Google

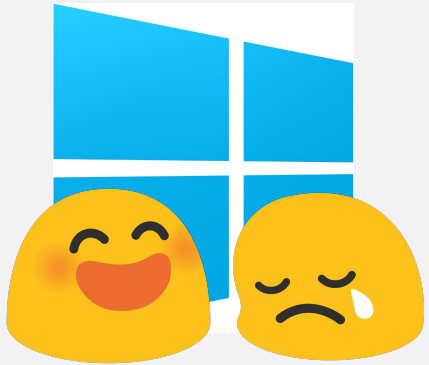
Windows 10

2 Steps Forward,

1 Step Back

James Forshaw @tiraniddo

Ruxcon 2015

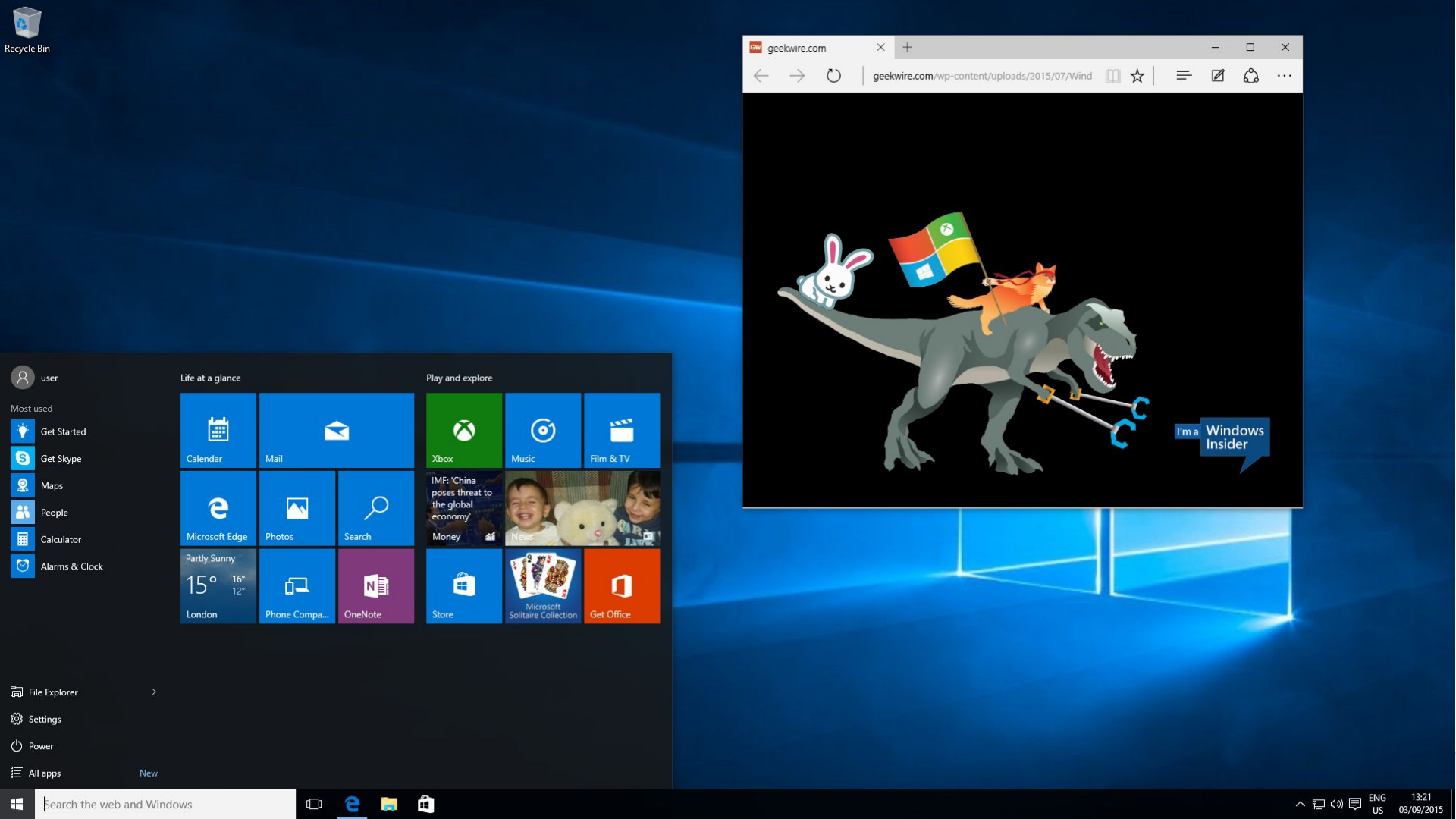


Obligatory Background Slide

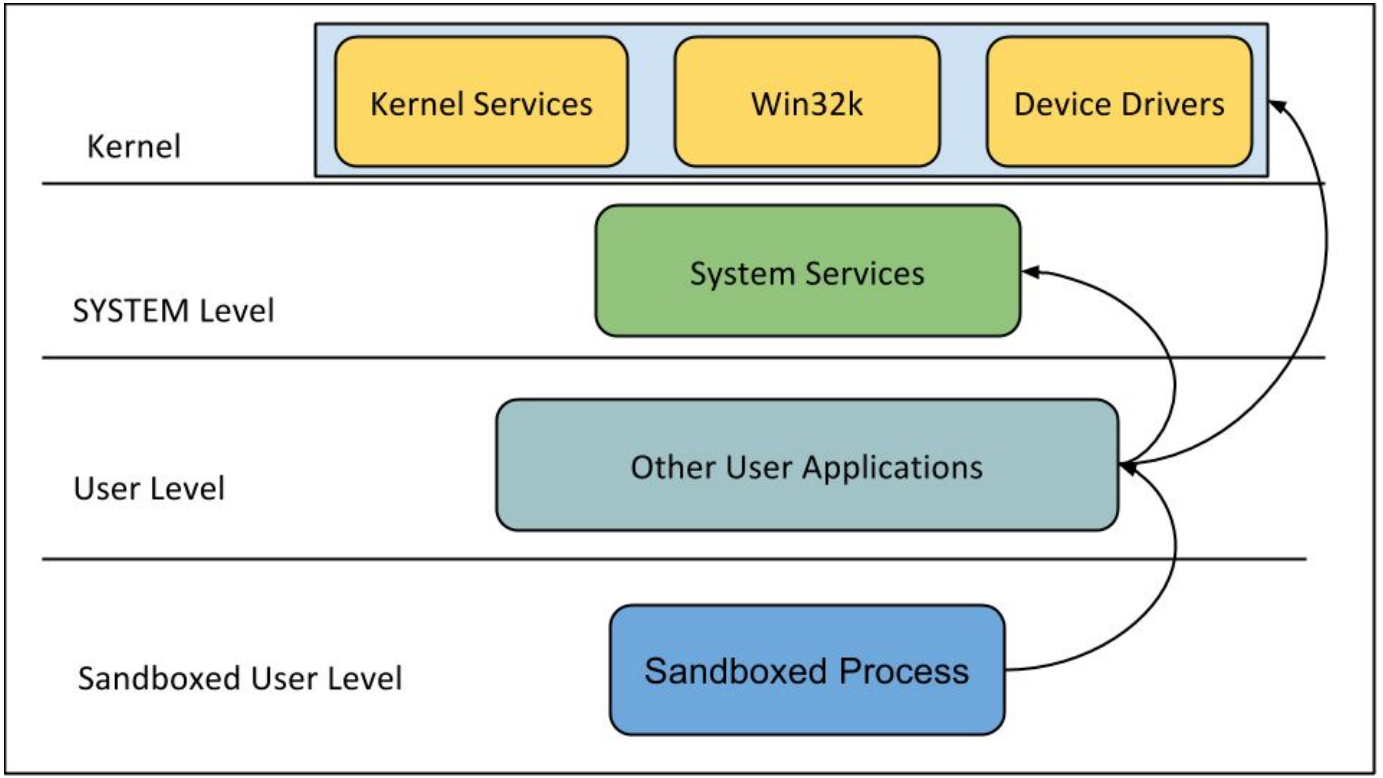
- Researcher in Google's Project Zero team
- Specialize in Windows
  - Especially local privilege escalation
- Never met a logical vulnerability I didn't like

What I’m Going to Talk About

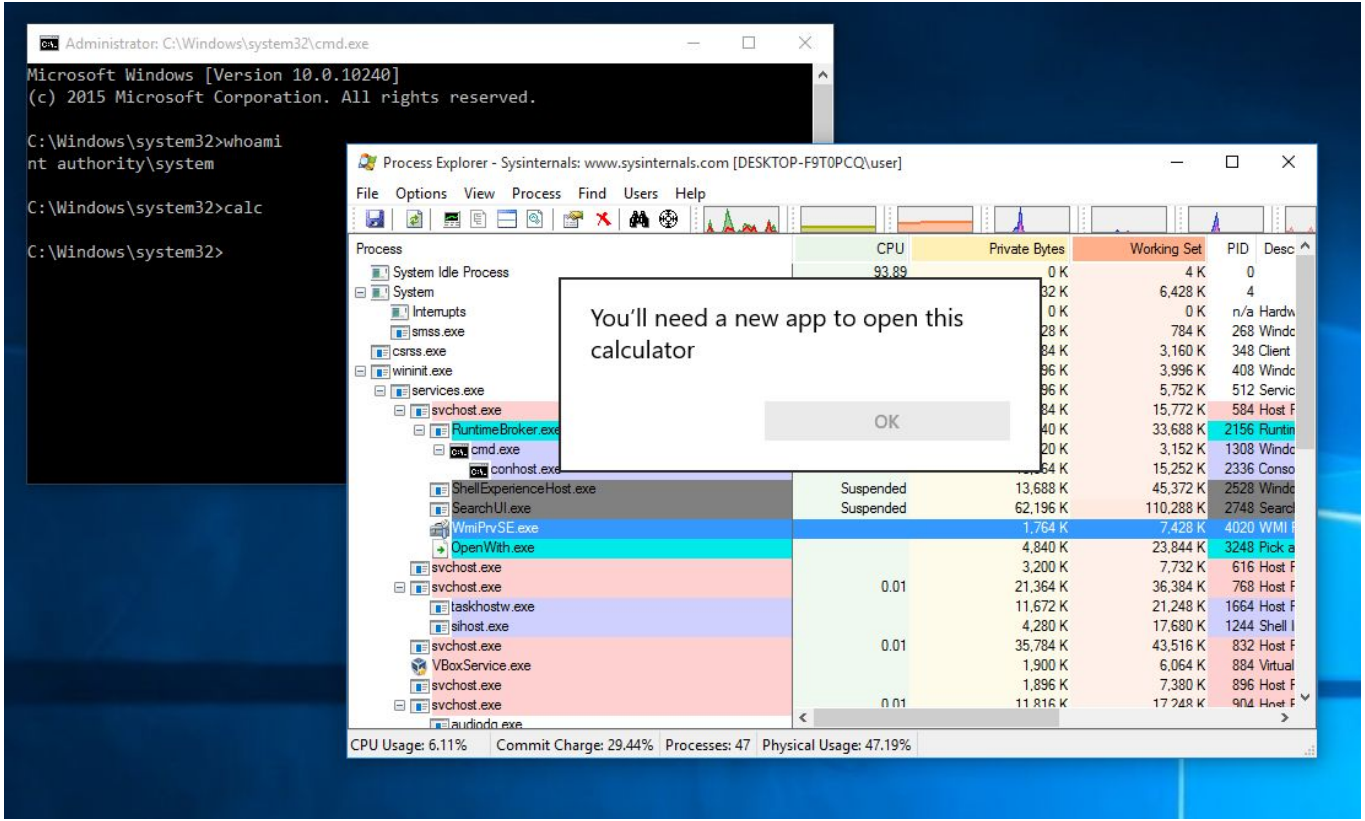
- Some research on Windows 10 from the early preview builds
- Why Windows 10 is awesome for security
- Except for when it isn’t!
- Very much looking at things from a local privilege escalation perspective


Recycle Bin
geekwire.com
geekwire.com/wp-content/uploads/2015/07/Wind
I'm a Windows Insider
user
Most used
Get Started
Get Skype
Maps
People
Calculator
Alarms & Clock
Life at a glance
Play and explore
Calendar
Mail
Xbox
Music
Film & TV
Microsoft Edge
Photos
Search
IMF: 'China poses threat to the global economy'
Money
News
Partly Sunny
15°
16°
12°
London
Phone Compa...
OneNote
Store
Microsoft Solitaire Collection
Get Office
File Explorer
Settings
Power
All apps
New
Search the web and Windows
ENG
US
13:21
03/09/2015

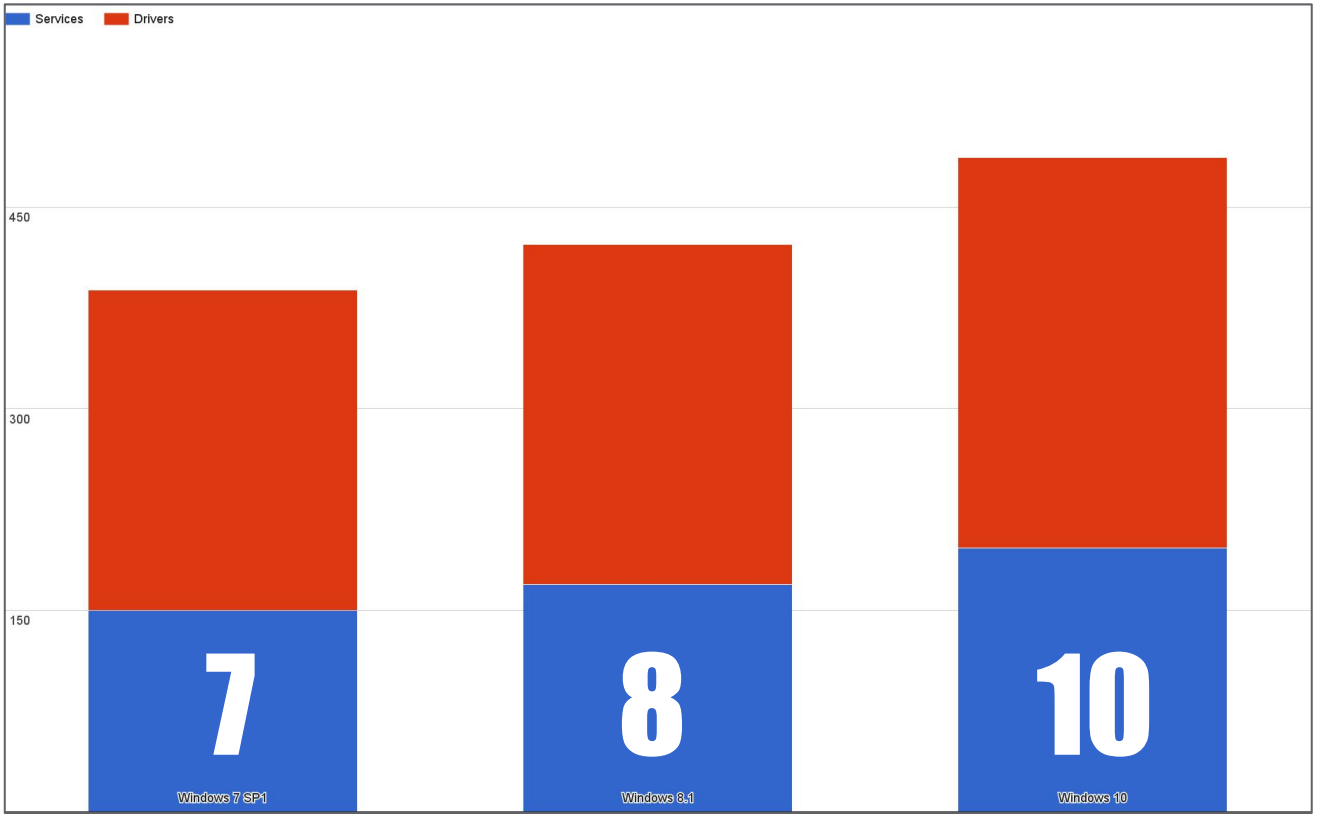
Windows Local Attack Surface



Local System Vulnerabilities are Dead!

