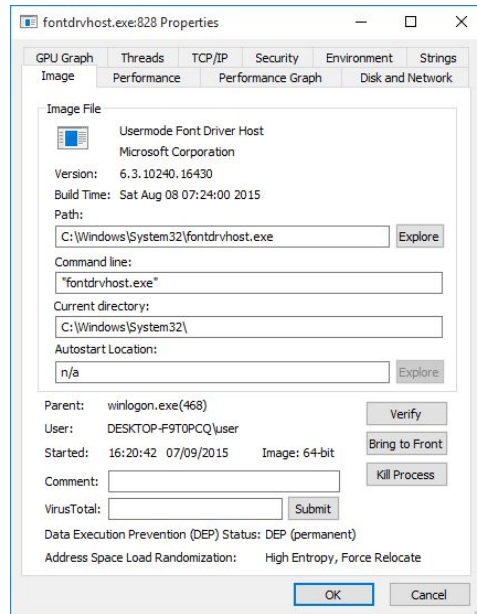
Disable Custom Font Policy (undocumented)

```
PROCESS_MITIGATION_FONT_DISABLE_POLICY  
policy = { 0 };
```

```
policy.DisableNonSystemFonts = 1;  
policy.AuditNonSystemFontLoading = 1;
```

```
SetProcessMitigationPolicy(  
    ProcessFontDisablePolicy,  
&policy,  
    sizeof(policy));
```

User Mode Font Driver



User Mode Font Driver



Running as user in AppContainer

The screenshot shows the 'Token View' window for a user in an AppContainer. The 'Main Details' tab is active, displaying the following information:

- Token User: DESKTOP-F9T0PCQ\user
- User SID: S-1-5-21-2673626951-3238146658-4259417727-1001
- Token Type: Primary
- Impersonation Level: None
- Token ID: 00000000-0024AB57
- Authentication ID: 00000000-00055A0E
- Origin Login ID: 00000000-000003E7
- Modified ID: 00000000-0024AA31
- Integrity Level: Low (with a 'Set Integrity Level' button)
- Session ID: 1
- Elevation Type: Limited (with a 'Linked Token' button)

Below the token details, the 'Source' section shows:

- Name: User32
- Id: 00000000-000559AE

A 'Permissions' button is located at the bottom left of the window.

Only SYSTEM can open process?

The screenshot shows the 'fontdrvhost.exe: 828 Permissions' dialog box. The 'Security' tab is active, and the 'Group or user names' list contains 'SYSTEM'. Below the list are 'Add...' and 'Remove' buttons.

The 'Permissions for SYSTEM' table is as follows:

	Allow	Deny
Full Control	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Read	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Special permissions	<input checked="" type="checkbox"/>	<input type="checkbox"/>

At the bottom, there is a note: 'For special permissions or advanced settings, click Advanced.' with an 'Advanced' button. 'OK' and 'Cancel' buttons are at the very bottom.

Process Token Default DACL



Before September Patch

Token View

Main Details Groups Privileges App Container **Default Dacl** Misc Operations

Default Owner:

Primary Group:

Group	Access	Flags	Type
DESKTOP-F9T0PCQ\user	GenericAll	None	AccessAllowed
NT AUTHORITY\SYSTEM	GenericAll	None	AccessAllowed
S-1-5-0-233041	GenericExecute, GenericRead	None	AccessAllowed
S-1-15-2-95739096-486727260-2...	GenericAll	None	AccessAllowed

After September Patch

Token View

Main Details Groups Privileges App Container **Default Dacl** Misc Operations

Default Owner:

Primary Group:

Group	Access	Flags	Type
OWNER RIGHTS	GenericAll	None	AccessDenied
NT AUTHORITY\SYSTEM	GenericAll	None	AccessAllowed
S-1-15-2-95739096-486727260-2033287795-385358...	GenericAll	None	AccessAllowed