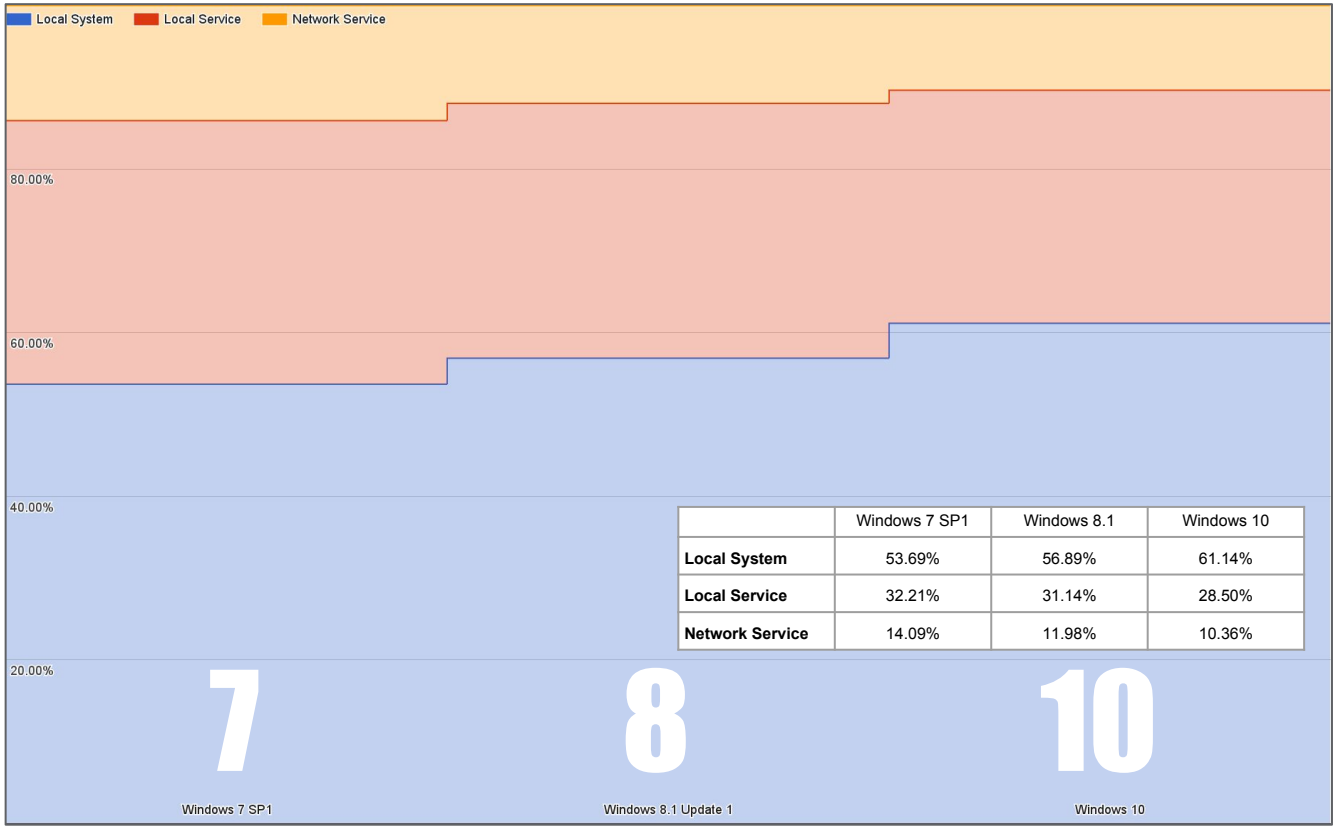


System Services and Drivers



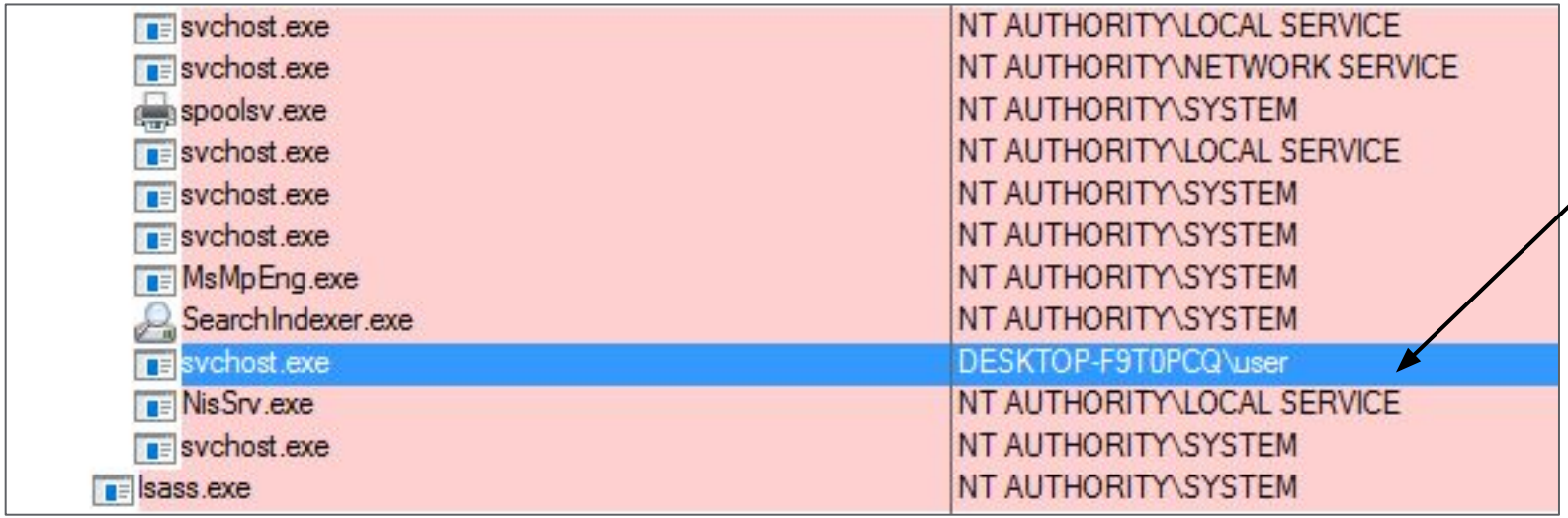
	Windows 7 SP1	Windows 8.1	Windows 10
Services	150	169	196
Drivers	238	253	291

Service Privilege Levels



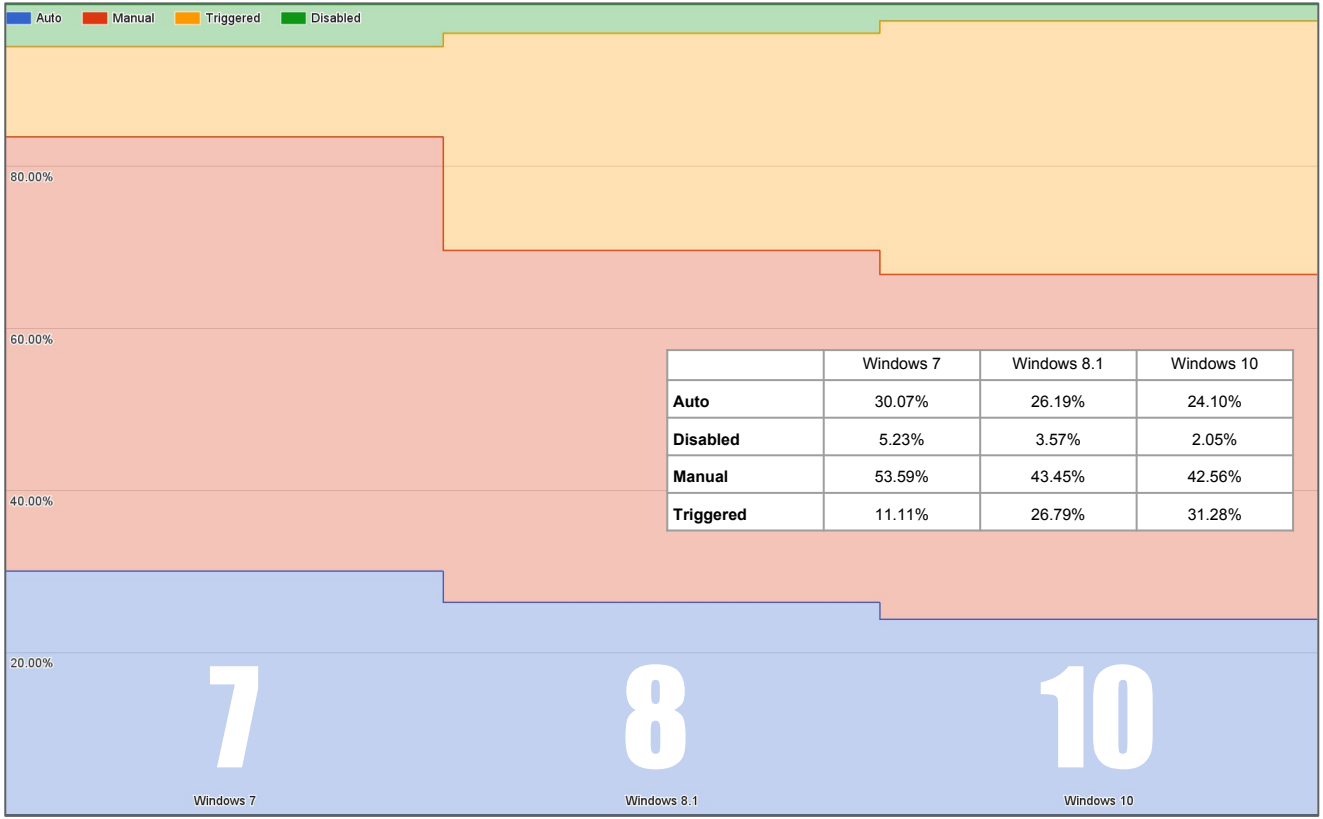
	Windows 7 SP1	Windows 8.1	Windows 10
Local System	53.69%	56.89%	61.14%
Local Service	32.21%	31.14%	28.50%
Network Service	14.09%	11.98%	10.36%

SVCHOST Running as User?



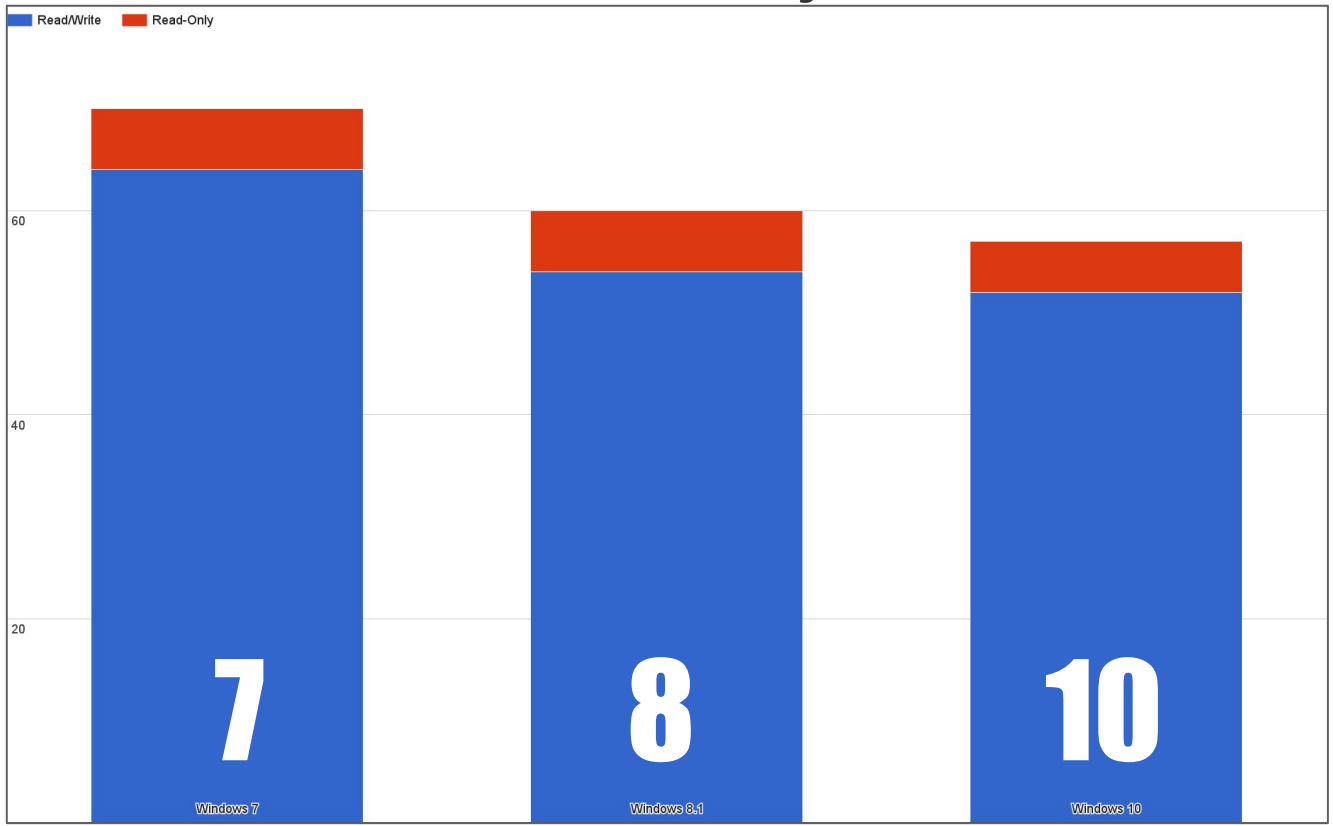
svchost.exe	NT AUTHORITY\LOCAL SERVICE
svchost.exe	NT AUTHORITY\NETWORK SERVICE
spoolsv.exe	NT AUTHORITY\SYSTEM
svchost.exe	NT AUTHORITY\LOCAL SERVICE
svchost.exe	NT AUTHORITY\SYSTEM
svchost.exe	NT AUTHORITY\SYSTEM
MsMpEng.exe	NT AUTHORITY\SYSTEM
SearchIndexer.exe	NT AUTHORITY\SYSTEM
svchost.exe	DESKTOP-F9T0PCQ\user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

Service Start Mode



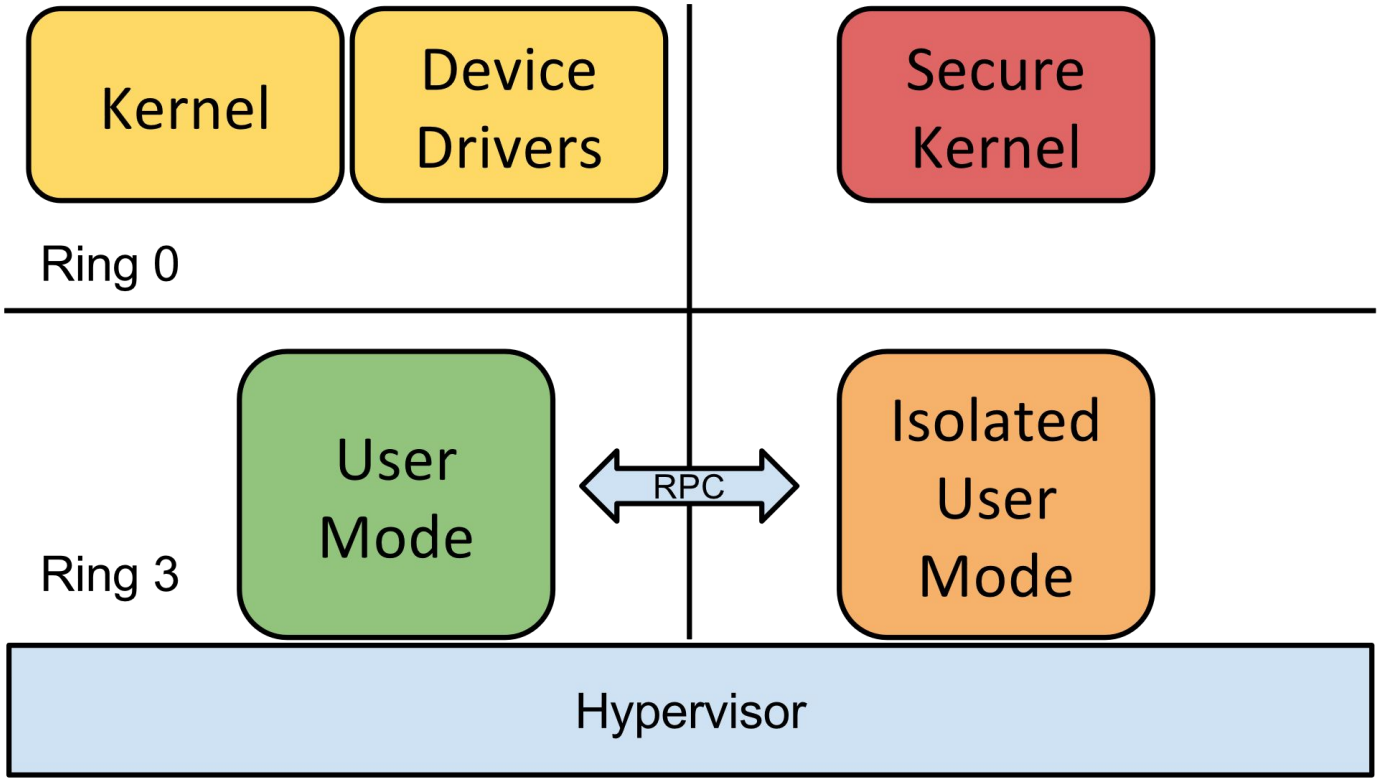
	Windows 7	Windows 8.1	Windows 10
Auto	30.07%	26.19%	24.10%
Disabled	5.23%	3.57%	2.05%
Manual	53.59%	43.45%	42.56%
Triggered	11.11%	26.79%	31.28%

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

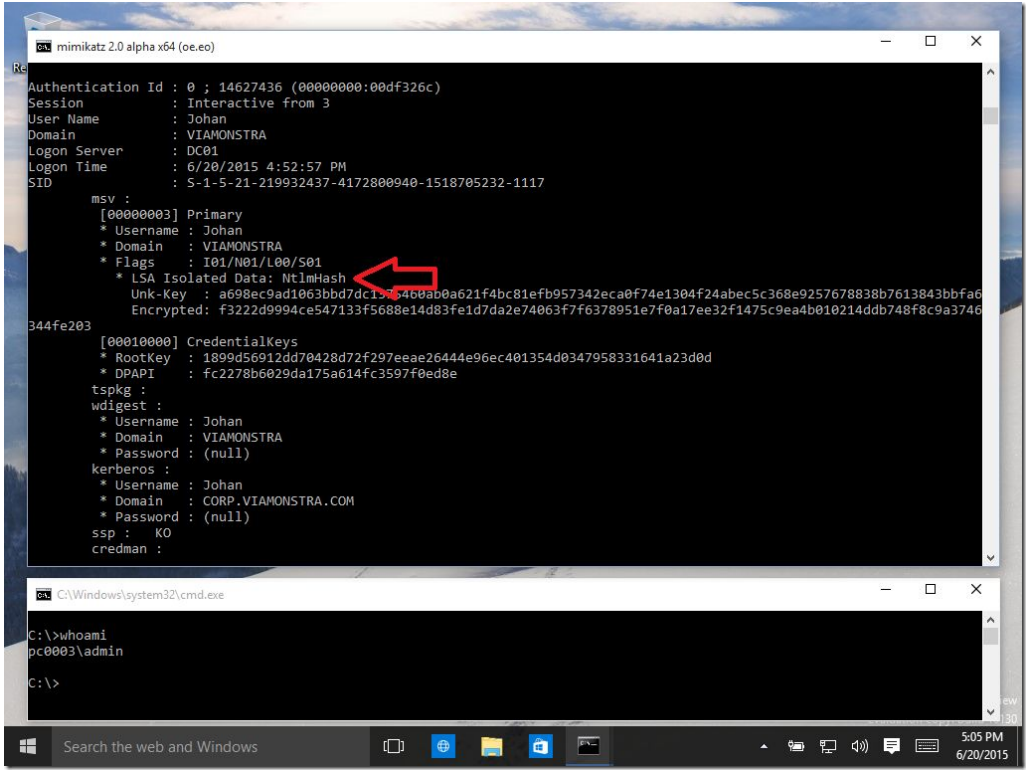


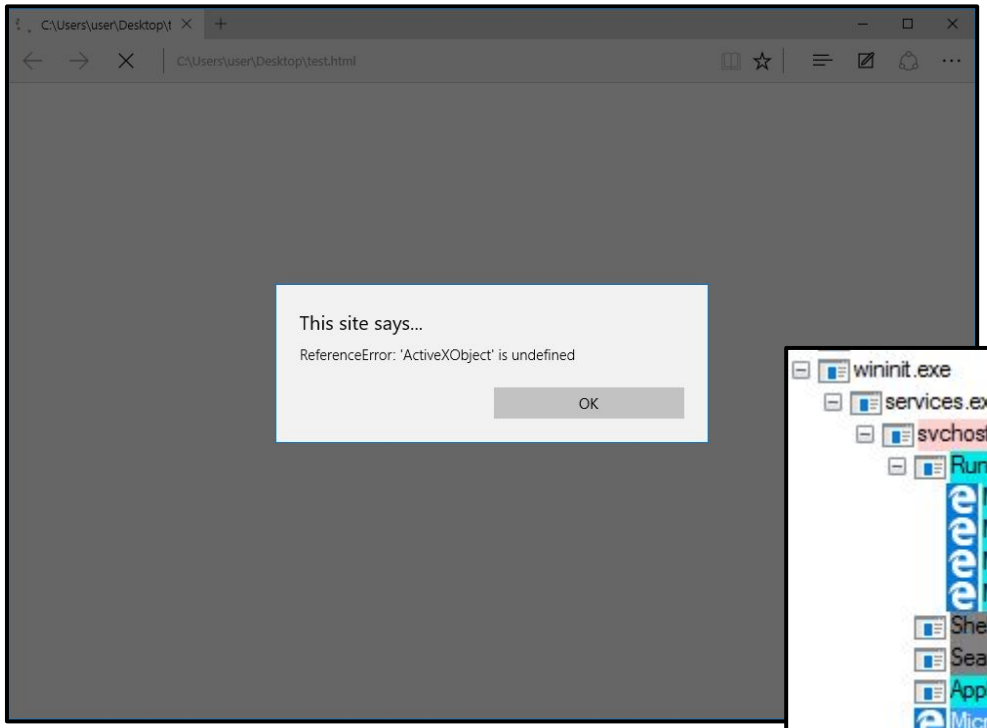
Image from http://deploymentresearch.com/Research/Post/490/Enabling-Virtual-Secure-Mode-VSM-in-Windows-10-Enterprise-Build-10130

But Sadly

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

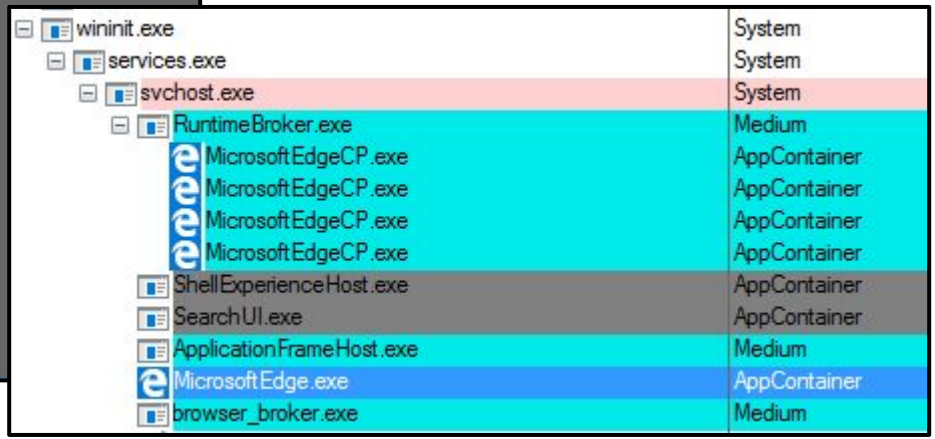
Edge Browser

Microsoft Edge Security

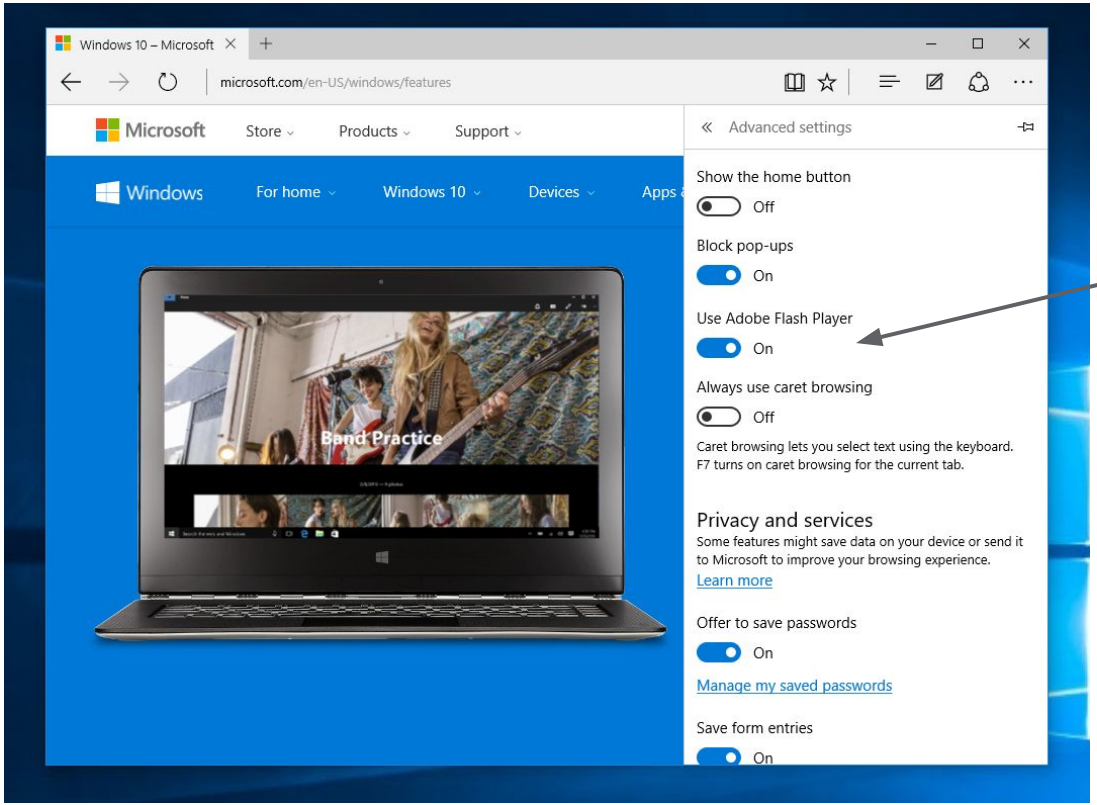


ActiveX is gone(ish)

AppContainer Sandbox Always On

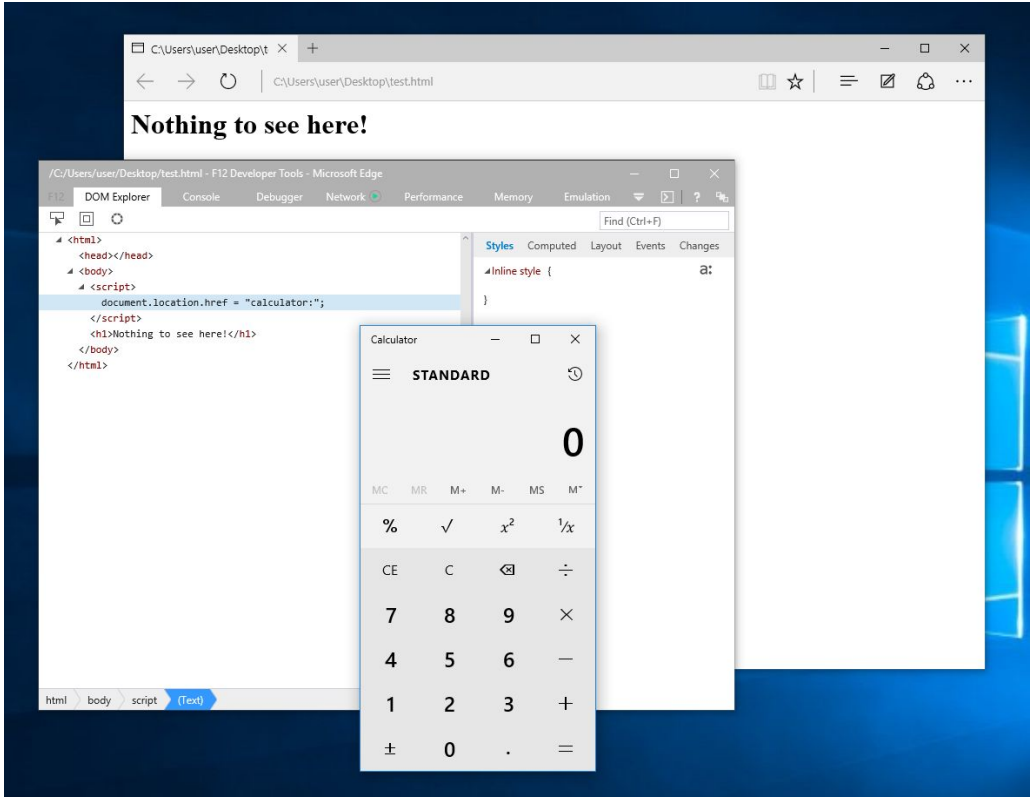


Microsoft Edge and Flash

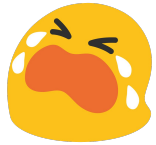


Nope!