Thread DACLs Allow User Access



The screenshot displays the Process Hacker application interface. The main window shows a list of processes with columns for Name, PID, CPU, and I/O Total. The 'fontdrvhost.exe' process (PID 2452) is selected. A secondary window titled 'fontdrvhost.exe (2452) Properties' is open, showing the 'Threads' tab with a list of threads. Thread 2024 is selected. A third window, 'Permissions for Thread 2024', is open, showing the 'Security' tab. This window lists the group or user names: 'Account: Unknown(S-1-15-2-95739096-486727260-203328...)', 'SYSTEM', 'user (DESKTOP-F9T0PCQ\user)', and 'LogonSessionId_0_233041 (NT AUTHORITY)'. Below this list is a table of permissions for the account 'Unknown(S-1-15-2-95739096-486727260-203328...)' with columns for 'Allow' and 'Deny'. The permissions listed are: Full control, Query limited information, Query information, Set limited information, and Set information. The 'Allow' column has checkboxes checked for all these permissions. The 'Deny' column has checkboxes that are unchecked. At the bottom of the permissions table is an 'Advanced' button. The 'Permissions for Thread 2024' window also has 'OK', 'Cancel', and 'Apply' buttons at the bottom.

TID	CPU	Cycles Delta	Start Address
3496			fontdrvhost.exe
2616			fontdrvhost.exe
2584			fontdrvhost.exe
2024			fontdrvhost.exe
632			fontdrvhost.exe

Permissions for Account	Allow	Deny
Full control	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Query limited information	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Query information	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Set limited information	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Set information	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Extra, UMFD Only Win32k Escape Calls



<i>NtGdiEscape Command</i>	<i>UMFD Escape Call</i>
13	UmfdEscEngGetFileChangeTime
14	UmfdEscEngGetFilePath
15	UmfdEscEngComputeGlyphSet
16	UmfdEscEngCreateFile
17	UmfdEscParseFontResources
18	atmfdFontManagement (enable kernel ATMFD driver)
And Others	...

UmfedEscEngCreateFile in Win32kFull.sys



Before September Patch

```
// Name is \SystemRoot\System32\QLCLF.ATM,  
// ATMLIB.DLL or FAC.ATM  
HANDLE EngCreateFile(UNICODE_STRING Name,  
                    BOOL ReadOnly) {  
    ACCESS_MASK Access = FILE_GENERIC_READ;  
    OBJECT_ATTRIBUTES Attrs;  
    HANDLE Handle = -1;  
    ULONG Disposition = FILE_OPEN;  
  
    if (!ReadOnly) {  
        Access |= FILE_WRITE_DATA;  
        Disposition = FILE_OPEN_IF;  
    }  
  
    InitializeObjectAttributes(&Attrs, &Name,  
                             OBJ_CASE_INSENSITIVE,  
                             ...);  
    ZwCreateFile(&FileHandle, Access, &Attrs, ...,  
                Disposition, ...);  
  
    return FileHandle;  
}
```

After September Patch

```
// Name is only \SystemRoot\System32\FAC.ATM  
HANDLE EngCreateFile(UNICODE_STRING Name) {  
    ACCESS_MASK Access = FILE_GENERIC_READ;  
    OBJECT_ATTRIBUTES Attrs;  
    HANDLE Handle = -1;  
    ULONG Disposition = FILE_OPEN;  
  
    InitializeObjectAttributes(&Attrs, &Name,  
                             OBJ_CASE_INSENSITIVE, ...);  
    IoCreateFile(&FileHandle, FILE_GENERIC_READ,  
                &Attrs, ..., FILE_OPEN, ...,  
                IO_FORCE_ACCESS_CHECK);  
  
    return FileHandle;  
}
```

UmfedEscEngCreateFile in Win32kFull.sys



Before September Patch

```
// Name is \SystemRoot\System32\QLCLF.ATM,  
// ATMLIB.DLL or FAC.ATM  
HANDLE EngCreateFile(UNICODE_STRING Name,  
                    BOOL ReadOnly) {  
    ACCESS_MASK Access = FILE_GENERIC_READ;  
    OBJECT_ATTRIBUTES Attrs;  
    HANDLE Handle = -1;  
    ULONG Disposition = FILE_OPEN;  
  
    if (!ReadOnly) { ← Attacker Controlled  
        Access |= FILE_WRITE_DATA;  
        Disposition = FILE_OPEN_IF;  
    }  
  
    InitializeObjectAttributes(&Attrs, &Name,  
        No Security Check → OBJ_CASE_INSENSITIVE,  
        ...);  
    ZwCreateFile(&FileHandle, Access, &Attrs, ...,  
                Disposition, ...);  
  
    return FileHandle;  
}
```