Has No One Learnt from the Past?



Guess Trident Wasn’t a Suitable Base?



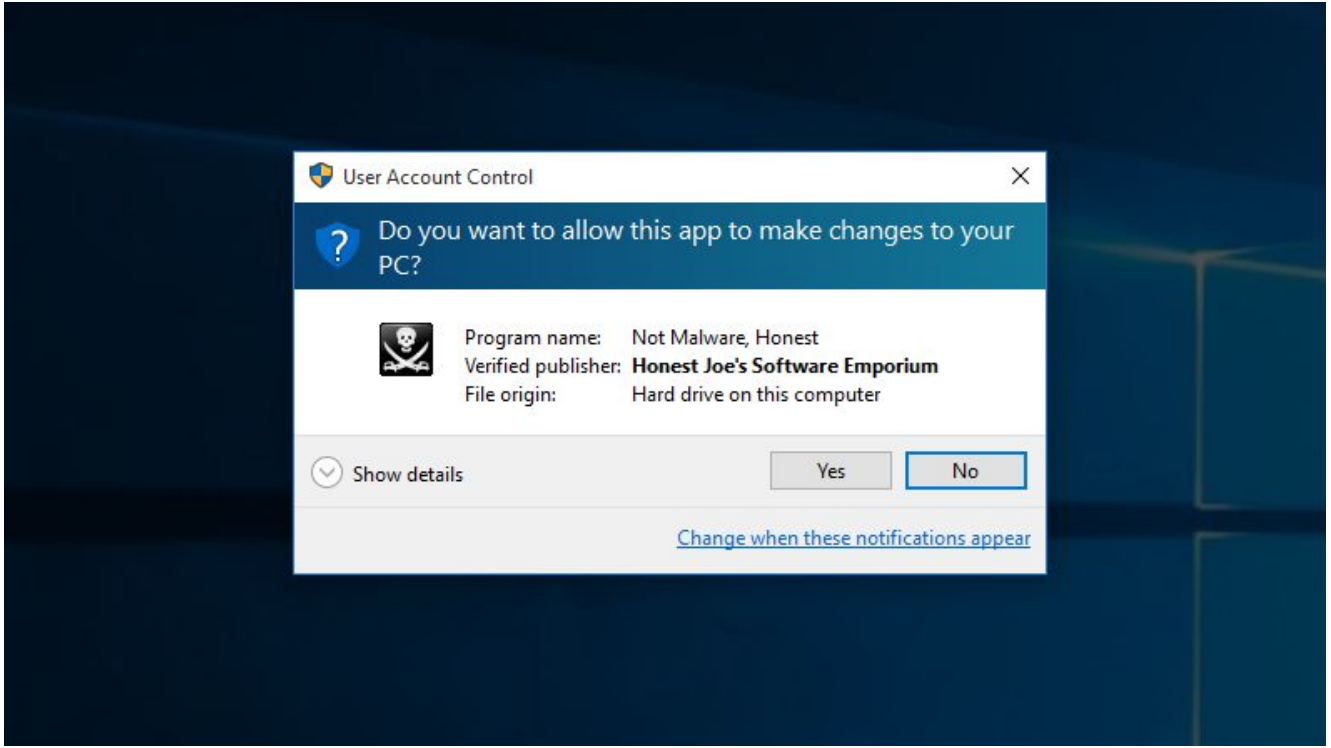
h0wl / edge_crash.html

Last active 9 days ago

edge_crash.html — Raw

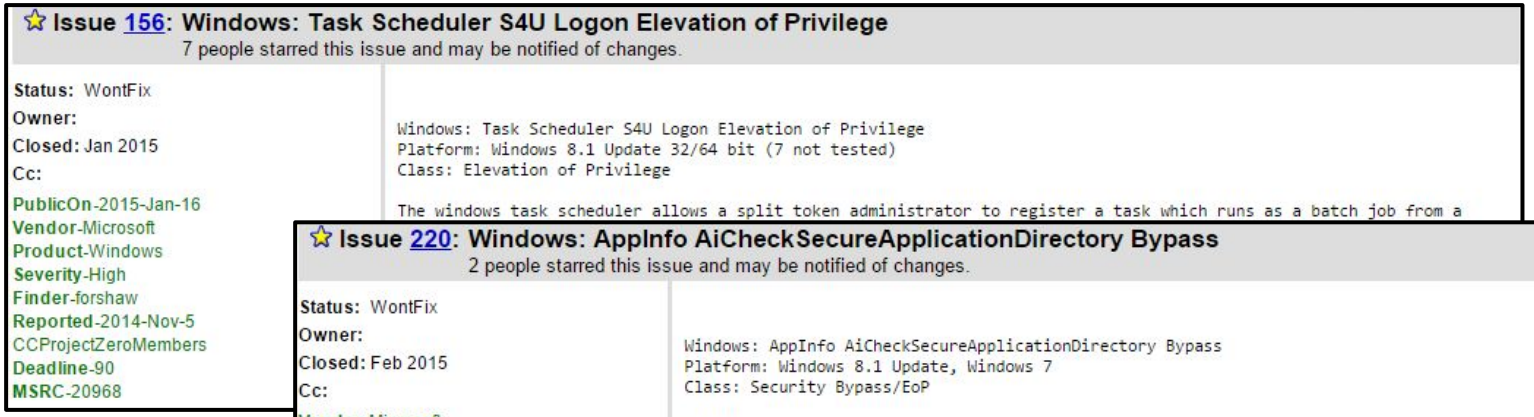
```
<!-- based on https://connect.microsoft.com/IE/feedback/details/1683347/ms-edge-combination-of-iframe-anchor-h
<!doctype html>
<html>
<head>
<script>
function boom() {
        var iframe = document.getElementById('iframe1');
        document.location.href = '#';
        iframe.parentNode.removeChild(iframe);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

User Account Control



They've Fixed Some Bugs I've Reported

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

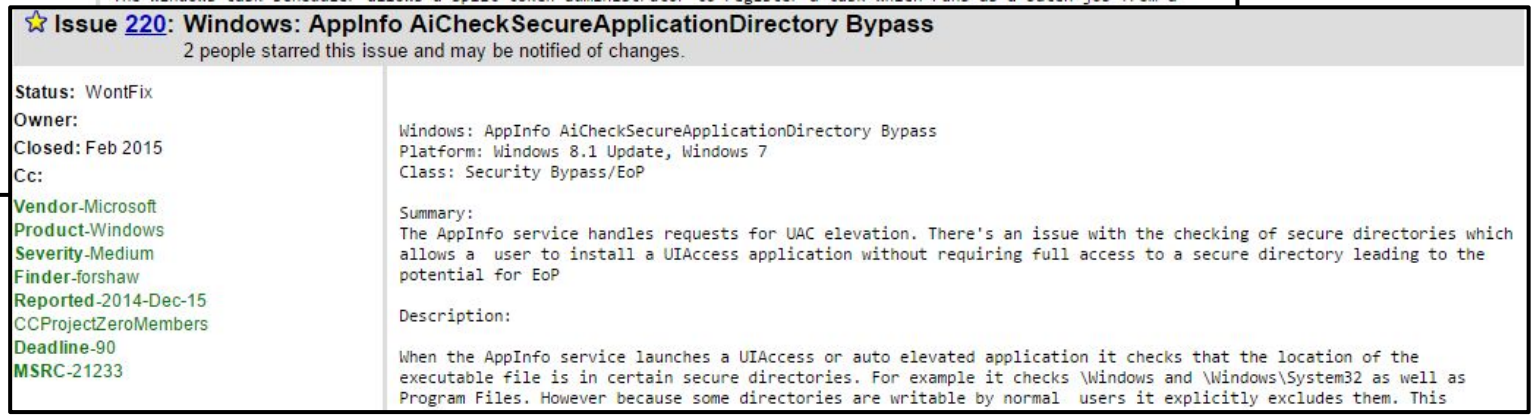


Issue 156: Windows: Task Scheduler S4U Logon Elevation of Privilege

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

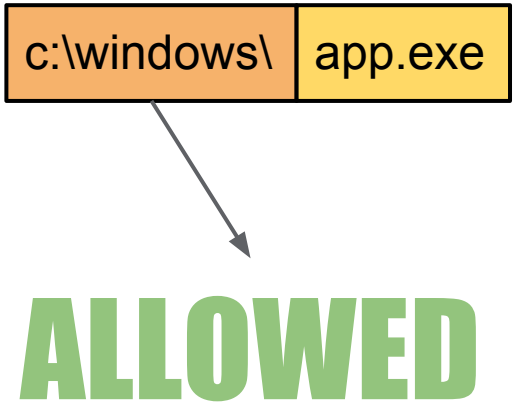
2 people starred this issue and may be notified of changes.

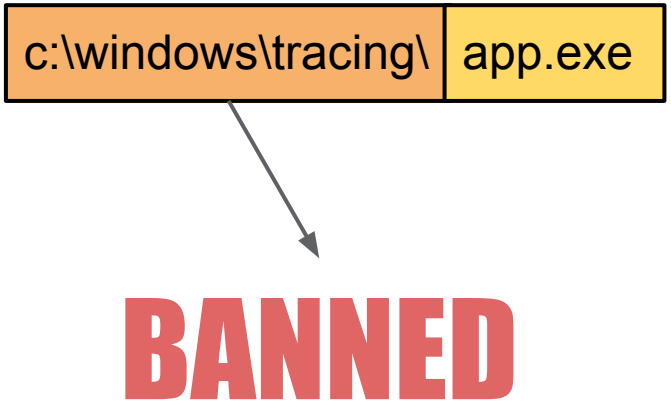
Status: WontFix
Owner:
Closed: Feb 2015
Cc:
Vendor-Microsoft
Product-Windows
Severity-Medium
Finder-forshaw
Reported-2014-Dec-15
CCProjectZeroMembers
Deadline-90
MSRC-21233