After September Patch

```
// Name is only \SystemRoot\System32\FAC.ATM  
HANDLE EngCreateFile(UNICODE_STRING Name) {  
    ACCESS_MASK Access = FILE_GENERIC_READ;  
    OBJECT_ATTRIBUTES Attrs;  
    HANDLE Handle = -1;  
    ULONG Disposition = FILE_OPEN;  
  
    InitializeObjectAttributes(&Attrs, &Name,  
        OBJ_CASE_INSENSITIVE, ...);  
    IoCreateFile(&FileHandle, FILE_GENERIC_READ,  
                &Attrs, ..., FILE_OPEN, ...,  
                IO_FORCE_ACCESS_CHECK);  
  
    return FileHandle;  
}
```

All Gone ←

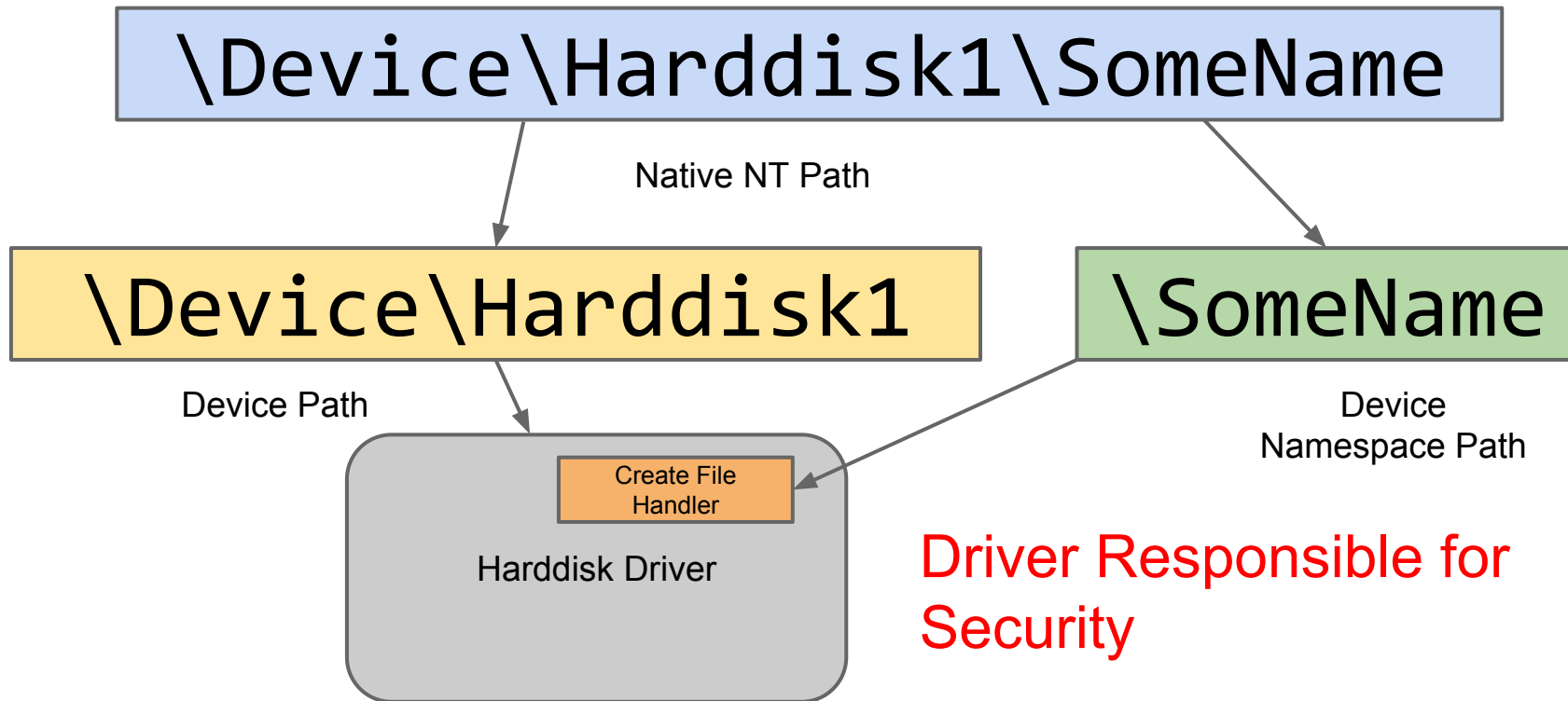
Process Silos



- New process container mechanism
- Possibly related to docker support
- Works in a similar fashion to process jobs

```
NTSTATUS NtCreateSiloObject(  
    PHANDLE handle,  
    ACCESS_MASK DesiredAccess,  
    POBJECT_ATTRIBUTES ObjectAttributes);
```

Opening Device Object



Replace the Root Object Directory



```
// Create anonymous directory object
InitializeObjectAttributes(&ObjectAttributes,
                          NULL, 0, NULL, NULL);
NtCreateDirectoryObject(&hDir, &ObjectAttributes, ...);

NtSetInformationSiloObject(hSilo, SiloObjectRootDirectory,
                          &hDir, sizeof(hDir));
NtAssignProcessToSiloObject(hSilo, GetCurrentProcess());
// Process root directory now empty
```

Exploit: <https://code.google.com/p/google-security-research/issues/detail?id=459>

Fixed in RTM



- Silo functionality rolled into Job objects
- Changed object directory now behind a TCB check
- Shame for Chrome, would have been a useful feature

Public Service Announcement



Doing too much security
research on Beta software
can make you sad

(Dis)Honourable Mentions

- 😊 Control Flow Guard (CFG)
- 🤔 Privacy Options
- 😞 Cumulative Updates
- 😡 Microsoft Cross-Signed Drivers

Conclusions




2 steps forward, 1 step back. Still plenty of things to attack!



DEMO

Local System Elevation

Good Old Issue 222



google-security-research

Google Security Research

[Project Home](#) [Wiki](#) [Issues](#) [Source](#) [Export to GitHub](#)

[New issue](#) Search for [Search](#) [Advanced search](#) [Search tips](#) [Subscriptions](#)

Issue [222](#): Windows: Local WebDAV NTLM Reflection Elevation of Privilege