```
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
```

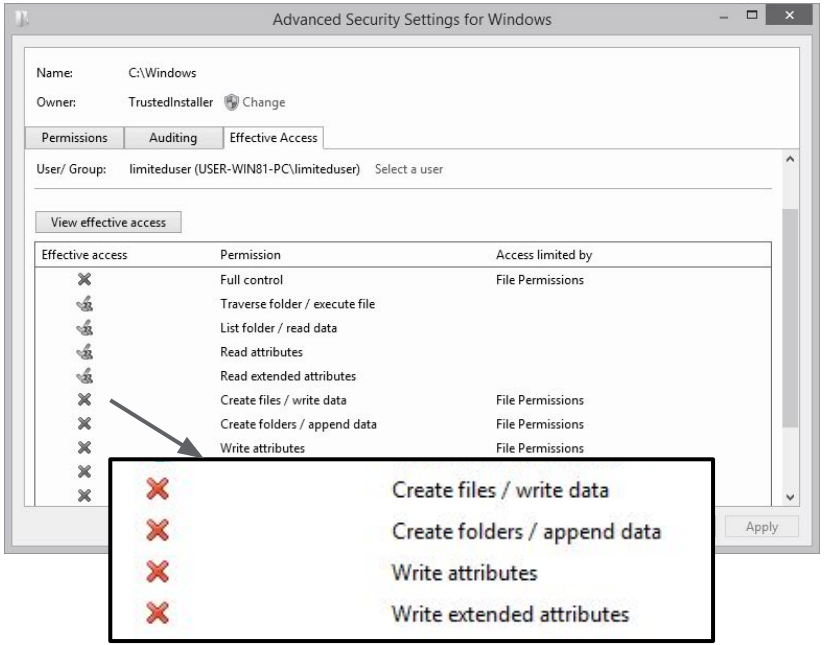
https://code.google.com/p/google-security-research/issues/detail?id=220

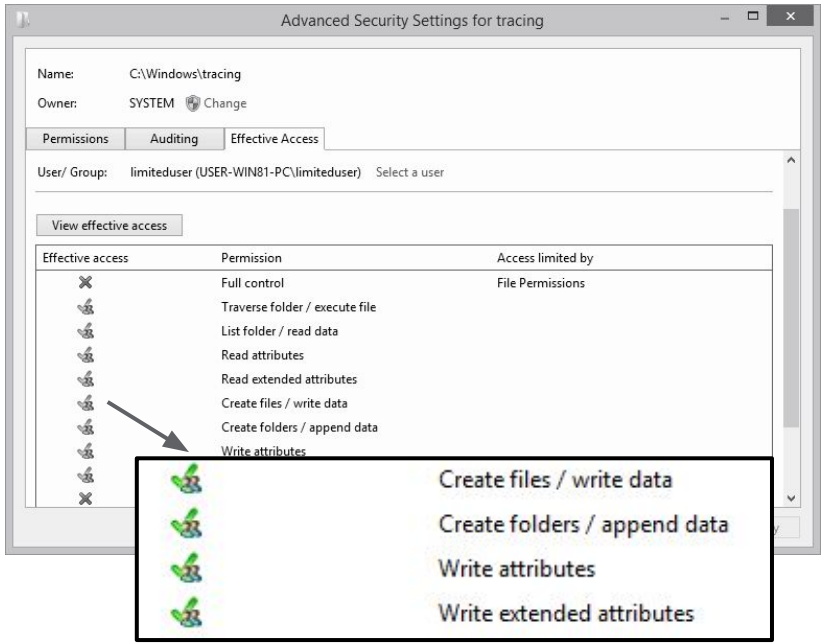
UAC Auto Elevation Directory Check





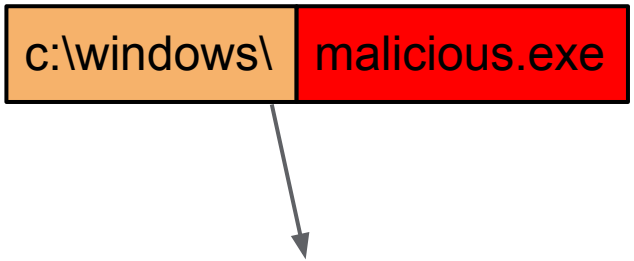
Folder Permissions



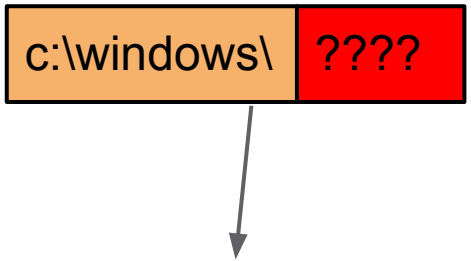


AiCheckSecureApplicationDirectory Bypass

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

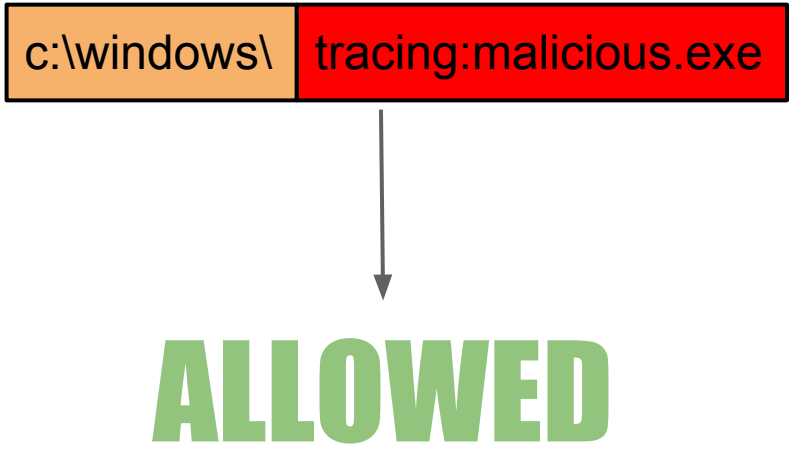


ALLOWED



ALLOWED?

NTFS Alternate Data Streams FTW!



- 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.

Status: Fixed
Owner:
Closed: Mar 10
Cc:
Vendor-Microsoft
Product-Windows-Kernel
Severity-medium
Finder-forshaw
Reported-2014-Dec-15
CCProjectZeroMembers
Deadline-90
MSRC-21232
CVE-2015-0078
MS15-023

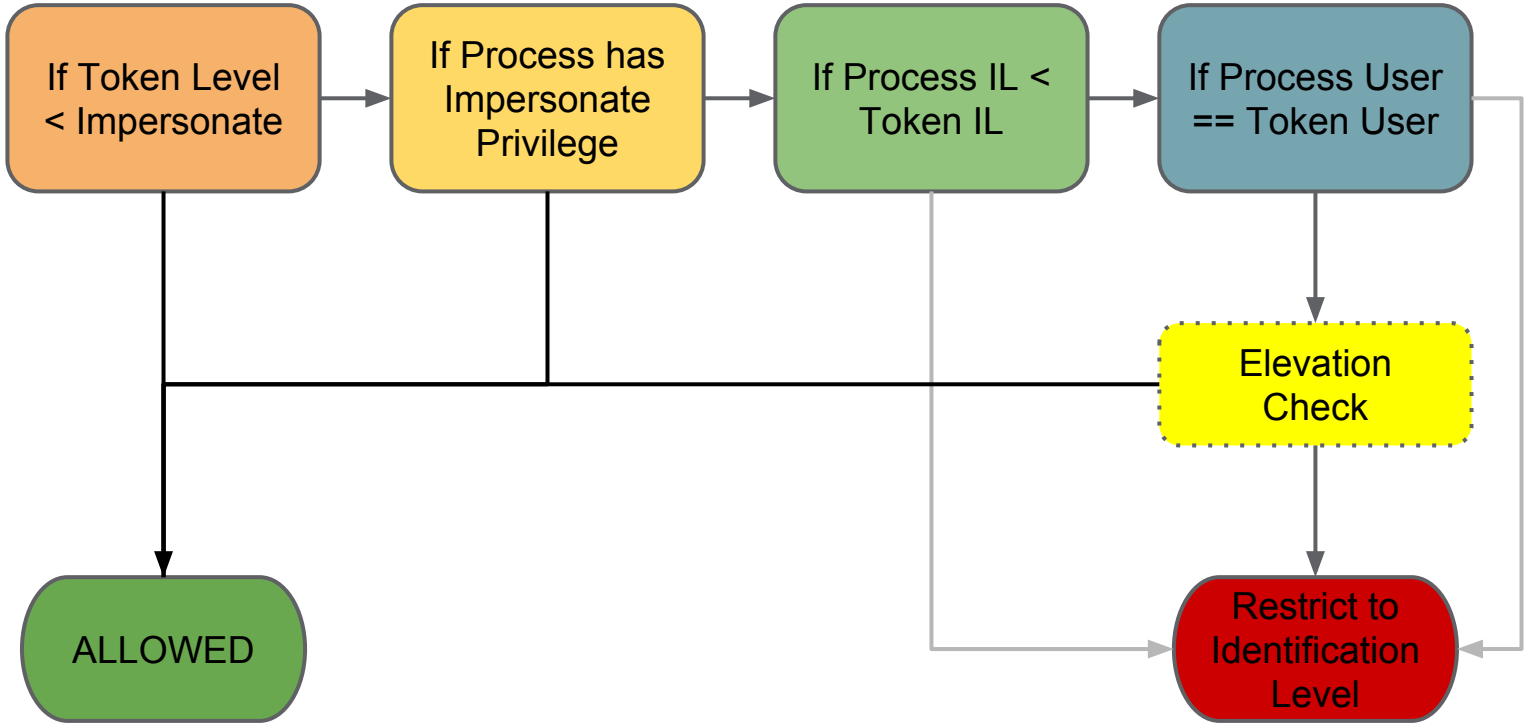
```
Windows: NtUserGetClipboardAccessToken Token Leak
Platform: Windows 8.1 Update (Windows 7 not vulnerable)
Class: Security Bypass/EoP

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.

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.

The actual code goes something like:
```

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

DEMO

Elevated Token Capture

Well MS Almost Did

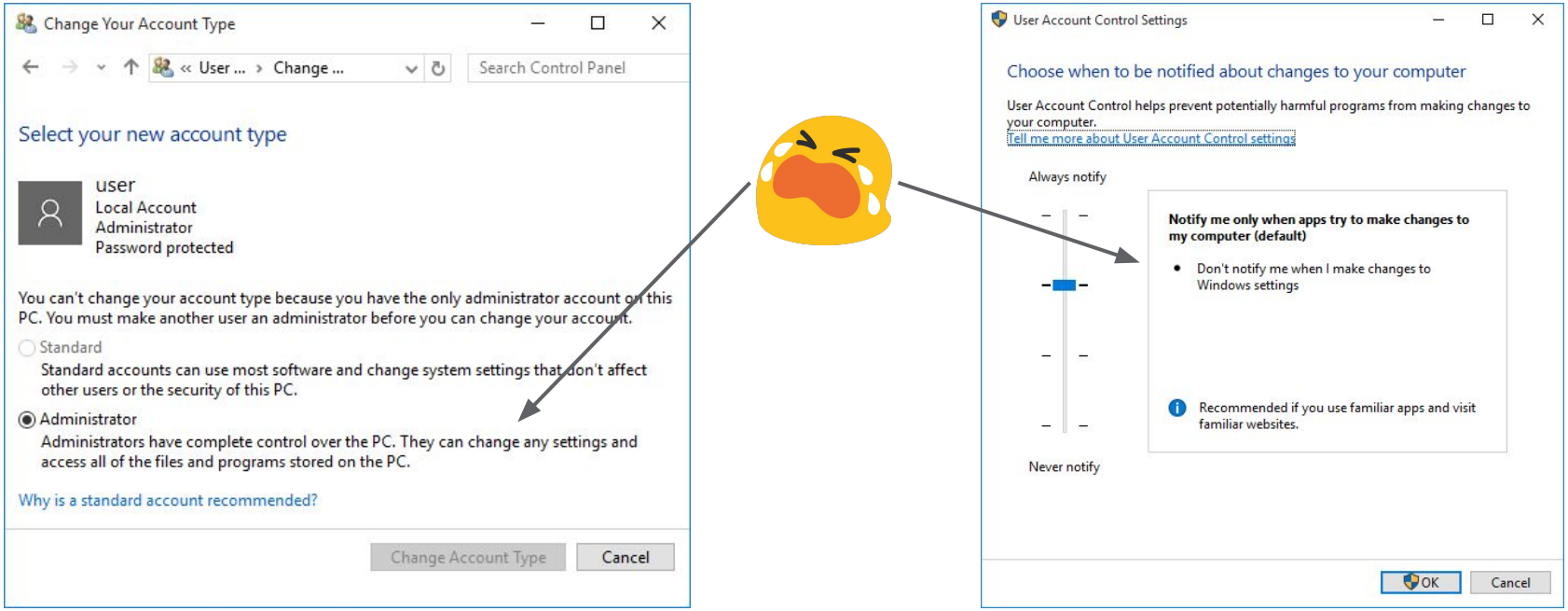


Elevated Token Impersonation

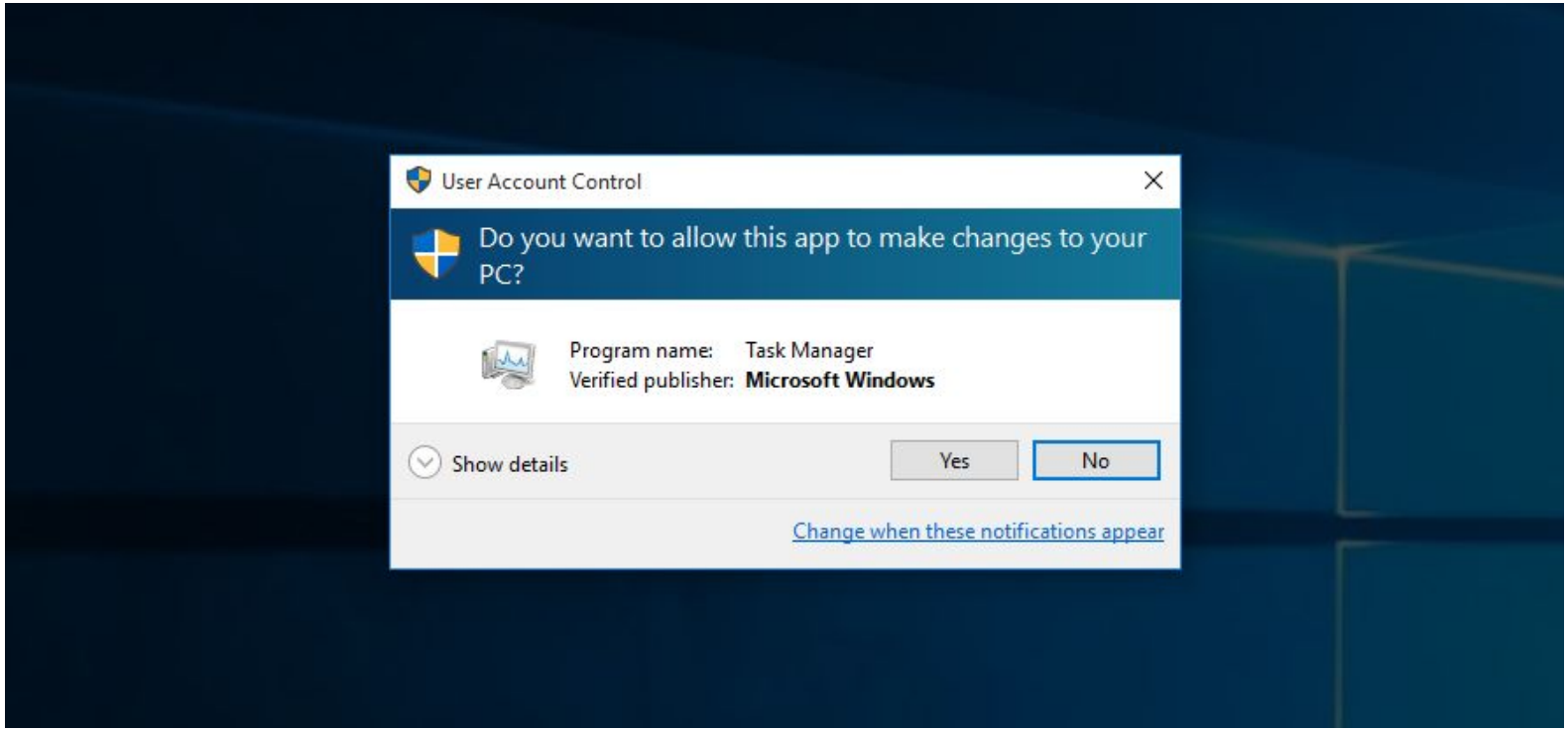
- Blocks impersonating an elevated token unless process token is also elevated
- Must be enabled in SeCompatFlags kernel flag

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

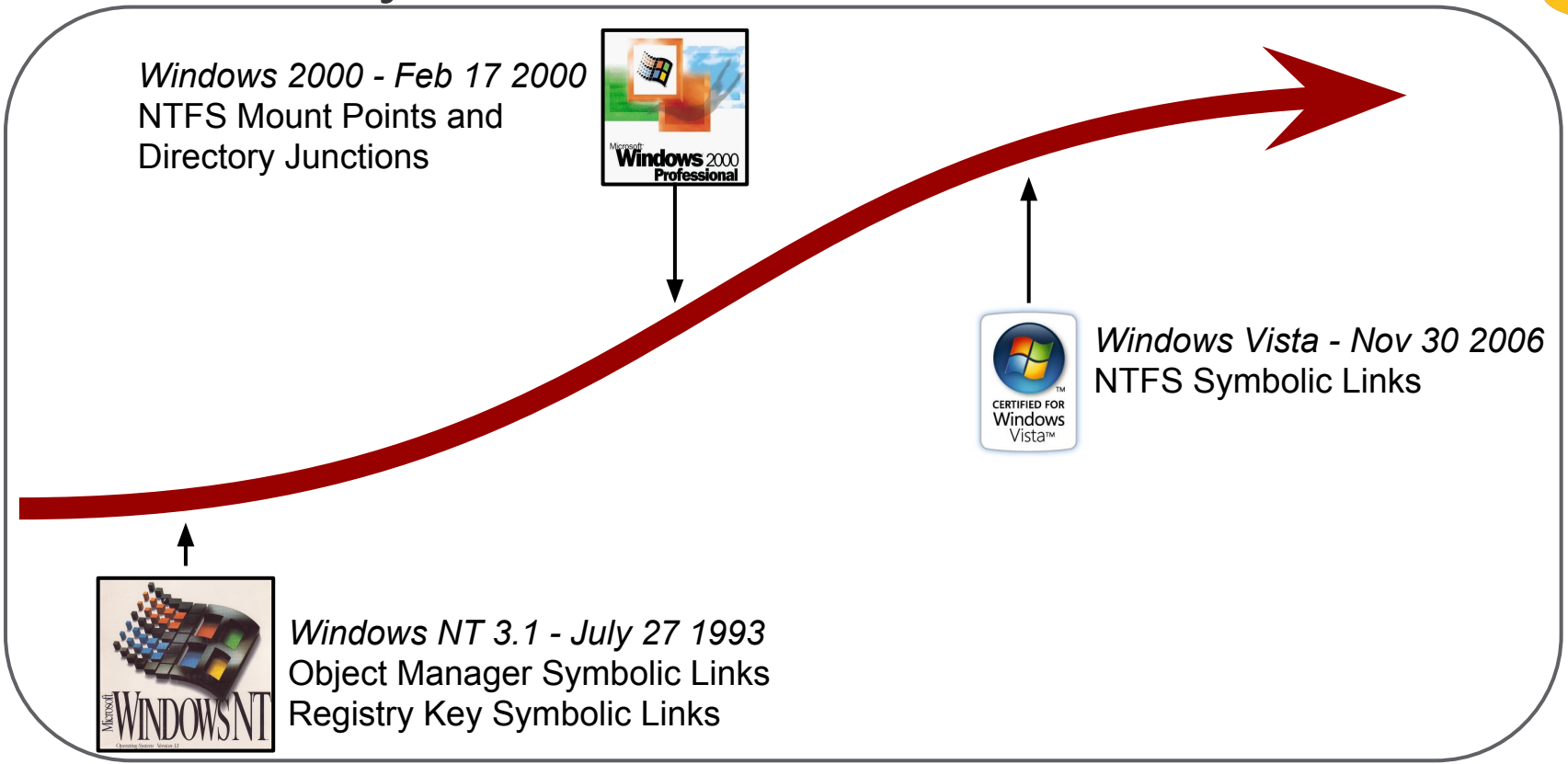
In The End Still the “Wrong” Default IMO!



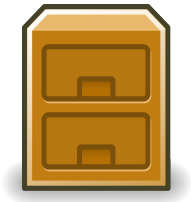
If You Change Task Manager Needs a Prompt



Windows Symbolic Links

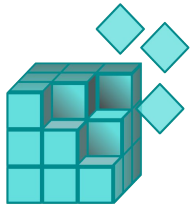


Mitigated in Sandboxes



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);
    }
  }
}
```

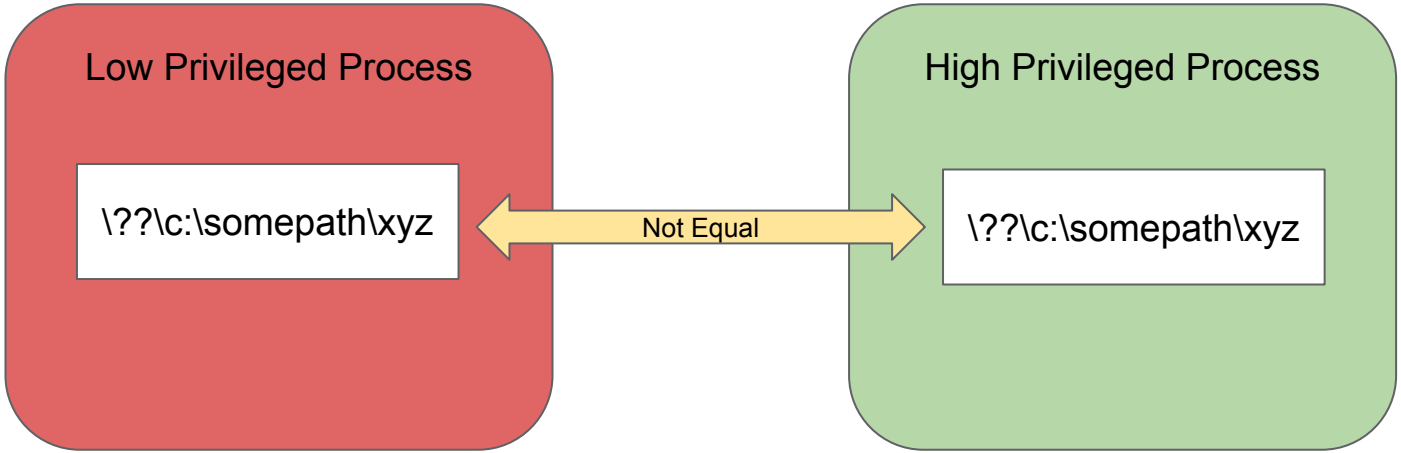
Mount Point Mitigation Bypass

```
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);
    }
  }
}
```

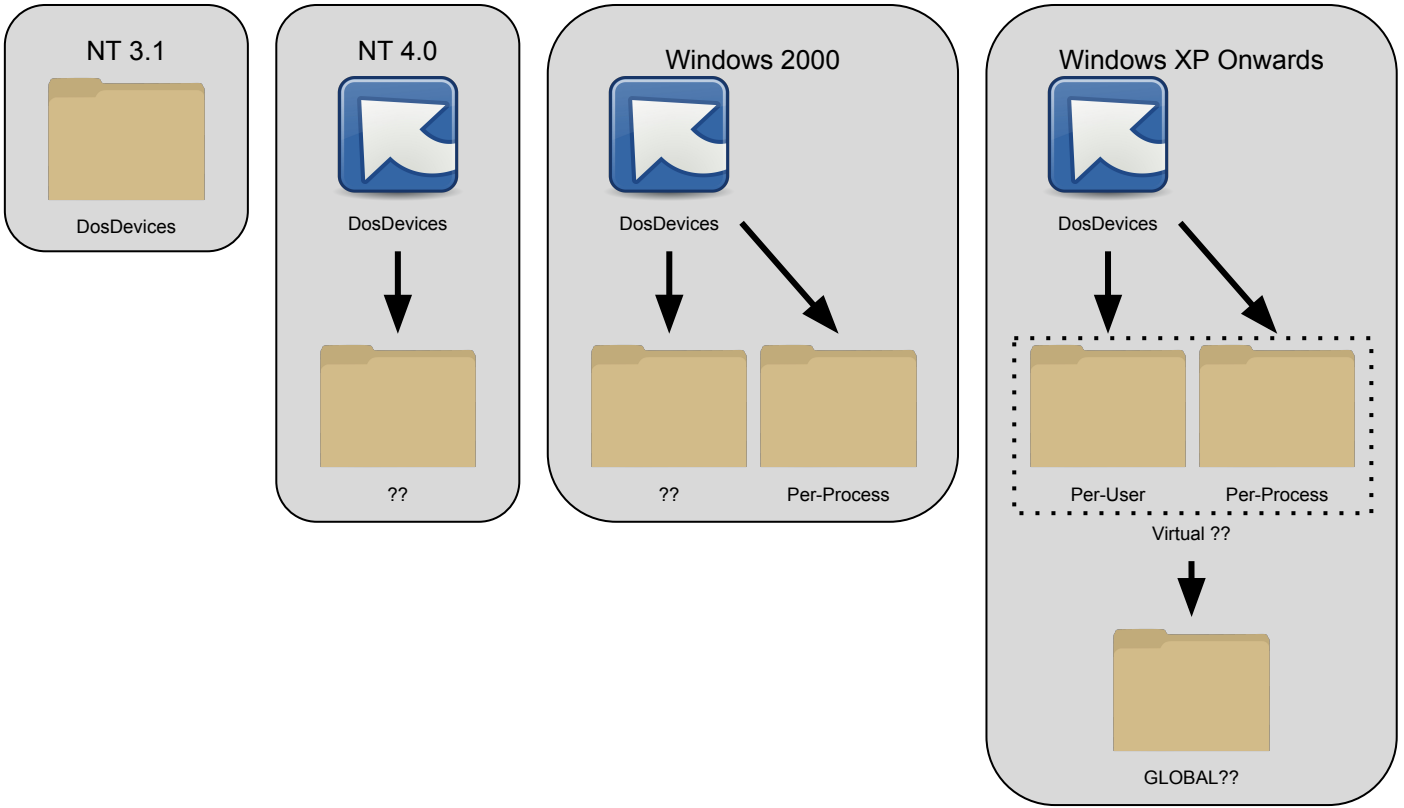
Mount Point Mitigation Bypass

```
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);
    }
  }
}
```