10 people starred this issue and may be notified of changes. [Back to list](#)

<p>Status: WontFix</p> <p>Owner: fors...@google.com</p> <p>Closed: Mar 18</p> <p>Cc: project...@google.com</p> <p>Vendor: Microsoft</p> <p>Product: Windows</p> <p>Severity: High</p> <p>Finder: forshaw</p> <p>Reported: 2014-Dec-18</p> <p>CCProjectZeroMembers</p> <p>Deadline: 90</p> <p>MSRC: 21243</p> <p>Sign in to add a comment</p>	<p>Project Member Reported by fors...@google.com, Dec 18, 2014</p> <p>Windows: Local WebDAV NTLM Reflection Elevation of Privilege Platform: Windows 8.1 Update, Windows 7 Class: Elevation of Privilege</p> <p>Summary: A default installation of Windows 7/8 can be made to perform a NTLM reflection attack through WebDAV which allows a local user to elevate privileges to local system.</p> <p>Description:</p> <p>NTLM reflection is a well known issue with Windows authentication. It's typically abused in networked scenarios to reflect credentials from one machine to another. It used to be possible to reflect credentials back to the same machine but that was mitigated in MS08-068 by not honouring NTLM authentication sessions already in flight. However this did nothing to stop cross-protocol attacks.</p> <p>The WebClient service for WebDAV (which is installed and enabled by default, although you'd need to start it using its service trigger) also does NTLM authentication if the server requests it. As Windows has no block on binding to TCP ports < 1024 from a normal user account then we can setup our own WebDAV server running as a normal user bound to localhost (so also no firewall issues). If we can convince another user, ideally local system to connect to the WebDAV server we can start an NTLM authentication session. This can then be replayed locally to the TCP/IP CIFS service endpoint to authenticate as that user. If this was a local system account then that gives you full local admin privs, you can read/write any file on the system through the admin shares. You could also bind to local named pipes such as the service manager and create a new privileged service.</p>
--	---

<https://code.google.com/p/google-security-research/issues/detail?id=222>

Questions?