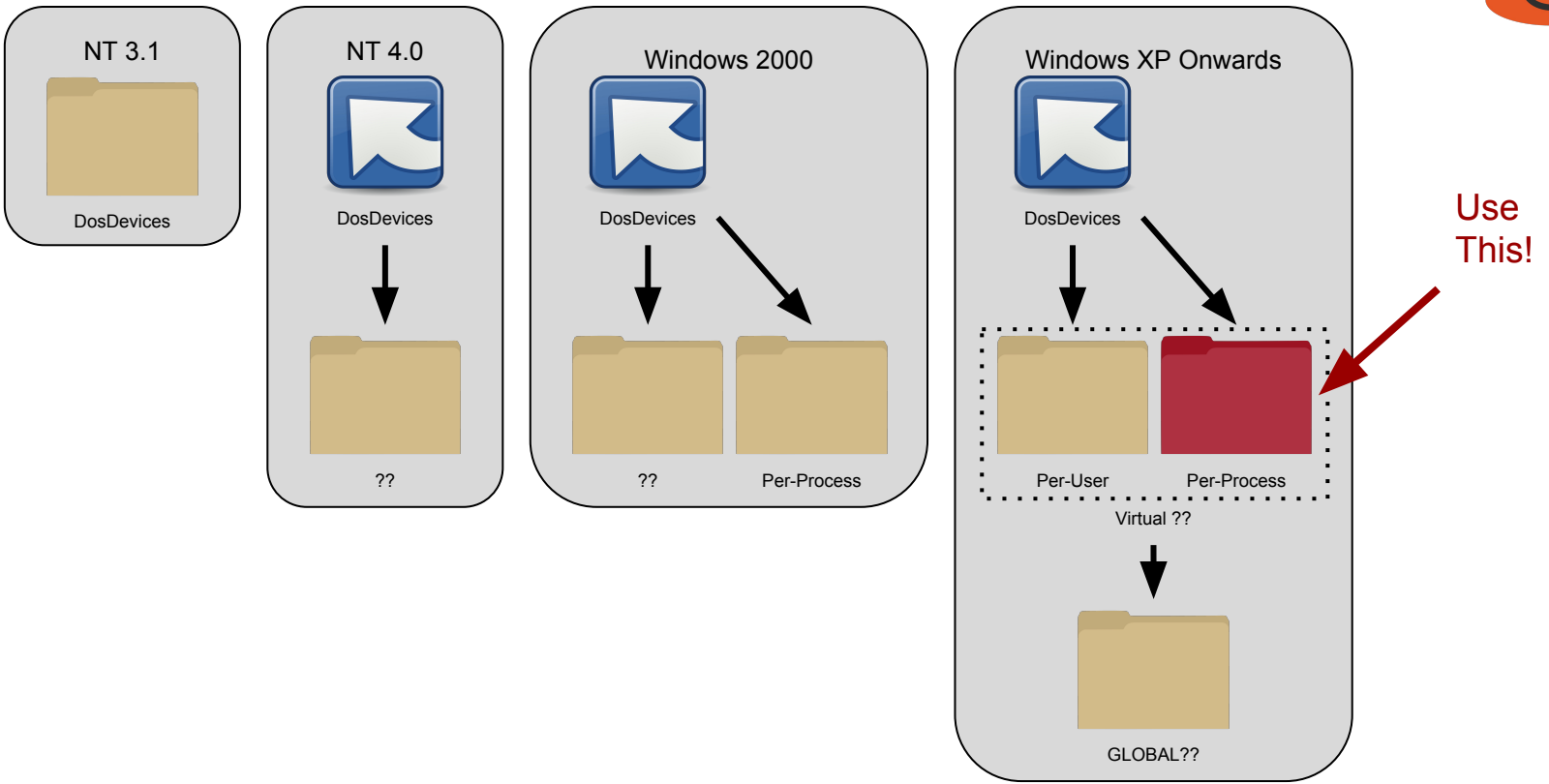
Time of check-Time of use



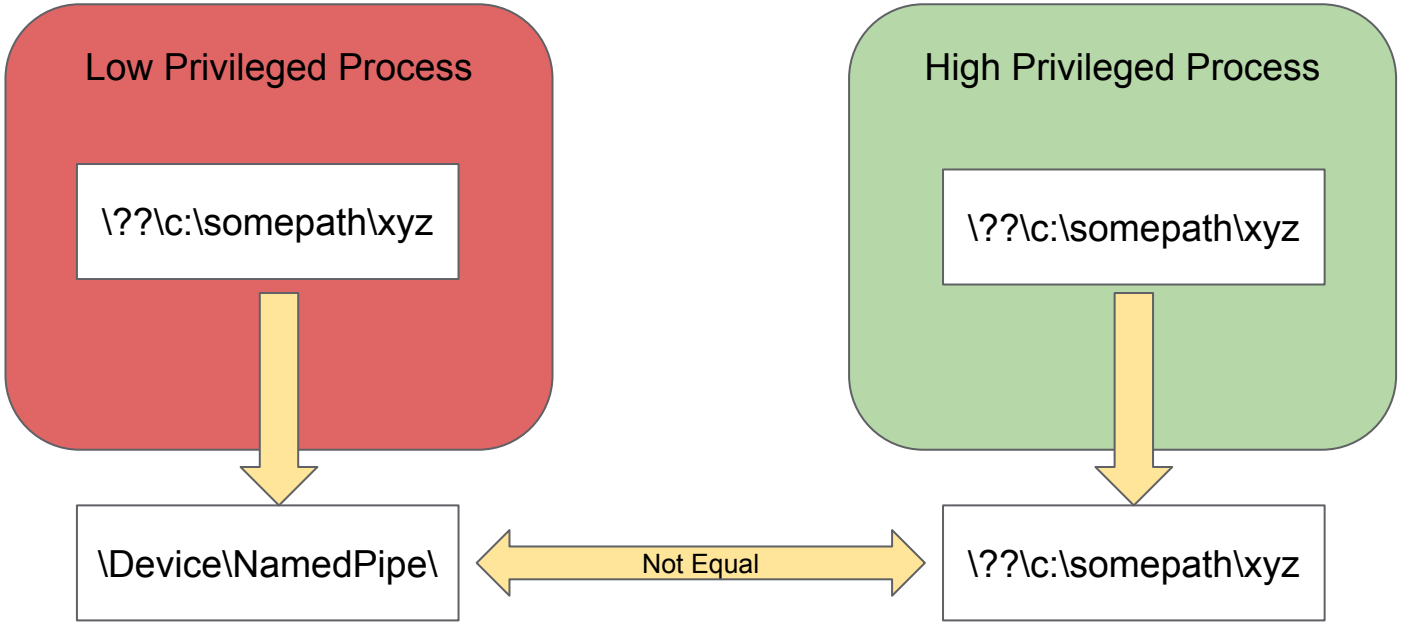
DosDevices History



DosDevices History

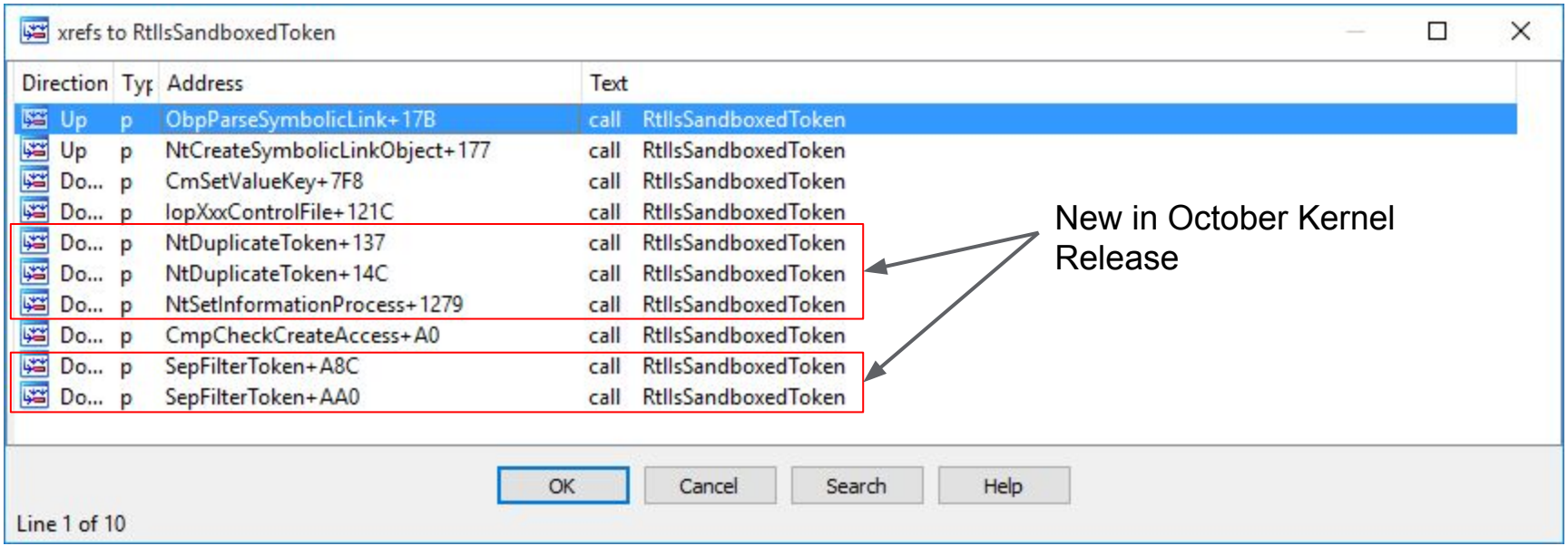


Abusing Per-Process Device Map



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

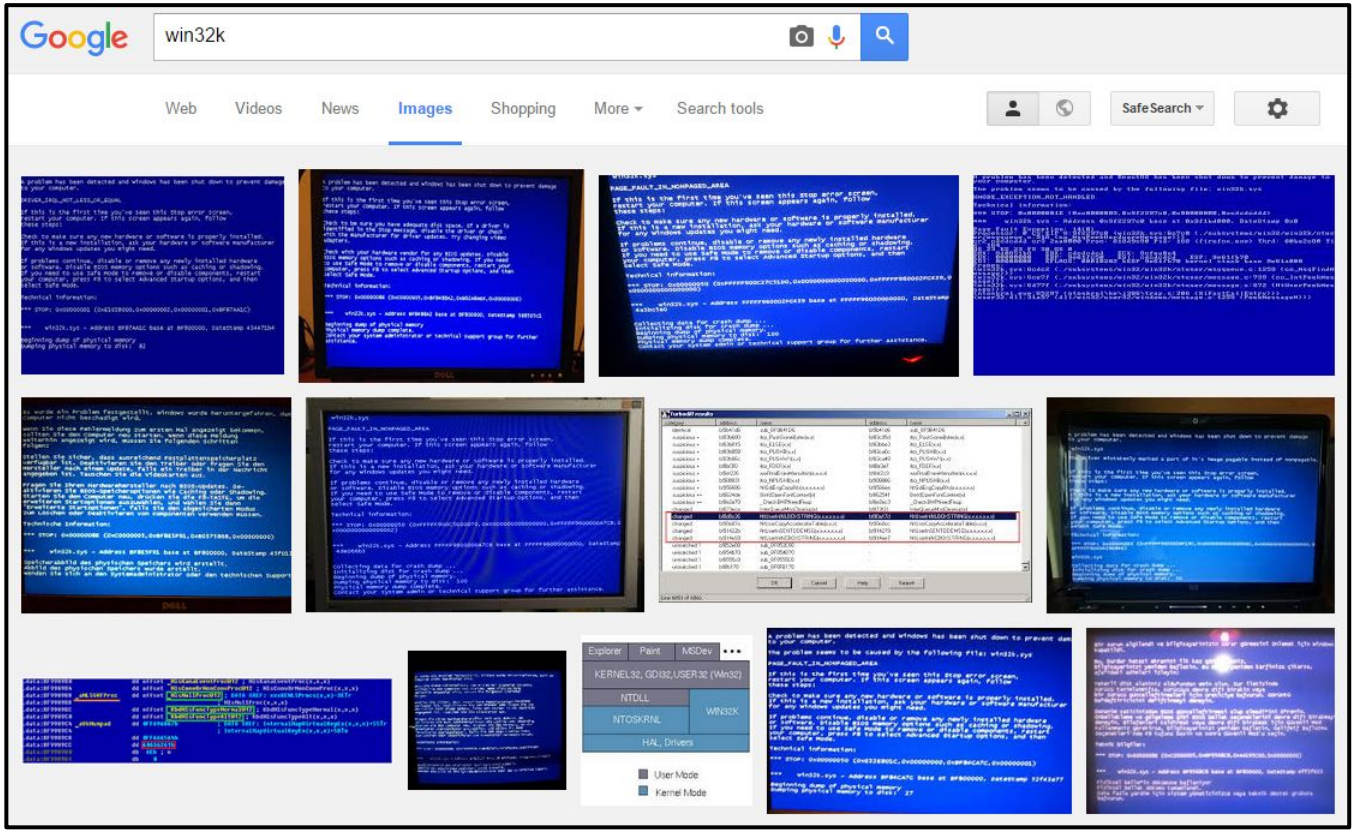
Sandbox Winter is Coming!



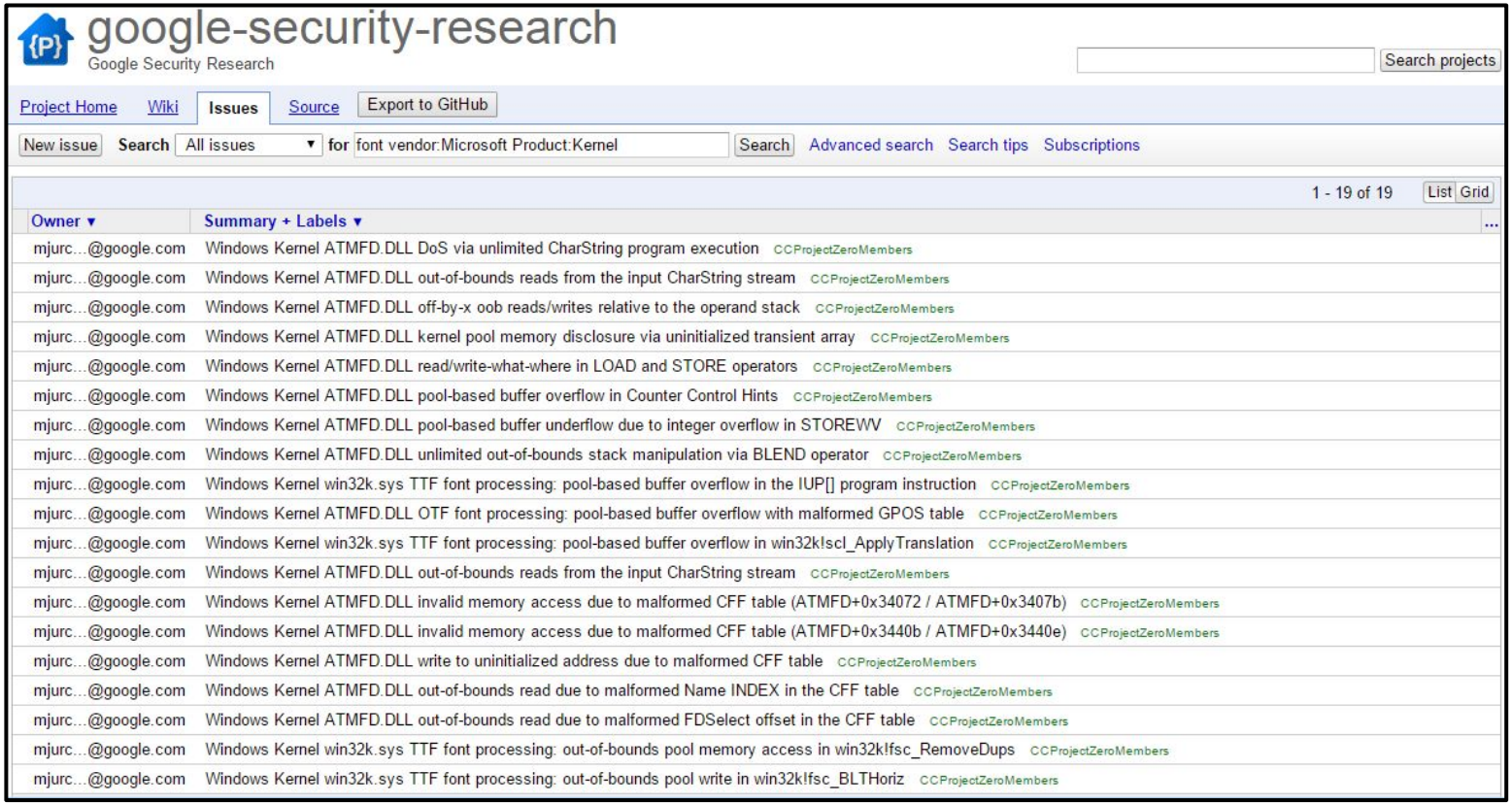
DEMO

NTFS Mount Point Mitigation Bypass

Win32k Hardening



Fonts Are Bad



google-security-research

Google Security Research

Search projects

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

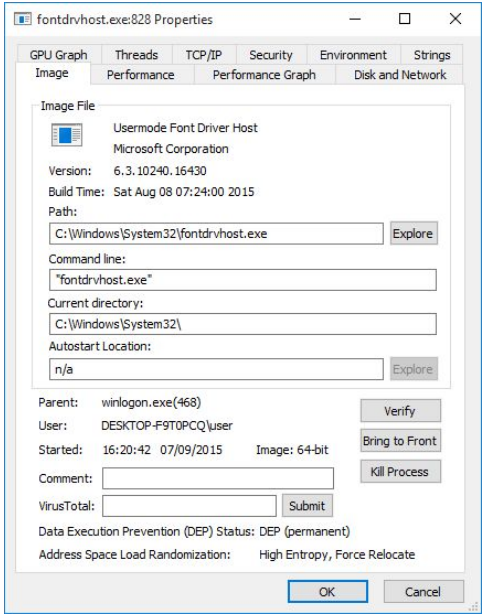
Owner	Summary + Labels
mjurc...@google.com	Windows Kernel ATMFD.DLL DoS via unlimited CharString program execution CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL out-of-bounds reads from the input CharString stream CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL off-by-x oob reads/writes relative to the operand stack CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL kernel pool memory disclosure via uninitialized transient array CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL read/write-what-where in LOAD and STORE operators CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL pool-based buffer overflow in Counter Control Hints CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL pool-based buffer underflow due to integer overflow in STOREWV CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.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 ATMFD.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!scl_ApplyTranslation CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL out-of-bounds reads from the input CharString stream CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL invalid memory access due to malformed CFF table (ATMFD+0x34072 / ATMFD+0x3407b) CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL invalid memory access due to malformed CFF table (ATMFD+0x3440b / ATMFD+0x3440e) CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL write to uninitialized address due to malformed CFF table CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.DLL out-of-bounds read due to malformed Name INDEX in the CFF table CCProjectZeroMembers
mjurc...@google.com	Windows Kernel ATMFD.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)

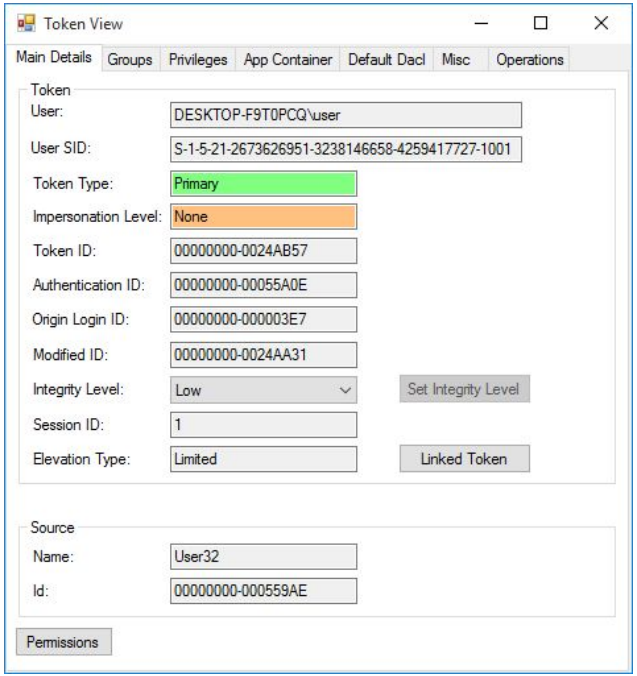
```
PROCESS_MITIGATION_FONT_DISABLE_POLICY
      policy = { 0 };

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

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

User Mode Font Driver

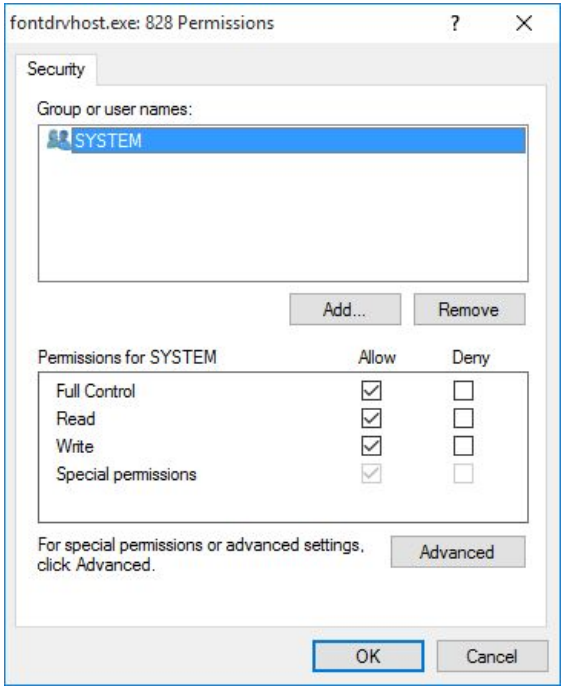


User Mode Font Driver

Running as user in AppContainer

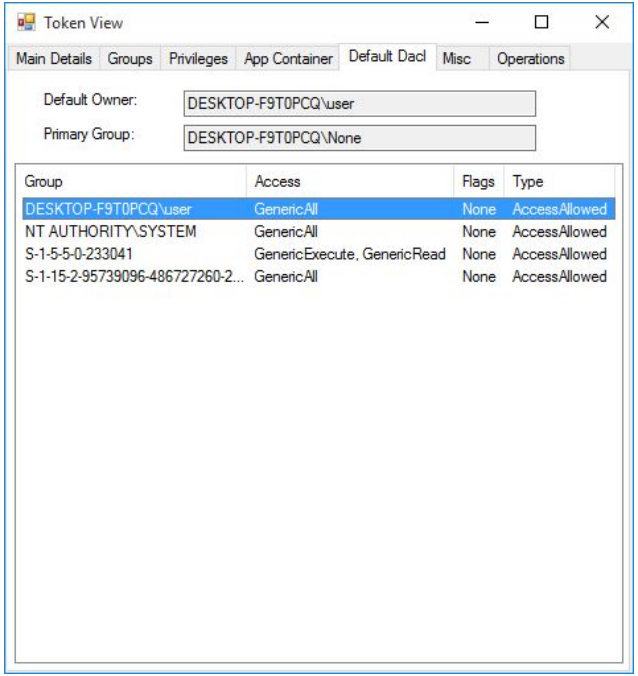


Only SYSTEM can open process?

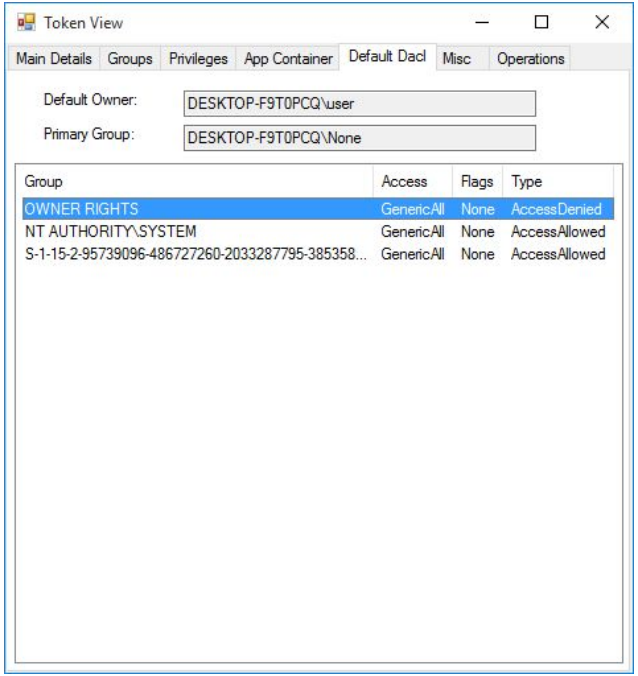


Process Token Default DACL

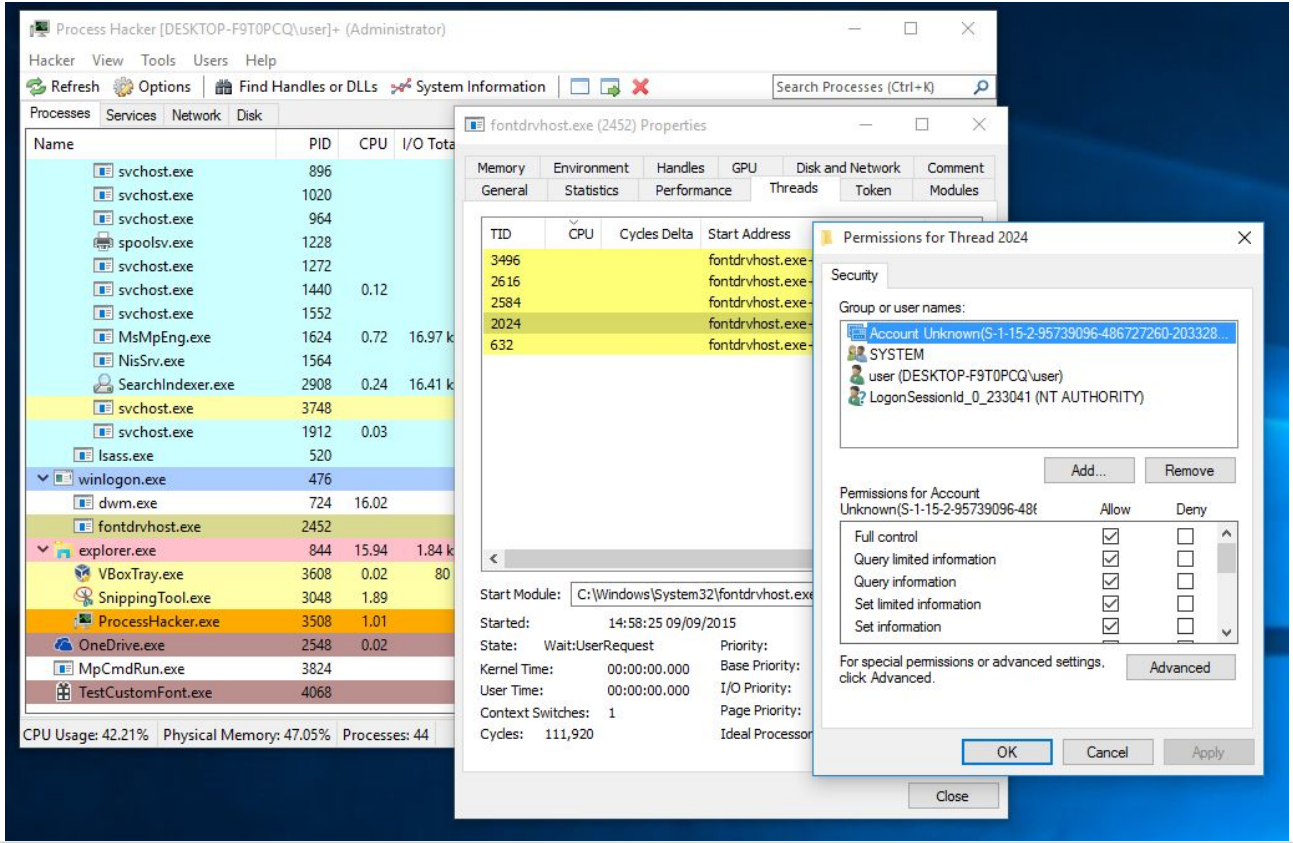
Before September Patch



After September Patch



Thread DACLs Allow User Access



Extra, UMFD Only Win32k Escape Calls

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

UmfdEscEngCreateFile 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;
}
```

UmfdEscEngCreateFile 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;
}
```

Attacker Controlled

No Security Check

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;
}
```

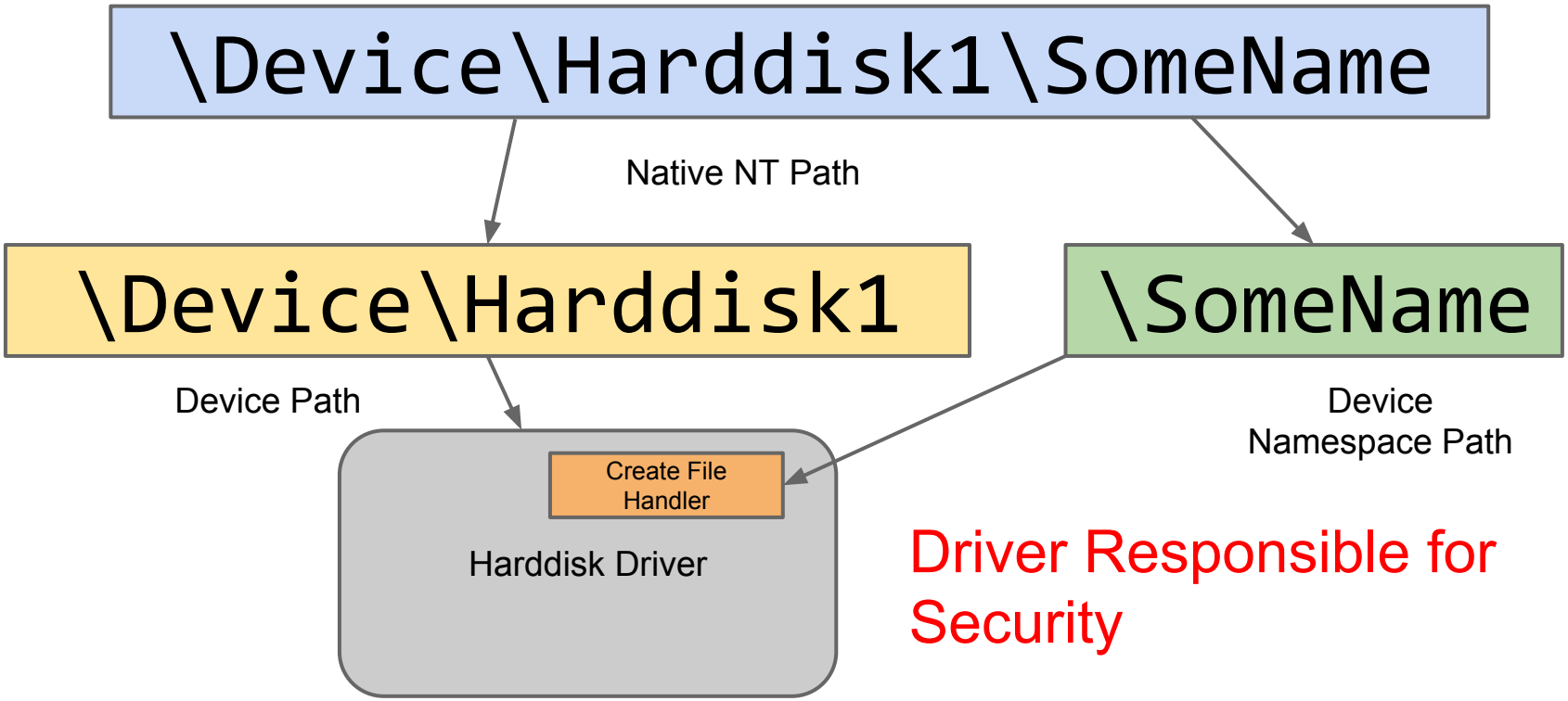
All Gone

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

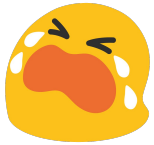
```
// Create anonymous directory object
InitializeObjectAttributes(&ObjectAttributes,
                           NULL, 0, NULL, NULL);
NtCreateDirectoryObject(&hDir, &ObjectAttributes, ...);

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

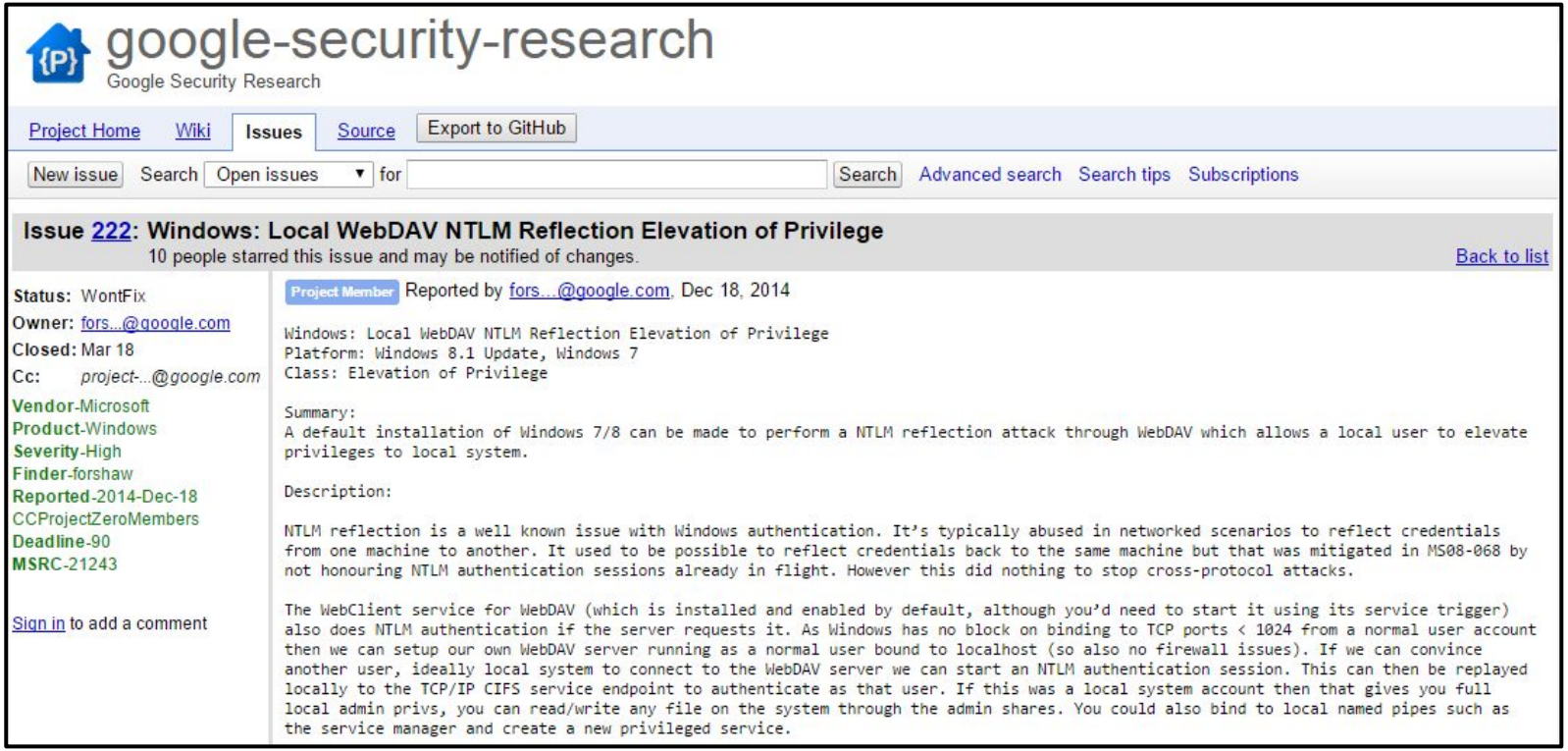
Conclusions

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 | Open issues | 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

Status: WontFix
Owner: fors...@google.com
Closed: Mar 18
Cc: project-...@google.com
Vendor-Microsoft
Product-Windows
Severity-High
Finder-forshaw
Reported-2014-Dec-18
CCProjectZeroMembers
Deadline-90
MSRC-21243

Sign in to add a comment

Project Member Reported by fors...@google.com, Dec 18, 2014

```
Windows: Local WebDAV NTLM Reflection Elevation of Privilege
Platform: Windows 8.1 Update, Windows 7
Class: Elevation of Privilege

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.

Description:

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.

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.
```

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

Questions?